An Intelligent Client-Centric Framework for Responsive, Privacy Conscious Personalisation

Rebekah Clarke

A thesis submitted for consideration to the University of Dublin, Trinity College

in fulfilment of the requirements for the degree of

Doctor of Philosophy (Computer Science)

Thesis Submission: September 2019

Approval by Council: February 2021

Declaration

I declare that this thesis has not been submitted as an exercise for a degree at this or any other university and it is entirely my own work. I agree to deposit this thesis in the University's open access institutional repository or allow the library to do so on my behalf, subject to Irish Copyright Legislation and Trinity College Library conditions of use and acknowledgement.

Rebekah Clarke

Dated: February 02, 2021

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this Dissertation upon request.

Rebekah Clarke

Dated: February 02, 2021

Acknowledgements

I would like to thank my supervisor Dr. Vincent Wade for his help and guidance which has made this work possible.

To Dad, for his unconditional support and encouragement, for keeping me going when things were tough. To my sisters, Emma and Rachael, for being there, whenever and wherever I needed them.

To Niall, whose advice and assistance were invaluable, for being a cheerleader every step of the way. To Brendan, for keeping me grounded and to all of my other friends who provided respite in trying times.

To Kenny, for enduring thorough tests of his patience and understanding. For providing the constant source of happiness which got me through.

Finally, to mum, my biggest supporter and advocate, not only during this PhD but throughout my life. This is dedicated in your memory, I know that you would have been tremendously proud.

Abstract

Personalisation is used extensively to improve user engagement, to optimise user experience and to enhance marketing and advertising online. However, it inherently requires the collection of users' personal information. In recent years, privacy has become a core consideration of consumers, providers, and governments. This is due to rising unauthorised data breaches as well as increasing awareness of the value and private nature of data. Many now market their products and services based on privacy and Governments have also recently begun to take action, for example, through GDPR. Users are becoming less likely to share or give their personal information to private companies, who are then limited in the level of personalisation they can provide. This has resulted in a personalisation-privacy paradox. The more information there is about a user, the better the system can adapt to the user's needs, but the less the privacy of the user's personal data is protected.

Privacy-conscious web frameworks such as Client-Side Personalisation (CSP), attempt to shift the data storage approach, storing user data with trusted third parties or on the client's own device. This keeps the user data under the control of the client, allowing users to enjoy personalised content without compromising the privacy of their personal data. However, this comes with a number of issues such as intellectual property concerns,

limited inference data, infeasible data overheads, restrictive client processing power and the inability to easily sync profiles across devices. Architectures have been proposed which tackle most of these issues, however, these solutions still have significant problems with scalability and performance.

This thesis introduces the Intelligent Client-Centric Personalisation (ICCP) framework. This builds on existing frameworks augmenting them with predictive prefetching to tackle performance issues and reduce user latency. This responsive, lightweight, privacy-conscious approach to personalisation enables companies to continue offering personalisation services while ensuring that consumers hold onto and have control over their personal data.

The research detailed in the thesis examines the design challenges, privacy and performance trade-offs of an ICCP framework through extensive examination and discussion. The thesis evaluates the framework, focusing on the effect of the prefetching strategy on user latency as well as the effect of the framework configurations on this latency. Ultimately, the thesis outlines the circumstances under which an ICCP framework provides marked benefits over alternative frameworks.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

"Investigating the performance of a microservice-based, privacy conscious, web personalisation architecture"

1.1 Motivation

Personalisation is used extensively to improve user engagement, to optimise user experience and to enhance marketing and advertising online [1]. It is becoming ever more important in the delivery of timely, contextually aware information, with websites capable of recomposing and adapting content on-the-fly for the user. In 2011, 20-30% of Amazon purchases and 60% of Netflix views were the result of personalised recommendations [2].

However, there is growing concern by both users and legislators over privacy on the web. This concern is particularly seen in the personalisation domain, which inherently requires collection of users' personal information [3]. The EU recently adopted the General Data Protection Regulation (GDPR), aiming to give control to individuals over their personal data. A similar act, California's Consumer Privacy Act (CCPA), will go into effect in 2020 [4, 5] and there are numerous other regulations under consideration in the US, China, Brazil and many other countries [6]. Regardless, companies continue

to employ personalisation techniques as these have consistently shown to increase both user engagement and economic returns [7]. For users, personalising webpages to their individual needs makes browsing more convenient, efficient and relevant. Thus, there is a conflict between privacy and personalisation; the more information a system has about a user, the better it can adapt to their needs but the less the privacy of the user's personal data is protected. This situation has been referred to as the ***personalisation-privacy paradox*** [8]. While privacy has always been an issue in personalised websites, only recently have we seen a noteable change in consumers' behaviour. The personal information harvested, stored and shared by content providers [7] has become more apparent through regulations and users are seeing frequent consumer data breaches. As users realise how exposed they are to personal information leakage, they are increasingly adjusting privacy controls [9], thus negatively impacting the effectiveness of personalisation services. As personalised content is essential to the success of many online providers, securing customer privacy and therefore trust is necessary for the future of personalisation.

Privacy-conscious web architectures such as ***Client-Side Personalisation*** (CSP), attempt to shift the data storage approach, storing user data with trusted third parties or on the client's own device [10, 11, 12, 13]. This keeps the user data and user model under the control of the client, allowing users to enjoy personalised content without compromising the privacy of their personal data. CSP approaches successfully reduce the leakage of the user's personal information, gaining some privacy on their behalf. Pure client-side CSP solutions propose not only storing user data on the client, but also performing the personalisation of webpages at the client-side. This comes with a number of issues such as intellectual property concerns, limited inference data, infeasible data overheads, restrictive client processing power and the inability to easily sync profiles across devices [12]. Distributed CSP architectures propose tackling most of these issues by utilising trusted external services to perform operations, while maintaining control and storage of the data on the client. However, both solutions have significant problems with scalability and performance. In particular, where the personalisation is

not purely based on the server-side, previous research has identified latency issues as a significant factor in the criticism of such approaches. Thus, techniques which could both address the latency issue whilst enhancing privacy could be of significant benefit to the user and their experience of the personalisation. As client-devices are resource limited, they struggle to handle the client-side processing requirements for performing and co-ordinating personalisation and data inference. With an ever-increasing demand for rich multimedia, particularly on more lightweight mobile devices, improving performance is critical to provide a seamless user experience.

This thesis explores a responsive, lightweight, privacy-conscious personalisation approach, termed ***Intelligent Client-Centric Personalisation (ICCP)***. This aims to enhance the performance of current distributed CSP approaches through **predictive prefetching**. The framework is microservice based, employing trusted third-party services for personalisation and user modelling, while maintaining data storage on the client. Maintaining user privacy in an ICCP framework requires a series of principles to be followed:

- User data must be stored exclusively on the client-side
- User data can only be shared explicitly with trusted service providers
- None of these external services may store any user data

This enables privacy while also enhancing performance, service scalability and protection of content provider's intellectual property.

The performance improvement of this privacy enhancing framework is achieved through the use of predictive prefetching i.e. predicting and caching the next page the user will request. Client-Side web caches are widely used to improve the performance of web applications by keeping resources likely to be used soon on the client device. Augmenting these client-side systems with the ability to predict which webpage a user will request next, allows the necessary content to be prefetched and stored on the client before the user has actually requested a webpage. This predictive prefetching strategy reduces

the lag experienced by a user browsing the web. The research detailed in this thesis proposes that incorporating predictive prefetching with a distributed CSP framework, could realise the privacy offered by a client-side personalisation approach while enhancing the performance of existing techniques. For this framework to be effective, the system must balance predictive accuracy and performance. This will enable it to determine **which** webpages to prefetch and personalise, as well as **when** to do this.

There are many techniques seen in the literature to perform this prediction, differing in the algorithmic approach as well as the data utilised, for example, predicting based on the user's browsing history, their interests or their click patterns. For this research, the focus is in exploring the effect of the predictive prefetching strategy on latency, thus, the framework should be examined under heavy load. This research chooses, therefore, to focus on a more complex user model trained using machine learning techniques. ***User interaction behaviour*** such as clicks and mouse movement data is utilised to emulate high load on both the framework and the microservices. In this research, utilising user interactions to inform the prefetching strategy is termed ***'Propensity Prediction'***. The prefetching strategy attempts to predict what page will be requested next by examining the propensity or tendency of different users to interact with different elements of the webpage.

This research aims to examine the framework described, exploring privacy as well as performance in terms of user latency.

1.2 Research Question

The question which this research seeks to answer is:

> *"What are the design challenges, privacy and performance trade-offs, for a privacy-conscious web architecture using a distributed client-side personalisation approach, augmented with predictive prefetching?"*

Privacy in the context of this thesis specifically outlines two aspects of the user model, (i) that the user model is not stored anywhere other than the client and (ii) the user data can be shared with trusted 3rd parties to execute a personalised service being used by the user.

The design assumption inherent in this research question is that the addition of predictive prefetching to a distributed CSP will improve the latency of such a system. The research proposes that a personalisation framework with this addition will address the latency problem and still enhance the privacy by strictly implementing client side data storage. This hypothesis is evaluated in the iterative design of the framework, outlined in Chapter 3.

To address this question, the research detailed in this thesis will examine both the design challenges and privacy trade-offs of an Intelligent Client-Centric Personalisation framework through extensive examination and discussion. The privacy of the framework is discussed in terms of user information leakage and personal information sharing.

A quantitative evaluation of the framework's performance is then performed. Specifically, in evaluating the performance of the ICCP, this thesis takes a user-oriented approach focusing on user latency. As the success of most online services rely upon user satisfaction, evaluations of such systems employ user-oriented metrics. User latency, i.e. the time it takes a webpage to load for a user, is seen as an essential metric, frequently used in the literature to evaluate online systems [14, 11, 15]. This research will examine the effect of two elements of the framework on this latency:

1. The effect of the prefetching strategy on user latency, looking specifically at the predictive accuracy

2. The effect of the framework configurations on user latency. Specifically, the prefetch and activity thresholds which are introduced in Chapter 3

The privacy and performance trade-offs of the ICCP framework will be discussed in comparison with a traditional client-server approach, as well as with a distributed CSP framework. This comparison aims to motivate the use cases for appropriate application of an ICCP framework.

1.2.1 Research Objectives

In order to answer the above research question, the following objectives were formed:

(i) Perform a State of the Art review of Client-Side Personalisation to determine technologies, techniques and approaches, identifying the privacy and performance challenges of each

(ii) Identify the design requirements for an ICCP framework, researching the design orchestration and algorithm selection challenges with microservice-based prediction and personalisation

(iii) Identify and evaluate the privacy and performance trade-offs in an ICCP framework

(iv) Evaluate the predictive accuracy of the propensity prediction

1.3 Contribution

This research provides two major contributions and one minor contribution to the domain as follows:

Major Contribution:

- The novel introduction of prefetching into a Distributed CSP architecture and the comparative evaluation and analysis of the effects of this prefetchixng on user latency.

Minor Contributions:

- A novel combination of evaluation techniques and metric specification which focus on user latency across multiple frameworks

- The examination of the ability of mouse dynamic data to predict which webpage a user will request next and its improvement over a comparative baseline

1.4 Publications from this Research

Three peer-reviewed conference papers were published on the research presented in this thesis. This are outlined below.

The first publication focused on the prediction of propensity addressing research objective (iv). It examined the trade-offs between latency and predictive accuracy:

- Rebekah Storan Clarke. *"Propensity Modelling for Intelligent Content"* In Adjunct Publication of the 25th Conference on User Modeling, Adaptation and Personalization - UMAP '17, pages 119-120, Bratislava, Slovakia, 2017. ACM Press.

The design and design challenges for an ICCP framework was then examined through a publication in Web Intelligence, addressing research objective (ii). Specifically, this focused on trade-offs between the frequency of requests to the PPS, the accuracy and the system latency:

- Rebekah Storan Clarke and Vincent Wade. *"Intelligent client-side personalisation"* In Proceedings of the International Conference on Web Intelligence - WI '17, pages 1218-1221, Leipzig, Germany, 2017. ACM Press.

Finally, a paper published in ICWE focused on objective (iii), examining and presenting the results of the ICCP framework evaluation presented in Chapter 6 of this thesis:

- Rebekah Storan Clarke and Vincent Wade. *"Distributed Intelligent Client-Centric Personalisation"* In Proceedings of the International Conference on Web Engineering - ICWE '19, volume 11496, pages 498-505, Daejeon, South Korea, 2019. Springer International Publishing.

1.5 Methodology

The methodological approach of the thesis is to first explore the current state of the art in client-side personalisation and examine the strengths, weaknesses and technical approaches. The investigation will examine the architectural design of several state of the art systems, examining the location of personalisation & user modelling operations as well as the user data privacy offered by each approach. Based on this analysis, the thesis will design and develop an ICCP framework. The ICCP framework uses predictive prefetching to enhance the performance of current distributed CSP approaches. In an ICCP, the prediction training and personalised re-composition are achieved through trusted third party services.

To evaluate the framework, this thesis performs a comparative evaluation, utilising comparative analysis methods. The intelligent client-centric personalisation framework is evaluated across privacy and performance against a typical client-service approach and a distributed CSP without prefetching. A large dataset of real user interactions is collected across three contrasting websites. The evaluation specifically compares the performance of each framework in terms of user latency i.e. webpage load time. This is discussed along with the privacy benefits of each framework to establish relative trade-offs. Finally, the effect of framework configurations and the predictive accuracy of the prefetching strategy are evaluated. The predictive accuracy evaluation examines how well a user's next page request can be predicted by the interaction behaviour data utilised in the research. This calculates the log loss for the predictions made against what page the users actually requested next.

1.6 Thesis Structure

This thesis is ordered sequentially to ease comprehension of the research, however an iterative approach was followed developing each aspect of the research in parallel. This is further outlined in section 3.1.

State of the Art To accomplish the research objectives identified above, it was necessary to provide an overview of approaches to Privacy Conscious Personalisation with a specific emphasis on architectural solutions. Aiming to identify the technologies, techniques and the challenges in such systems, four State of the Art systems are examined and compared. This highlights the performance challenges in existing systems, identifying a gap in current research and framing the contributions of this research. Following this, background research is necessary to inform the design of the end system. A review of approaches to both Intelligent Web Caching and Interaction Based Prediction is provided giving insight into the foundations and current trends in these areas. These sections provide a basis for discussion around the design of the ICCP framework in particular framing the design of the predictive precaching solution established in chapter 3.

Design and Implementation Chapter 3 starts by using the state of the art research to identify the core requirements for the design of an Intelligent Client-Centric Personalisation (ICCP) Framework. A minimal viable product approach was followed to build the system testing the design choices and weighing the trade-offs of each. These requirements are presented in contrast to the existing SOA systems examined in the previous chapter. A more detailed architecture for the framework is then presented, describing the components required to realise the established requirements and incorporate predictive prefetching with a distributed CSP (DCSP) architecture. Finally, the implementation of the ICCP framework is outlined, discussing technical challenges and solutions.

Research Methodology Chapter 4 outlines the research methodology for a comparative performance evaluation of the ICCP. The selection of two frameworks used for this comparison is described and their design examined, namely these are a client-server framework and a DCSP without prefetching. The experimental setup is then outlined, explaining the streaming approach used to replay user data through each of these three frameworks.

Data Gathering Chapter 5 details the gathering of real user data through the implementation of a reverse proxy. Data collection, validation and verification methods are discussed. Finally, analysis of the gathered data is presented.

ICCP Evaluation Chapter 6 performs a comparative evaluation of the three frameworks based on their performance. The chapter aims to evaluate:

1. The effect of the prefetching strategy on user latency, looking specifically at the predictive accuracy
2. The effect of each of the framework configurations on user latency

PPS Evaluation Chapter 7 evaluates the prediction service independently from the framework used. This aims to establish the accuracy of predictions made in the experiments conducted along with discussing the difficulties in implementing a predictive system.

Conclusions Finally, the thesis conclusions are presented and discussed in Chapter 8. This summarises the contributions of this dissertation and discusses areas for future work.

Chapter 2

State of the Art

This chapter examines the state of the art in Privacy Conscious Personalisation. Initially, a scoping literature review of the area is performed, aiming to explore research in the area, identifying the different approaches used to protect user privacy online and establishing current evidence. Finally, the review leads to the identification of research gaps, specifically current performance issues with client side personalisation. This frames the contributions and focus for this research and leads to a concentrated systematic literature review in client-side personalisation.

Having established a research question, examining the implications of augmenting distributed client-side personalisation with predictive prefetching, a more narrow search for relevant research is undertaken. Establishing a citation map in this field ensures full coverage of papers in the area. This leads to the identification of four state of the art systems for examination and comparison through relevant criteria.

Finally, the chapter also presents the state of the art in two areas required to inform the design of the end system, Intelligent Web Caching and Interaction-based Prediction.

2.1 Privacy Conscious Personalisation

There is growing concern by both users and legislators over privacy on the web. This concern is particularly seen in the personalisation domain, which inherently requires collection of user's personal information [3]. The EU recently adopted the General Data Protection Regulation (GDPR), aiming to give control to individuals over their personal data, a similar act, California's Consumer Privacy Act (CCPA), will go into effect in 2020 [4, 5], and there are numerous other regulations under consideration in the US, China, Brazil and many other countries [6].

For users, personalising webpages to their individual needs makes browsing more convenient, efficient and relevant. However, it also arises significant privacy concerns as personal data is collected and stored by an ever increasing number of content providers. Regardless, companies continue to employ personalisation techniques as these have consistently been shown to increase user engagement and purchasing [7]. An average of fourteen percent of marketing budgets are now dedicated to the pursuit of personalisation [16]. While user engagement and improved sentiment are beneficial to companies, for example motivating student learning [17] or influencing political stances [18], the connection of metrics like consumer satisfaction are hard to relate to business value. Thus, focus is shifting toward e-commerce use cases where the impact can be more tangibly measured through purchase behaviour, showing a demonstrable return on investment [16]. E-commerce currently represents about half of all personalisation engines and investment in personalisation is projected to continue growing.

Thus, there is a conflict between privacy and personalisation; The more information there is about a user, the better the system can adapt to the user's needs, but the less the privacy of the user's personal data is protected. This situation has been referred to as the *"personalisation-privacy paradox"* [8]. To address this problem, web personalisation techniques have to balance both the needs of the website user and website publishers.

2.1.1 User Attitudes to Privacy

Personalisation is critically dependent on both a vendor's ability to acquire and process information and consumers' willingness to share information and use personalisation services [19]. Despite privacy concerns, users are willing to give up some personal data if there are benefits, like convenience, to be obtained in return [19, 2]. User attitudes to privacy must be considered across two facets, both the users' stated privacy attitudes and their observed behaviours. Norberg et al. found that there is considerable discrepancy between these [20]. Users are concerned that providers collect too much information about them, suspecting their private information is being tracked without their knowledge [21, 22]. This creates negative feelings and a lack of trust in personalisation services [3].

Smith et al. [23] developed the Concern for Information Privacy (CFIP) measurement instrument, which identifies four dimensions of an individual's concern for privacy. Although some work has been done to update and augment this [24], it frames the basic threats from a user perspective as follows [23]:

- **Collection** Individuals often perceive that great quantities of data regarding their personalities, background, and actions are being accumulated, and they often resent this. The user may also be completely unaware of the collection, or it may be unsolicited.

- **Errors** Users may be concerned that incorrect inferences will be made over the data. For example, using protected attributes like race or gender to make potentially racist or sexist decisions [25], or perhaps providing an incorrect medical diagnosis.

- **Secondary Use** Sometimes information is collected from individuals for one purpose but is used for another secondary purpose without authorisation from the individual. This can be by the original service provider, by third parties who are granted or sold access to the data, or even by other users of the system. Private information could also be uncovered by un-trusted parties through inference or exploitation.

- **Unauthorised Access** Data may be made available, accidentally or otherwise, to unauthorised parties, for example through data breaches.

Despite these concerns, surveys have repeatedly shown that users do not assess privacy policies in a rational manner before agreeing to the terms and conditions of services [26]. This is probably because they feel they have little choice; if they wish to use a service, then they have no alternative but to accept the privacy policy. Foregoing use of the service because of privacy concerns does not appear to be a realistic option for the vast majority of users. Some studies frame this as *privacy calculus*, positing that an individual's intention to disclose personal information is ultimately based on risk-benefit analysis; in these cases the benefit for the user outweighs the risk. Other studies show that privacy fatigue [27] has a larger impact on observed behaviour. Users may simply be tiring of privacy prompts and hastily agreeing without reading or understanding the consequences.

More recent studies into privacy behaviour have shown that these trends are changing. Refusals to reveal information have risen over time [9]. Reasons suggested for this changing behaviour include an increase in experience with IT (and thus an awareness of potential privacy concerns), a change in the composition of the online population, and/or a change in underlying preferences for privacy. Goldfarb et al. [9] found evidence to suggest that the changing trends are partly due to broadening perceptions of the

contexts in which privacy is relevant. While users were always privacy-protective in financial and medical contexts, they are increasingly becoming privacy-protective in less apparent areas of privacy concern such as answering questions about consumer packaged goods or movies.

Studies have consistently found that users are reactive in their privacy behaviour. Context and environment [28], notifications about privacy breaches [29] and even simply mentioning privacy [30] cause users to reduce the disclosure of personal information. Several online companies (e.g. the search engine DuckDuckGo[1]) use enhanced privacy as a way of differentiating their product, or even build their business model around the protection of privacy. Users are less reluctant to provide personal information if they trust the system to preserve the privacy of their personal data.

More visibly, the issue of privacy regulation has entered the policy agenda with important challenges being raised, for instance, regarding the scope of government surveillance and the legal framework surrounding data sharing. As these regulations progress, many user concerns are beginning to be addressed. Article 5 of GDPR sets out several principles addressing the processing of personal data [31]:

- Lawfulness, fairness and transparency — Processing must be lawful, fair, and transparent to the data subject.
- Purpose limitation — You must process data for the legitimate purposes specified explicitly to the data subject when you collected it.
- Data minimization — You should collect and process only as much data as absolutely necessary for the purposes specified.
- Accuracy — You must keep personal data accurate and up to date.
- Storage limitation — You may only store personally identifying data for as long as necessary for the specified purpose.

[1] www.duckduckgo.com

- Integrity and confidentiality — Processing must be done in such a way as to ensure appropriate security, integrity, and confidentiality (e.g. by using encryption).

These regulations go towards resolving user concern over *data collection* as the user should now be aware of all data being collected along with its purpose. While *data errors* are still of concern, GDPR enforces much stricter limitations around the use of 'special case' data which includes such information as race and gender. Small steps were also taken in the realm of *secondary use*, assigning the data controller some responsibility for the use of data by a third party though user data is still readily shared. **Unauthorised access** is not substantially addressed by GDPR though perhaps less will be stored and therefore leaked in the case of data breaches.

While GDPR addresses many areas of user concern, there still exist many practical challenges to its implementation. The regulations apply only to EU member states and users must trust that these rules are being upheld and stringently enforced. There are also many concerns unanswered by the regulations.

The approach to privacy proposed in this research presents a more rigorous method of compliance and enforcement through the use of an architectural solution. In complement with GDPR legislation, this reduces the burden of trust from the user as well as tackling a larger proportion of user concerns e.g. unauthorised access and secondary use.

As personalised content is essential to the success of online providers, particularly in e-commerce and marketing, securing customer privacy and therefore trust is necessary for the future of personalisation. Privacy considerations also extend to the service provider, as the algorithms used may be subject to intellectual property concerns. A personalisation system aiming to preserve privacy should protect the privacy of the user and also the intellectual property of the service provider, while also balancing requirements regarding performance, security and personalisation quality.

2.1.2 Proposed Solutions

A range of techniques have been investigated aiming to provide personalisation without unduly compromising users' privacy. These can be grouped into four broad categories [32]:

1. Algorithmic Solutions

   Data modification approaches and formal guarantees that ensure that leaked personal data discloses only modified or encrypted information.

2. Legislative Solutions

   Policies and regulations, which may be imposed by governments and legislative bodies, or adopted as self-regulatory industry practices.

3. User-centric Solutions

   User behaviours, attitudes, and decisions related to the disclosure of their personal data.

4. Architectural Solutions

   Software architectures, platforms, and standards designed to minimise personal data leakage. These techniques are the focus of the research detailed in this thesis.

The following sections will discuss each of these solutions in depth, giving examples of systems in each area.

2.1.2.1 Algorithmic Solutions

There are several algorithmic techniques for data protection found in the literature, for example, anonymisation, pseudonymisation, differential privacy, and homomorphic encryption. These aim to ensure that any leaked data will disclose only modified or encrypted information about users.

Data Anonymisation

This performs information sanitation either through concealing the relationship between a particular user and their personal data, or releasing only encrypted data to ensure it cannot be meaningfully interpreted if accessed by unauthorised parties. To protect users' privacy through anonymity, several types of anonymity must be present in a user-adaptive system [33]:

- **Environmental anonymity** is determined by external factors, including the number and diversity of users in a system. For instance, if there is only a single user of an app in a specific location then their data can be linked to them from the location alone. User adaptive systems should be enabled to detect environmental situations critical to anonymity, however mitigating such issues must be handled by means outside of the system [34].

- **Procedural anonymity** is determined by the communication protocol and the underlying communication layers. This ensures that the design of a system has sufficient anonymisation protections in place. For example, denying unauthorised components access to certain information about a user. Encryption, a data security technique, can be used to ensure procedural anonymity, protecting personal data from inspection when it is being exchanged between the user model and its clients. Using an appropriate cryptographic system, the authorised recipients of the information can also be specified in the encryption process.

- **Content-based anonymity** is achieved by the system ensuring that the content of personal information cannot be meaningfully linked to reveal the identify of the user. For example, one study [35] found that linking users' contacts and interests could reveal the user's sexual orientation, even when this was in contrast to their reported orientation. Obfuscation [36] is one technique used to increase content-based anonymity, it involves deliberately degrading the quality of information in some way, like reporting a user's location with an intentional ambiguity of ± 500 metres.

However, real user data is required to perform accurate data analysis by third parties. Several techniques tackle the issue of enabling access by third parties without sacrificing user privacy, for example Pseudonymisation and Differential privacy.

Pseudonymisation

Pseudonymisation is a technique used widely in healthcare [37, 38] to enable analysis of personal data and its use by secondary parties, without encroaching on user privacy. Sensitive information in a dataset is replaced by one or more artificial identifiers, or pseudonyms. These cannot be associated with the real values without knowing a certain secret. For example, a public dataset may reveal that 10% of users spend 50% of their time on the same site, but not reveal which site or how much time this constitutes.

Differential Privacy

Dwork et al [39] proposed differential privacy to allow patterns to be extracted from datasets and reported without revealing information about individuals. This is achieved by constraining the algorithms used to publish aggregate information about a statistical database. For example, differentially private algorithms are used by some government agencies to publish demographic information or other statistical aggregates, while ensuring confidentiality of survey responses. The primary mechanism to achieve differential privacy is to add random noise to the aggregate data, without obscuring the aggregate statistics. Dwork [39] proposes mathematical proofs on how much noise is enough to achieve the requirement of differential privacy.

Homomorphic Encryption

To enable privacy during collaborative filtering Canny [40] utilises homomorphic encryption. The paper investigates a method which allows a community of users to compute a public "aggregate" of their data that does not expose individual users' data. Homomorphic encryption is applied to allow sums of encrypted vectors to be computed and decrypted without exposing the individual data of users.

While algorithmic solutions can provide great results, there is a trade off between privacy and the quality of the user data provided for personalisation. Often, these approaches are used in conjunction with other privacy solutions to retain a high quality of user data.

2.1.2.2 Legislative Solutions

In April 2016, the European Union (EU) passed the General Data Protection Regulation (GDPR). This introduced new legal requirements for the protection of personal data for data controllers operating within the EU territory [41], replacing the Data Protection Directive 95/46/EC (DPD) introduced in 1995. The rapid change in data landscape caused by the explosion of ubiquitous and mobile computing and the *big data* era, had led to the necessity for legislative updates [41]. The new regulation aimed to *"protect fundamental rights and freedoms of natural persons and in particular their right to the protection of personal data"*. GDPR strengthens the well-established data protection principles already specified in DPD, like consent and purpose limitation, and encompasses new principles such as the right to be forgotten, the right to data portability, the obligation for data protection impact assessments, and privacy by design, among others. Compared to the earlier DPD, GDPR also significant increases the penalties for breaches of privacy.

Regulations such as these are being considered across the globe [6], California's Consumer Privacy Act (CCPA), will go into effect in 2020 [4, 5] and there are numerous other regulations under consideration in the US, China, Brazil and many other countries [6].

These regulations force content providers to consider the privacy of users and implement effective measures to protect their data. The major drawback to legislative approaches to privacy is the length of time they take to become law. These can also face significant challenges and appeals in courts of law before legal precedent becomes established. Another major drawback is the significant amounts of corporate lobbying used to curtail or dilute the effectiveness of such legislation.

2.1.2.3 User-centric Solutions

User-centric solutions to privacy inform user attitudes and behaviour through privacy decision support. The now defunct Open Profiling Standard [42] set out core guiding principles for sharing data online:

1. Access to information should be controlled by the source of that information
2. Informed consent of the source must be obtained before the collection and use of information
3. No party should collect information without offering the individual value in exchange

These principles are reflected in the more recent area of **Open User Modelling**. These allow the user to retain full control of what they disclose to whom, and can be controlled in an automated manner or through explicit user action [32]. Especially related to increasing trust, different studies have indicated a positive effect if the users are given the opportunity to inspect their user model and to opt out of logging and/or personalisation [43]. Open User Models also provide a unified view of the user's personalisation expectations, relieving each individual site from keeping up with this. More significantly, it can relieve the user from the burden of editing privacy settings for every entity they interact with by automating this process based on their preferences.

However, model scrutiny can prove to be challenging due to the large quantity of prompts and decisions which the user must make. This can be addressed through visualisation techniques along with well defined defaults. Several studies explore methods of providing privacy settings in an intelligent way [44, 45, 46, 47]. These automatic settings aim to reduce the manual effort required by the user. For example, one study [46] proposes 'Trusted Virtual Domains' which employs a uniformly enforced security policy across a coalition of machines. Thus privacy settings for this entire coalition can be granted by the user in a single interaction.

When users can control exactly what they share, websites must provide an appropriate 'value exchange' for additional access to the user's personal information. However, as previously discussed in section 2.1.1, users often eschew the hassle of changing privacy settings [48], have misconceptions about the implications of these settings [49], and do not follow rational economic principles in making privacy decisions [50, 51]. Interface design for policy prompts is a very active area of research. Studies [48] indicate that well-thought out prompt design can significantly improve user comprehension of policy implications. Careful use of general policies, and limited use of prompts, can minimise issues where users are simply annoyed by repeated prompts and authorise all accesses [52].

2.1.2.4 Architectural Solutions

The research described in this thesis focuses on architectural solutions to protect the privacy of user data. These solutions compliment other approaches by mediating the control and access to sensitive user information. All four privacy solutions detailed here, and in the three previous sections (Algorithmic, Legislative and User Centric) can be used together to provide the user with a high level of privacy, while still enabling the provider to perform useful and accurate personalisation.

While algorithmic approaches aim to reduce the risk after personal data is leaked, architectural approaches aim to minimise this leakage in the first place. They aim to provide limited access to and linkability of user data. In web frameworks, client-server architectures are the norm. With these, the content server controls all interactions, they gather data about users on their servers, and use it to deliver customised content and targeted advertisements. Privacy conscious frameworks aim to give the client more control over these processes. Expanding past open user models, architectural solutions not only give the user control over how data is shared, but also where it is stored and processed.

Architectural solutions to user privacy vary in three broad aspects:

- Where the user data is stored
- Where the data analysis is performed
- Where the personalisation occurs

Previous architectural solutions [13, 11] have suffered from a privacy-performance trade-off. While theoretically offering a high level of privacy to the user, ultimately the performance issues, in terms of user latency, were too great to create a viable solution. However, recent developments in Machine Learning (ML) offer promising solutions to improve the performance of these systems.

This thesis explores whether a framework which combines predictive machine learning approaches, with existing architectural solutions, can offer privacy preservation with a faster response time i.e. reduced user latency.

The following two sections will explore existing solutions and issues in greater depth; first discussing Pure Client-Side Personalisation solutions, and then exploring a Distributed CSP approach, illustrating the state of the art through exploration of four existing systems in the domain.

2.1.3 Pure Client-Side Personalisation

Client-side personalisation (CSP) originally proposed that all of the processes involved in personalisation occurred on the user's own device i.e. the client [2]. This means that none of the user's personal data is stored or acted upon by the content server. In *Pure* Client Side Personalisation:

- The user data is stored only on the client
- The data analysis is performed only on the client
- The personalisation occurs only on the client

As no remote services can access this data, the user is protected against three facets of concern; unsolicited collection of data, secondary use by third parties and unauthorised access through data breaches. CSPs are not, however, free from all privacy concerns, as the data is stored on a physical device, there remain concerns over lost or stolen devices, as well as device access through wireless network hacking. The potential security concern of data access through malicious executable code must also be considered. A dishonest service provider or a third party may try to gain unauthorised access to private information through the browser. This malicious code may be inadvertently downloaded and executed on the client device, transmitting data to an unknown and untrusted location. Modern browsers, however, incorporate a series of security measures to prevent such malware attacks, for example, blocking popups, preventing the download of potentially harmful files and blocking or flagging malicious websites in web search. A final privacy concern with CSPs exists regarding errors such as incorrect inferences made over the data, as discussed in the preceding section.

In studying user perception of privacy in client-side personalisation, Kobsa et al. [2] compared CSP approaches to three other services, Amazon, a fictitious company, and the "Cloud". Although CSP was shown to provide the most security to user data, it did not lead in perceived privacy or privacy-related behaviour. Participants also gave roughly the same amount of personal data and tracking permissions in all four conditions. However, it was also found that with added security features, such as remote locking in case of loss or theft, the privacy perception could be raised significantly. These additional measures also raise the protection offered by a CSP approach.

Pure CSP approaches [10] offer a huge privacy benefit over client-server architectures, they also reduce the legal and data storage burden on the content provider; and they provide cross-site personalisation which inherently helps the cold start problem and provides additional sources of data for personalisation. These advantage are explored in the following sections

2.1.3.1 Cross-Site Personalisation

Currently, as users jump between websites, each has fragmented information about the user's preferences and behaviours. Cross-site personalisation addresses these information needs spanning independently hosted websites [53]; it allows personalisation code from one site to interface with data collected through interaction with another. This unifies the user's browsing experience, increasing the effectiveness of the personalisation offered. With client-side personalisation, user data is stored in only one place, the client's device. This facilitates extensions to CSP allowing a single system to develop a user profile that can be applied to a broad range of digital libraries. For example, Kaplana [11] uses semantic web techniques to maintain a uniform interface to data. They show how this can be used to aid a user who may wish to construct wishlists across e-commerce websites. If that wishlist were accessible by every e-commerce site visited, they could offer competing deals on the items.

Cross-site personalisation also inherently tackles the 'cold start' problem. When a user visits a site for the first time, personalisation cannot be immediately provided due to lack of data. Thus a model must be bootstrapped to the new user. With cross-site personalisation a model already exists for the user from interactions with other websites. This means that the cold start problem is an issue only for a completely anonymous user rather than each time a user visits a new site.

Cross-site personalisation can, however, also be a disadvantage to content providers. Websites may not want to allow other sites to use data collected by them as it may cause them to lose a valuable competitive advantage. There are also legal liability issues with sharing data as in some jurisdictions, like the EU, the site collecting the data i.e. the data controller would be liable for third party usage. A collaborative approach would be required to realise effective cross-site personalisation, establishing a system based on trust which incentivises providers to share complete and accurate data. Such trust issues

have been previously addressed and overcome to enable corporate data sharing, where key stakeholders subscribe to be part of an alliance. Cross-site personalisation must be designed with both the needs of the user and the content provider in mind. Allowing the provider some control over the information shared.

While pure client-side CSP approaches [54, 55] offer maximum protection of user data and can facilitate cross-site personalisation, they also face several challenges, namely:

- Processing power
- Proprietary concerns
- Limited inference
- Large data burden on the client
- Cross-device personalisation

Each of these issues is explored in detail below:

2.1.3.2 Processing Power

Performing both the data analysis and the personalisation on the client's own device puts a large processing burden on the client. This may significantly impact performance [13], particularly on the less powerful mobile phone devices which are increasingly being used by consumers. As the memory footprint of the state of the art algorithms continue to scale up [2], processing the data analysis and the personalisation on the client device becomes an increasingly impractical solution. In particular, deep learning, known to be data-hungry and computationally expensive, is increasingly being used in the personalisation domain. At the same time, consumer devices are becoming increasingly powerful, with the rise in GPUs and edge computing both enhancing performance. However, modern personalisation techniques are becoming heavyweight at a faster rate necessitating research into producing compact models [56], as well as compression techniques [57].

2.1.3.3 Proprietary Concerns

CSPs also raise proprietary concerns as the personalisation logic must be delivered to the client device in order to perform personalisation for the client. This code often includes confidential algorithms and is at risk of exposure through reverse engineering. A contemporary approach to address this is code obfuscation [58], which makes reverse engineering very difficult. This uses minification or uglification to create Javascript code which is difficult for humans to read but can still be executed by the browser. However, obfuscation has been proven as insufficient [59] and can never perfectly protect proprietary algorithms.

2.1.3.4 Limited Inference

The inference methods that can be used are limited by having all of the data on the client device [2]. Predictive methods which are based on the whole user population such as collaborative filtering and stereotyped learning, collect large amounts of information about their users in a central repository to find regularities that allow for future recommendations. These central repositories however may constitute attractive targets for unauthorised access, and could also be mined for individual user data by asking for recommendations using cleverly constructed profiles. Thus, these techniques can only be used if conducted in a separate privacy aware manner. For example, carrying out data collection with users who willingly share their information for market research. Additional protections could be offered during this data collection, such as anonymisation or homomorphic encryption.

2.1.3.5 Data Burden

Ideally in CSP, all possible personalisation content would be loaded onto the client. This avoids content being fetched individually when needed, which may indirectly reveal the user's personal information. For example, when serving ads to a user based on their preferences, separate requests for specific ad content may allow the server to construct a model of the user based on the known categories of each ad. Both PrivAd [60]

and Adnostic [61] preserve privacy by performing all behaviour tracking on the client, downloading all potential advertisements from the advertiser's servers, and selecting the appropriate ad to display locally on the client. However, as a website scales up in size, it becomes infeasible to fetch the entirety of the available content, e.g. the client could not possibly fetch all of the millions of books that Goodreads[2] may recommend to a user to read next.

2.1.3.6 Cross-Device Personalisation

Another drawback of CSP approaches is in the use of multiple devices. Using traditional approaches such as Dropbox[3] to synchronise data across devices causes privacy concerns to arise. Although the data is now shared with a single provider, which can be a selected trusted provider, this compromises the privacy of CSPs, as data is no longer strictly local.

2.1.3.7 Pure CSP Summary

From a technical and conceptual point of view, CSP frameworks increase the privacy of user data. In pure client-side personalisation, which denies remote data access unequivocally, it can prevent numerous types of privacy breaches [2], as well as facilitating more effective personalisation across domains. However, there are several issues with a purely client-side CSP approach; restrictive client processing power, IP concerns, limited inference data, infeasible data overheads and the inability to easily sync profiles across devices. Aside from these technical concerns, an issue with user perception has also been observed, though incorporating additional security measures has been shown to effectively combat this. In response to these issues, several hybrid distributed approaches to CSP have emerged. These diverge from the pure client-side CSP ideal, by incorporating some remote activity and mitigating this risk with extra security measures.

[2] www.goodreads.com

[3] www.dropbox.com

2.1.4 Distributed CSP

As concerns with pure client-side approaches have grown, a second branch of CSP has emerged in which third party software is used to create a distributed approach [12]. The user's data is still be stored on the client device, however, the data analysis and personalisation may occur remotely.

Keeping the user data on the client device means that advantages of CSP such as enabling cross-site personalisation, providing richer data and alleviating the cold-start problem can still be seen in distributed approaches. However, employing external services to provide data inference and personalisation greatly increases the privacy concerns compared to a purely client-side approach. At the same time, these distributed approaches tackle many of the disadvantages of a pure CSP approach, like processing power, proprietary concerns, limited inference and the data burden, providing a more realistic solution for both users and content providers. The following sections will outline four state of the art distributed CSP systems and compare these across a number of facets.

2.1.4.1 State of the Art Distributed CSPs

To date, only research prototypes of distributed CSPs have been developed [2]. Of these, the four most highly cited [4] in the literature have been selected for comparison. These are as follows:

1. **Kalpana** [11]: Ankolekar et al. use Semantic Web technologies to realise a vision of client-side personalisation. Here the focus is on delivering cross-site personalisation through a client-side aggregator, managing and providing a single point of access to the user's data.

[4] In top conferences such as, Hypertext, UMAP, ACSAC and the IEEE Symposium on Security and Privacy

2. **PersonisJ** [13]: PersonisJ specifically focuses on mobile client-side personalisation with the personalisation performed on the client. The paper is the earliest found in this literature review to discuss such a framework as it came at a time when the increasing computational power of mobile phones made the framework feasible. The paper examines a CSP on an android phone addressing the concerns that arise when CSP is deployed in a mobile environment.

3. **RePriv** [52]: The motivation behind the RePriv system is to enable the browser to infer user data in a CSP, to form a user interest profile. The system also offers the ability for content providers to provide their own data mining code, extending these user profiles to their own needs. RePriv does not not attempt to completely hide all user data from the personalisation service. They posit that content providers would not participate in such a a scheme as they would lose access to valuable data which improve products and increase efficiency.

4. **MoRePriv** [62]: With MoRePriv, Davidson et al. proposes that client-side personalisation be provided by a unified system instead of by individual apps. The paper describes MoreRePriv, a service which provides operating system support to personalisation, specifically studying mobile personalisation.

This research is concerned with the architecture of these systems, the privacy offered by them and issues arising from their design. Thus, the architectures of the systems are examined aiming to establish where personalisation and data inference are performed, as well as how these process are coordinated. The privacy offered by these systems is examined to determine how each copes with the decreased privacy when employing external services. Finally, the latency of each is discussed as this proves an issue in each system. These features are summarised for each systems in Table 2.1 and further expanded upon in the follwoing sections.

	Distributed CSP System			
	Kalpana [11]	PersonisJ [13]	RePriv [52]	MoRePriv [62]
Data Coordination	Browser	OS	Browser	OS
Cross-Site Personalisation	✓	✓	✓	✓
Personalisation Location	Client Browser	Client	Remote	Client
Data Inference Location	N/A	Client & Remote	Client & Remote	Client & Remote
Open User Model	✓	✓	✓	✓
Additional Privacy Measures	Access Policies, Mutual authentication	Access Policies	Encryption	Pseudonymity
Evaluation Criteria	N/A	Performance	Performance, Classifier Quality	Performance
Evaluation Metric(s)	N/A	Response Time	Response Time, Memory Footprint	Runtime, Network Utilisation, Battery Impact

Table 2.1: Comparison of Distributed CSP Systems.

2.1.4.2 Cross-site Personalisation and Data Coordination

In each of these systems the user data is stored and managed locally on the client. The data collected from each website, visited by the user, is stored by the client in a single, unified storage location. This allows their data to be understood by heterogeneous websites, enabling a seamless cross-site personalisation experience for users across websites. In the case of both PersonisJ and MoRePriv, data access is mediated by the operating system, whereas both Kalpana and RePriv use browser based data coordination. PersonisJ also has a broadcasting functionality, which allows client applications to listen in and react if a component of the user data has received new evidence from a separate application.

2.1.4.3 Data Inference Location

Introducing remote personalisation brings additional privacy risks as the user data is now leaving the client. The remote provider may suffer a data breach, the data transmission may be intercepted, the provider itself may misuse or pass the data on to unauthorised parties etc. However, as discussed in section 2.1.4.5 a number of techniques are used by these state of the art systems to mitigate these risks. For example, incorporating access control mechanisms and using algorithmic privacy techniques to provide anonymisation or pseduonymity as well as security techniques like encryption and authentication to protect the data. Further techniques to mitigate risk place controls around the 3rd party software, for example, employing microservices so that user data is distributed and coarse-grained or employing only trusted software verified through auditing.

Kalpana uses data from several sources, similar to a web mashup, but does not perform any inference on that data, instead simply presenting this raw data together in a single interface. With PersonisJ, some basic inference is performed on the client, in the form of 'Resolvers', before the data is provided to any applications. However, after meeting certain security measures, the data is then passed to the client application which is free to perform either remote or client-based inference. Both RePriv and MoRePriv allow for custom classifiers or *'miners'* to be utilised to expand the value of the data inference for specific providers. These allow 3rd party providers to register code that would perform additional data extraction to suit their needs. RePriv incorporates these as browser extensions and, to prevent privacy leaks, statistically verifies them at the time of submission. The user is then responsible for granting these classifiers specific access to elements of their user profile, enabling the provider to perform either client-based or remote personalisation.

2.1.4.4 Personalisation Location

Although in all of these systems, the user's data remains on the client, the client does not always have the ability to perform personalisation itself. Instead it may simply execute code supplied by a personalisation provider, using the client as a platform for delivering remote personalisation. In both PersonisJ and MoreRePriv, a personalisation service residing on the client is used. Enabling this client-side computing minimises the need to share user data with the server. However, recognising the latency limitations of client-based personalisation (discussed further in section 2.1.4.7), PersonisJ also suggests further work investigating a security framework which enables the user to control whether any communication can occur outside of the client for a specific application. In Kalpana, the website provides the personalisation 'script', while personalisation itself is performed as an aggregation that takes place on the client, specifically within the browser.

RePriv performs the personalisation remotely, with services accessing user data through access policies. The personalisation is also augmented through a *'Web Service Relay'*, enabling RePriv to act as an intermediary between the user's personal data and service APIs. For example, when a user navigates to a movie site, the site can query RePriv which will in turn consult the Netflix API to return useful derived information to the movie site for personalisation.

2.1.4.5 Additional Privacy Measures

These distributed CSP approaches often allow the transmission of user information to remote services. Consequently, additional measures must be in place to maintain privacy and regulate website access to user data.

Access control mechanisms in PersonisJ, RePriv and MoRePriv prompt the user to grant access to specific data. This enables the user to accept or decline requests either programmatically or via a high-level policy; aiming to minimise user involvement wherever possible. Kalpana posits the possibility of maintaining a white-list of trusted websites

and using access policies that determine which content can be transferred and which must remain private. Users across all of these systems can edit and revoke permissions through open user models. Ideally, this capability should be integrated into a secure browser platform.

Even with those services performing client-based personalisation, the specific mechanisms mean that it is still possible for a determined service provider to perform user tracking. For example, with browser-based personalisation as in Kalpana, there is a potential security concern over data access. The service provider may try to gain unauthorised access to private information from the browser. Using Javascript in their website allows code to be written that monitors the webpage, sending and routing the results of any query back to the provider.

Each of these systems offers another layer of protection to combat potential leaks and to compensate for those users who do not take initiative to protect their privacy through open user models. PersonisJ utilises security permissions to regulate access to the database of user information. Client applications are required to explicitly declare how they wish to interact with this database, and permissions are granted only for specific actions on specific data. Along with this, it proposes an access mechanism which mediates interactions and transmits only necessary data; for example, allowing a playlist application access to a person's favourite genre, but not their full catalogue of favourite songs. In RePriv, the user database is encrypted to prevent tampering or spying by other applications. With MoRePriv, sensitive information is distilled into a restricted, deliberately coarse-grained persona. These personae are used to group users and provide a degree of pseudonymity by abstracting away fine-grained personally identifiable information. The personae provide a way to declassify sensitive information, limiting the potential damage from information leaks. Although 3rd party apps receiving this personae info could leak it, the paper concludes that due to the pseudonymity, the consequences of this would be relatively benign. The paper discusses the potential for further privacy protection by incorporating privacy enhancing technologies such as those discussed in section 2.1.2.1. The service is also complimentary to recent advances in data leak detection. Kalpana

[11], incorporates a client aggregator which sits on the client-side and provides access to user data through a SPARQL[5] endpoint. It mediates access to the data according to the access policies, ensuring that the website only has access to the results of the personalisation. Mutual authentication schemes also exist to ensure that the two parties engaging in a transaction are who they claim to be.

2.1.4.6 Evaluation

Kalpana presents a proof of concept system and discussion, with no formal evaluation of the system. As the PersonisJ system was concerned with the feasibility of CSP on mobile clients, it is evaluated on performance and scalability. The first evaluation performed was concerned with benchmarking performance against simple operations. Secondly, the performance was examined when the full framework was in use. The metric used in this study was the *average response time* of various methods over 200 calls. MoRePriv uses a number of case studies and crowd-sourced user studies to perform its evaluation. In the MoRePriv system, the focus is on evaluating a specific component of the system; the classifier, which determines whether the user is presenting genuine data for personalisation. This classifier is evaluated on runtime, network utilisation and battery impact. In the RePriv paper, evaluations are performed to assess both the quality of the classifier used for behaviour mining and to assess the performance overhead of the two data inference methods; default and extension based. In evaluating the inference overhead, latency based metrics were used, recording the webpage response times and the memory footprint. The evaluation of the classifier was complicated by the difficulty of selecting an objective metric, ultimately the rate at which a user's interest profile converged was used. As a secondary evaluation, RePriv also presented two case studies, evaluating each on large quantities of real data. The case study based methodology, seen with RePriv and MoRePriv, is an evaluation approach used to aid understanding of a

[5]https://www.w3.org/TR/rdf-sparql-query/

particular problem in great depth. Specific information-rich cases are selected to fully explore a research question. This methodology will be followed in evaluating the ICCP, aiming to use these case studies to build a representative sample of web traffic. This is further explored in section 4.2.

2.1.4.7 Latency

A number of limitations to the Kalpana system are highlighted, specifically in regards to latency. As the personalisation is browser-based the latency of the webpage request is increased, caching is suggested as an aid in this. With RePriv, end-user latency of the system is considered as a potential issue, caused by three things: the potential effects of default in-browser behaviour mining; the effect that each proposed personalisation extension has on document loading latency; and the performance of primary extension functionality. The RePriv system uses a Naive Bayes classifier due to its low computation cost on most problem instances. While latency was evaluated in real-life browsing sessions with good results, this was a limited study. The classifier chosen does not reflect current trends in data inference and the computations performed were limited to very simple examples. To ensure that the cost of data inference does not affect browsing activities, these computations are performed by a background worker thread. There are many opportunities for the thread to use resources due to the interactive characteristics of Internet browsing, i.e. periods of 'bursty' activity followed by downtime for content consumption. With MoRePriv the performance of the classifier is parameterised by the number of feature words in the classifier. Again, a bayesian classifier is used which is trained offline on manually curated data. In this study, building each classifier can be a difficult task however applying the classification is very fast. PersonisJ specifically proposes light-weight user modelling to enable use on a mobile client and is evaluated on performance and scalability. The study concludes that the time performance was adequate for simple operations, but highlights that more complex tasks and reasoning should be performed as background tasks.

2.1.4.8 Trusted Software

One branch of research not examined in these systems, looks at the use of trusted software [32]. This is software that can make guarantees about data storage policies, linkability, and disclosure. It is proposed that these systems would undergo technical audits and obtain certification by a trusted third party in order to be incorporated into privacy conscious architectures.

2.1.5 Gap in the State of the Art

Studies show that people are increasingly using their mobile phones for Internet access, even when at home or with a computer nearby [63]. While consumer phones are becoming increasingly powerful, modern personalisation techniques are becoming heavyweight at a faster rate. Recently, the fields of Artificial Intelligence (AI) and Machine Learning (ML) have seen a lot of success in optimising personalisation, creating more and more accurate prediction models [64]. In particular, the recent advancements in neural networks have triggered a great deal of research in deep learning-based personalisation [65]. Due to the capability of Deep Learning in solving many complex tasks while providing start-of-the-art results, many companies, including Netflix[6] and Google[7] [63, 66], have employed deep learning models to further enhance the quality of their recommendations. Recent advances have gained significant attention by overcoming obstacles of conventional models and achieving high recommendation quality [65]. However, deep learning is known to be data-hungry and computationally expensive [65]. The increasing data volumes in the big data era pose challenges to real-world applications. Scalability and the time complexity of models are critical to their usefulness. As a result, research into producing small/compact models [56], as well as into compression techniques [57], is becoming prevalent.

[6] www.netflix.com

[7] www.google.com

For CSP based approaches to privacy, the recent trends in these two fields is problematic. Mobile phones are increasingly being used as client devices over more powerful laptop devices, while more and more computationally expensive data inference and personalisation is being performed. Each system studied reported latency as an issue, yet none of the performance evaluations undertaken reflected these recent trends in data inference. It is clear that for a privacy conscious framework to be utilised in real-world applications it will need to match the performance of current client-server approaches under these conditions. Zhang et al. points out three areas of required future research in applying deep learning to personalisation:

- Incremental learning for non-stationary and streaming data such as large volumes of incoming users and items.
- Computation efficiency for high-dimensional tensors and multimedia data sources.
- Balancing of the model complexity and scalability with the exponential growth of parameters.

As discussed later in this thesis (section 3.6.3), incremental learning is essential to maintain privacy in a distributed CSP. It is clear that performing this on the client would involve infeasibly high computational costs. There is a gap in current state of the art to explore the performance of distributed CSPs in current, machine learning environments.

2.1.6 Decentralized Personal Data

Another architectural approach to protection of user privacy involves decentralised personal data management [67, 68]. This means that the user data is no longer stored on the client but instead in a secondary location which is still owned and managed by the user.

For example, the Hub of All Things [69], confers full legal rights of personal data to individuals through their ownership of a personal data server.

While these approaches tackle the performance issues faced by client-based storage approaches, the level of privacy offered is reduced. Narayanan et. al. [67] states "For any hope at absolute control, users must at a minimum, host data on their own device resident on their physical property.".

2.2 Intelligent Web Caching

In order to improve the latency issues experienced by distributed CSP approaches, this research studies the incorporation of web prefetching into these systems. Incorporating prefetching makes distributed CSPs a viable privacy-conscious alternative to a client-server approach. The following section will survey the state of the art in web caching and prefetching to inform the design decisions for this work.

2.2.1 Web Caching

Web caching plays an important role in improving web performance. Resources that are likely to be visited in the near future are stored closer to the client. This strategy reduces bandwidth usage and server load as well as the perceived lag for the user. Resources can be stored on the content server, in a proxy or on the user's own device i.e. Client-Side Caching [70].

The main issues tackled in optimising web caching strategies are deciding **which resources should be cached** as well as **when these resources should be replaced** [71] The following sections will explore the state of the art used in designing these cache replacement policies and determining cache resource selection. Research in the domain aims to find solutions which make the best use of available cache space, improve hit rates, reduce network traffic, and alleviate loads on the original server [72, 73, 74].

2.2.1.1 Cache Replacement Policies

The design of an efficient cache replacement algorithm is required to achieve a high performance caching strategy. The policy should utilise the available cache space to balance the number of requests to the server with the hit ratio. Traditional cache replacement policies have shown high performance in areas such as CPU caching and virtual memory systems, but they translate poorly to use on the web [71]. For example, the most common approach used is the Least-Recently-Used (LRU) algorithm which removes the least recently accessed objects to make space for new items. This is easy to implement and works well for uniform sized objects, like a memory cache. However, in web caching LRU performs poorly as it does not consider the size or the download latency of individual objects. Another algorithm, Least-Frequently-Used (LFU) replaces the object with the least number of accesses. This however can pollute the cache with objects that were once frequently accessed but have not been used in a long time. Cache replacement in the context of web caching, can prove more challenging than these classical systems due to both the varying object sizes and the varying cost of a cache miss.

There are five considerations made when designing a cache replacement policy for the web:

- Recency: The time since last access of the object
- Frequency: The number of times the object has been requested
- Size: The size of the web object
- Cost: The cost of fetching the object
- Access: The access latency of the object

Extensive research has been performed in the area of web cache replacement policies. In general, algorithms are divided into Recency-based, Frequency-based, Size-based, Function-based, and Randomised policies [75]. However, it is difficult to have an omnipotent policy that performs well in all environments, as each policy has a different design rational to optimise different resources.

More recent advances have exploited web access log files to build intelligent cache replacement policies utilising machine learning. These schemes produce more efficient and adaptive policies to deal with the constantly evolving web environment. With these approaches, the time to train and the extra computational overhead must also be considered.

2.2.1.2 Cache Resource Selection

Traditionally, caching strategies simply chose frequently used or recently used resources to cache. For example, after a user visited a webpage, that webpage might be cached so that should the user revisit the webpage in a certain time frame, it could be quickly reloaded from the cache instead of requesting a complete new copy from the server.

However, even with a cache of infinite size, it has been shown that the hit ratio on the web, i.e. the number of requested resources that are cached, lies between 40-50% regardless of the caching scheme employed [76, 77, 78]. This is due to the fact that most users frequently request webpages they have not yet visited. To address this and improve the hit ratio, content providers are attempting to predict in advance what a user might be interested in visiting through web prefetching.

2.2.2 Web Prefetching

Prefetching is used to cache resources a user may never have interacted with by predicting the resources they might require in the future. Many studies have shown that the combination of caching and prefetching doubles the performance compared to only caching [79, 15, 80, 81]. According to [79, 15], a combination of web caching and prefetching can potentially improve latency up to 60%, whereas web caching alone improves the latency up to 26%.

However, if a prefetching scheme is deployed, and the user ends up requesting very few of these prefetched resources, the scheme can actually slow down performance [15, 80, 81]. The unused resources increase network traffic as well as the load on the content provider. Thus, a prefetching approach must be carefully designed to ensure a net benefit effect.

2.2.2.1 Prefetching Strategies

Prefetching strategies can be characterised by both the data they use to make predictions, and the algorithmic approaches to making these predictions.

In the literature, the data used to make predictions is generally separated into two types; ***content-based*** and ***behaviour-based***. Content-based prefetching analyses the layout and content on a webpage to predict the links the user might click [82]. Whereas behaviour-based prefetching observes the user's previous behaviour.

The most common approaches seen in the literature to perform predictions over this data include dependency graphs, Markov models, and data mining [83].

Dependency Graphs

One approach for prediction constructs a dependency graph to predict the next webpage. Each node in the graph represents a webpage and the edges indicate when one webpage was accessed after another webpage. These edges are assigned a weight to indicate the probability of transitioning from one page to another. The webpages are then prefetched based on their probability going over a certain threshold.

While dependency graph approaches have been shown to reduce latency time, they also increase network traffic. As the dependencies between only two webpages are examined the prediction accuracies also tend to be very low.

Markov Models

Markov model approaches predict the next page by examining the user's entire access sequence. This is matched against the user's historical web access sequences. The predictive accuracy of such techniques increases as longer sequences are utilised, however this is at the expense of lower coverage. Another issue is in storage of long historical sequences, algorithms used in cache replacement can also be applied here for example to store only the most frequently seen sequences.

These approaches tend to be poor at adapting to new patterns as they rely heavily on the historical behaviour of the user. This impacts the efficiency of these algorithms. Another issue with Markov model approaches is that they are not designed to work on client machines. As client devices request webpages from the entire Internet, the differences in request frequency for every webpage is not significant enough to build an effective management strategy for historical sequences.

Data Mining

Data mining approaches to prefetching can be further divided into association rules-based approaches, and clustering-based approaches.

Association rules based approaches to prefetching aim to discover groups of webpages which are commonly accessed together in the same user session. The main problem with this method is that too many useless rules are produced. This causes inconsistent predictions, especially when the dataset is large.

Clustering based approaches are used to find similar groups in data. Clustering is heavily used to improve the efficiency and scalability of real-time personalisation tasks. Clustering of data can be achieved in a model based, or a distance based fashion. Model based approaches often specify the model type a priori. Clusters can be formed by grouping either similar webpages or similar user sessions.

2.2.2.2 Prefetching Evaluation Metrics

Domenech [14] classifies metrics relating to prefetching into three main categories according to the part of the system they wish to evaluate: prediction, resource usage, and latency. The research also makes three recommendations for evaluating prefetching in a system:

- Performance studies should at least include latency related metrics. Depending on the goal and context of the performed study, **latency per webpage** or **latency per object** is preferred. For instance, if the goal is to analyse the user's point of view, the latency per webpage must be included

- Latencies cannot be used as the only metrics to check performance. Performance comparisons should be made by means of a useful cost-benefit analysis. Studies must analyse how the latency reduction has been achieved for a given proposal. In this sense, resource usage indexes should be taken into account. **Traffic increase** is the one that provides most information; therefore, performance studies should include at least this index.

- It is not advisable to perform studies about the behaviour of prefetching techniques focusing only on the algorithm. Nevertheless, to evaluate this part, studies should include **recall** as a performance metric because it is the most correlated to the latency per object and latency per webpage respectively.

The highlighted metrics are defined below:

Latency per page ratio The ratio of the latency that prefetching achieves to the latency with no prefetching. The latency per webpage is calculated by comparing the time between the browser initiation of an HTML webpage GET and the browser reception of the last byte of the last embedded image or object for that webpage. This metric is important as it represents the benefit perceived by the user.

Traffic Increase The bytes transferred through the network when prefetching is employed divided by the bytes transferred in the non-prefetching case. This metric allows the overhead of prefetching to be analysed from the service provider's perspective.

Recall i.e. Hit Ratio The ratio of prefetch hits to the total number of objects requested by users.

$$R_c = \frac{\#PrefetchHits}{\#Requests} \tag{2.1}$$

The hit ratio alone, indicates how many requests hit and does not give any indication of the bandwidth or latency involved in these requests. Thus the previous metrics are required to add context to the evaluation.

Precision The ratio of prefetch hits to the total number of objects prefetched.

$$P_c = \frac{\#PrefetchHits}{\#Prefetches} \tag{2.2}$$

These metrics will be utilised in the evaluation of this thesis.

2.2.3 Conclusions

Examining the state of the art in web caching and prefetching frames the architectural approach for this research.

The prefetching predictions made as part of the experiments detailed in this thesis, will be preformed on behaviour-based data, as outlined in the next section. This behaviour-based data provides a large data overhead, enabling more rigorous framework performance testing than could be achieved using low overhead content based personalisation data. The most common approaches used in web prefetching are Markov Models and Data Mining. However, the Markov model based approaches are not suitable for client-side prefetching and do not adapt well to new patterns in the data. In recent years,

data-mining approaches have dominated the research efforts as these have shown to be the most effective [84, 80, 81]. Thus, this research will employ machine learning approaches to predict user behaviour, best reflecting the state of the art and addressing the challenges in such a system.

The next section will focus on the data used for the behaviour-based, propensity prediction, examining the state of the art in interaction based prediction.

2.3 Interaction-based Prediction

In this research, the prefetching strategy attempts to predict what page will be requested next by examining the propensity or tendency of different users to interact with different elements of the webpage. This is termed ***'Propensity Prediction'*** and utilises user interaction behaviour to inform the predictions made.

This section looks at the prediction of a user's navigational patterns based on this interaction behaviour. Following on from section 2.2.2.1 discussing data mining, the section outlines the specific data used during prediction for this research.

As this research is concerned with evaluating framework performance, data with a heavy load was chosen for prediction. This aims to test the framework under a 'worst-case' scenario. Thus, behaviour-based prediction is performed, specifically using interaction data like mouse movements and click patterns. Unlike data such as the user's interests or browsing history, interaction data provides a near constant stream of information to a prediction system. As the user browses a website they produce a wealth of data by clicking, scrolling, typing and moving their mouse.

Interaction-based predictions are heavily utilised in both search optimisation and advertising, aiming to predict which result or ad a user will click on. In some cases, predictions based solely on interaction behaviour have been shown to give better results than profile based prediction [85] i.e. based upon the user's preferences. However, the vast majority of studies use click history and its derived features as the only implicit interaction feature, with only a few studies examining the effect of other behaviours such as mouse movement. The following sections outline the state of the art in each.

2.3.1 Click Based Prediction

Predictions based on a user's click behaviour have been widely used in a variety of applications including, search engine optimisation [86], interface personalisation [87] and minimising latency through caching [88]. Due to the wealth of research in this field, well established *click models* exist [89], which reflect user click behaviour and allow for easy derivation of click based features. These reflect behaviour from numerous user studies accounting for user biases such as the position bias effect [90], novelty bias to previously unseen documents [91], and attention bias to visually salient elements [92].

Although click based prediction is widely used and provides accurate models the vast majority of studies utilise only click history. These studies ignore other, potentially useful, mouse dynamic information such as click speed and mouse movements. This research will examine the usefulness of such mouse dynamic data.

Studies often follow a *simulation based* approach in sourcing evaluation data. The existing click models can be utilised to build probability based user interaction data, removing the laborious task of collecting real data. Although models for user dwell time and click speed patterns could be built from the existing state of the art, there was not sufficient data to follow a simulation-based approach in this research. This is fully explored in section 4.3.

2.3.2 Mouse Dynamic Based Prediction

Studies examining interaction based features aside from click data have shown both mouse movement and scroll data to be valuable additions to a predictive model. These have been shown to more accurately infer searcher intent and interest than a click model alone [93].

Mouse movement data is most widely used in search engines and security, modelling user behaviour to inform search engine design or provide authentication. The most prevalent use of mouse movement data on search engine results webpages (SERPs), is in providing *relevance feedback*. Mouse movement data has been shown to improve estimations of user satisfaction as well as of document utility and document relevance [94]. Due to its close correlation with eyegaze [95, 96], mouse data has also been shown to be an effective proxy for a user's visual attention [97]. More generally, mouse movement data has also been used to evaluate usability on a standard webpage [98, 99]. In the context of Security, studies have found that mouse behaviour is an effective biometric characteristic for user authentication [100, 101]. Features like the mouse direction, acceleration, curvature, degree of overshoot, and click duration, have been used to positively identify individual users with high levels of accuracy. Some limited studies have also found promising results in analysing this biometric data to determinine a user's gender [102] and personality [103].

Although it has been successfully applied in other areas, no research could be found attempting to predict a user's navigational intent using mouse movement data.

2.3.3 Summary

Research into interaction based prediction is almost entirely focused on click history. For this research, data with a heavy load was desired to test framework performance. As other interaction based features like mouse dynamics, provide a heavy load and have been shown to perform well in predictions outside of user navigation, these will be used

in this research. As a minor contribution, this research aims to explore the relationship between interaction behaviour and user navigation profiles i.e. their pattern of navigation through a website. No previous research could be found aiming to predict a user's clicks on a webpage based only on mouse dynamics.

The state of the art investigation provided here, establishes many interaction features which have been successfully used to infer about the user. The ability of these features to predict a user's navigational intent will be tested in the machine learning model in Chapter 7.

2.4 Chapter Summary

This chapter presented the state of the art in Privacy Conscious Personalisation, Intelligent Caching and Interaction-based Prediction.

It first reviewed the existing research on privacy conscious personalisation, in particular exploring architectural solutions; identifying variations in approaches and the limitations of each. This review was performed to provide the theoretical foundations for the design of a distributed client-side personalisation framework in this thesis. The design considerations explored in this chapter are outlined in table 2.2. This forms the basis for the ICCP design outlined in the next chapter.

No existing studies address the major contribution of this thesis, designing and evaluating a framework capable of improving upon the performance of such frameworks through the use of predictive prefetching.

Finally, this chapter identified a minor contribution in attempting to predict a user's clicks based solely on mouse dynamics.

Design Consideration	SOA Approaches
Storage of User Data	Client, Decentralised 'hub'
Data Coordination	Browser, OS
Personalisation Location	Client, Remote
Data Inference Location	Client, Remote
Additional Privacy Measures	Open User Model, Access Policies, Mutual authentication, Encryption, Pseudonymity
Cache Replacement Policy	Recency-based, Frequency-based, Size-based, Function-based, Randomised
Prefetching Data	Content-based, Behaviour-based
Prefetching Algorithm	Dependency graphs, Markov models, Data mining
Interaction Data	Click based data, Mouse Dynamic data

Table 2.2: Design Approaches from the State of the Art

Chapter 3

Design & Implementation

This chapter aims to explore the core design requirements and challenges for an Intelligent Client-Centric Personalisation (ICCP) Framework, fulfilling the second research objective set out in the previous chapters of this thesis.

The research proposes that incorporating predictive cache prefetching with a distributed CSP architecture, could realise the privacy offered by a client-side personalisation approach, while enhancing the performance of existing techniques. The framework must balance predictive accuracy and performance to determine **which** webpages to prefetch and personalise, as well as **when** to do this.

State of the art research, from the previous chapter, guides the design and implementation of the ICCP Framework along with supporting services. Specifically, two services are designed and implemented, a Propensity Prediction Service (PPS) which performs interaction based prediction and a Personalisation Service.

3.1 Design Strategy

Although this thesis is organised sequentially, the design of the ICCP framework was performed iteratively. A Minimum Viable Product (MVP) strategy was followed for development. This mean that for each stage of the experiment i.e. The Design, Data Gathering, Implementation and Evaluation the most basic possible version was initially performed. An MVP approach develops all aspects of the process in parallel stressing the importance of learning and testing the design assumptions early. This allows insights to be garnered quickly and with the least amount of effort possible.

The initial proof of concept for the thesis was built to test the design assumption that prefetching would improve the page response time i.e. the latency. An ICCP framework which employed no personalisation or prediction was built, this simply prefetched the next URL based on the correct solution seen in the data. This system was evaluated with user data gathered from only 5 users. The latency reduction achieved with a simple prefetching mechanism was demonstrated against that with no prefetching and shown to be a notable improvement. Having proved the design assumption central to the research, this could then be iterated upon and built out to produce the statistically significant results presented in this thesis.

Iterating in this way meant that each stage could be improved upon based on the results of a later stage. For example, the initial proof of concept lead to a change in the way data was gathered from users. Having used this data to perform prediction evaluation, improvements in the data formatting were highlighted, additional data that would be useful to collect was underlined and requirements for the data quality were established.

3.2 Design Considerations from the State of the Art

The analysis conducted in the state of the art chapter influenced various aspects of the applied research detailed within this thesis. The aim of this section is to provide a summary of the design considerations arising from the state of the art for the ICCP framework developed in this research. To enable the privacy of a distributed CSP approach, as detailed in section 2.1.2, the following six design considerations must be examined:

- Storage of the user data
- A coordination strategy to manage data requests and responses
- The personalisation location
- The data inference location
- Additional privacy measures to protect user data
- Predictive Prefetching techniques aiming to minimise latency

The following sections provide a discussion of each design element contributing to the overall architecture of the ICCP framework. This can be compared with the design of the state of the art systems examined in section 2.1.4.

3.2.1 Data Storage

An essential principal of client-side personalisation (CSP) systems is that user data is stored only on the client. This forms the basis of the privacy offered by both CSP and distributed CSP approaches. The ICCP framework must be designed to follow this. This data includes both the raw user data (herein referred to as the 'user data') as well as the user model, which is the result of data inference performed over this user data.

3.2.2 Coordination

The management of user data must also be performed on the client in CSP systems. In the systems examined in the state of the art, this was either performed by the browser or by the operating system (OS). For this research, a browser based coordination strategy is utilised for its simplicity. This did not have any disadvantage over an OS-based strategy in the context of this thesis. Further work could extend this to an OS level which would enable more data to be available to the personalisation e.g. the user's calendar. Each of the systems examined in the state of the art chapter enabled cross-personalisation through the coordination strategy, this was due to the maintenance of a single user data store rather than one per website. As the benefits of this approach outweigh the negatives as discussed in the previous chapter, this research will also implement a single data storage location.

3.2.3 Personalisation and Data Inference Location

In order to implement client-side personalisation several processes must be carried out including data collection, data storage, data inference and finally personalisation based on this inference. As discussed in the state of the art chapter, having the client device carry out the data collection and data storage duties maintains a high level of privacy for the user as the data remains on the device. For this research to produce a minimal implementation of client-side personalisation along with predictive prefetching it was necessary to design two services performing data inference and personalisation respectively.

In the systems examined in the state of the art, personalisation was performed either on the client or by a remote service. Client-side personalisation places an extra processing burden on the client device as well as potentially exposing confidential personalisation algorithms. Many of the systems examined put forward problems with latency as an issue when performing personalisation on the client-side. An even greater problem is the latency issue with client-side data inference. Unlike personalisation, each of the

state of the art systems examined offers a method for remote inference due to its high computational cost. This issue will be experienced even more acutely with the rise in machine learning based inference techniques, which have a large computational burden, particularly when computing and updating the user model.

As the framework was built upon iteratively, different design choices and trade-offs could be investigated. Once the latency improvement from prefetching was demonstrated through the MVP, the effect of using remote personalisation and data inference services could then be tested. Although the network load increased, the latency overall was seen to decrease. Given the increased service load for a full scale system it was likely this effect would be more significant as the research progressed. Weighing the privacy-latency trade-off for this design decision the latency was seen to a be more significant concern. Thus, this research employed remote personalisation and data inference to minimise latency and provide a forward looking solution. However, as remote personalisation and data inference come with more privacy concerns, implementing additional security measures was essential as discussed in section 3.2.4.

3.2.3.1 Micro Services

As cloud computing has grown in recent years, new trends have emerged in software engineering architectural styles. Traditional systems used a monolitic approach i.e. an application with a single, large, codebase that offered tens or hundreds of services using different interfaces. Now, many large companies (e.g. Amazon, Netflix and eBay) are moving towards a microservice based approach to software design. This is an approach that promotes the use of small, finely grained services which can run independently of each other. These modular services are much faster to implement, easier to understand and maintain, and each service can be developed, deployed and scaled independently. In addition to this, the development costs are significantly reduced, as many entities can be reused from existing software enabled components.

Microservices break up personalisation to perform distinct tasks, for example one service might train a user preference model on the user's taste and another might then recompose suggested reading based on top preferences supplied. From a privacy perspective, the inherent distribution of knowledge in microservices is advantageous. In designing the ICCP, this microservices feature can be built upon to enhance privacy. The client can be used to control how information is shared, ensuring that no single service processes the same user's data.

As microservices are becoming more and more prevalent in the cloud computing space and they offer an additional layer of privacy for the user, a microservice based architecture was chosen for this research. Thus, both remote personalisation and data inference will be performed by microservices.

3.2.4 Privacy Design

The privacy principles behind the ICCP framework aim to provide maximum protection of both user and provider data, while delivering an efficient personalisation experience.

Privacy is provided by a distributed client-side personalisation approach through the client coordinator. These approaches do not attempt to completely hide all personal information from the party responsible for providing personalised content. Instead strict controls are used to govern how user data is shared with external services while maintaining storage on the client itself. There are many additional measures that can be implemented to further increase privacy, e.g. encryption, authentication or pseudonymity. Each of these has been discussed in the state of the art chapter.

In the systems surveyed in chapter 2, an open user model was implemented to enable user control over the sharing of user data. This leaves it to the user to decide which parties may access the various types of data stored and manages dissemination accordingly in a secure manner. However, given the increased management burden this places on the user, open user models are not always seen an advantageous. As discussed in section

2.1.2.3, users often eschew the hassle of changing privacy settings [48]. Thus, it is necessary in designing such a system to alleviate the management burden by utilising techniques discussed in the state of the art such as automation, generalised policies and visualisation. These lessen the overhead on the user and can minimise issues where users are simply annoyed by repeated prompts and authorise all accesses [52]. Although not explicitly implemented in this research, the design of the ICCP is undertaken with an open user model in mind.

For the ICCP framework, access management must be enforced by the client coordinator to ensure that the privacy requirements for the architecture are upheld. The client coordinator regulates access to user data through provision of an endpoint with appropriate access policies in place. The external microservices therefore receive only the minimal data required to achieve their tasks. The user is also provided with an interface to their data, to view the data being collected and set their own access policies; reducing the personalisation to a level with which they are comfortable. While these measures are important for increasing the privacy provided by an ICCP framework, the evaluation in this thesis was concerned with the performance of the ICCP. Thus, access polices and an open user model were not implemented in the system. However, there were additional privacy requirements that arose in the design and implementation of the microservices.

As information is being shared with remote microservices, there are additional requirements needed to maintain a privacy conscious architecture. While processing data, the microservices must not store any user data or profiles. Therefore, the following requirements arise for microservice in a distributed CSP:

- User data must be processed as a stream
- User profiles must be updated incrementally
- User data and profiles must be discarded after use

These data access restrictions impose limitations on the specific learning algorithms and methods that can be used by the microservices. These are explored further in section 3.6. Another consequence of utilising a microservice based architecture is the distribution of knowledge. No single service processes all of the user's data, each receiving only partial information.

The design of the ICCP described thus far, causes content providers to loose access to user data, which may prove valuable in improving products and increasing efficiency. However, a trusted microservice could be used to provide aggregate statistics to service providers in a privacy conscious manner. These would implement techniques such as differential privacy and homomorphic encryption as identified in the state of the art (Section 2.1.2.1). These services could also aggregate over multiple websites and provide additional insights that service providers could otherwise not access.

Another privacy consideration arises during the prefetching process, the webpages requested from the content server during prefetching could inadvertently reveal information about the user. For example, if the webpages prefetched are always focused on a single new topic, the interests of the user can be inferred by the content server. Approaches such as obfuscation through the use of dummy prefetches, could be used to hide the true intent of the user, though this would increase both the data and network burden of the framework. Strategies to combat this is an area for future research.

3.2.5 Predictive Prefetching

This research proposes that the addition of predictive prefetching to a distributed CSP could improve upon the limitations and performance issues in current client-side personalisation approaches. Augmenting these systems with the ability to predict which page the user will request next, allows this content to be prefetched and stored on the client device for faster retrieval. Thus, the design of the ICCP must also address:

- A cache prefetching strategy
- User modelling which informs cache prefetching

The design must therefore incorporate a cache in which both prefetched resources, and previously accessed resources can be stored. As this research is concerned with evaluating the effect of the prefetching strategy, the cache replacement policy is consistent throughout the experiments. A Chrome browser cache is used for evaluation. According to [104] most browsers utilise an LRU-FP [105] cache replacement policy. As discussed in the state of the art, 2.2.1.1, a Least Recently Used (LRU) approach removes the least recently accessed objects to make space for new items. As shown in the evaluation chapter, however, the cache replacement policy used by the browser does not impact the evaluation as no experiments fill the cache to maximum capacity at the moment. Therefore, there is no need to replace pages in the cache.

For this research, a prefetching strategy utilising machine learning techniques is employed to best reflect the state of the art in predictive prefetching. The impact of these large scale algorithms will be evaluated along with the limitations in deploying these in a client-side system. This is further discussed later in this chapter.

3.2.6 Summary

This section explored the design considerations arising from the state of the art. The ICCP design decisions made based on this research are summarised below:

- Storage of the user data and user model on the client-side only
- A browser based coordination strategy to manage data requests and responses
- Trusted microservices to perform:
  - Personalisation
  - User Modelling to inform the personalisation
- Predictive Prefetching techniques aiming to minimise latency

A summary of the ICCP design is given in Table 3.1, in the context of the state of the art systems examined in section 2.1.4.

This research is focused on the effect of the ICCP design on the user, choosing to focus on evaluating user latency i.e. webpage response time. However, other consequences of the design must also be considered, for example the resource usage and the data storage burden placed on the client device. Thus, an exploratory evaluation considering these factors is also performed and presented in section 6.5.4. Balancing the latency with the storage burden is a design consideration which can be made on a use-case basis when implementing an ICCP framework.

	Distributed CSP System				
	ICCP	Kalpana [11]	PersonisJ [13]	RePriv [52]	MoRePriv [62]
Data Coordination	**Browser**	Browser	OS	Browser	OS
Cross-Site Personalisation	✓	✓	✓	✓	✓
Personalisation Location	**Remote**	Client Browser	Client	Remote	Client
Data Inference Location	**Remote**	N/A	Client & Remote	Client & Remote	Client & Remote
Open User Model	✓	✓	✓	✓	✓
Additional Privacy Measures	**Any mentioned privacy measure, shown to be implementable**	Access Policies, Mutual authentication	Access Policies	Encryption	Pseudonymity
Evaluation Criteria	**Performance, specifically user latency**	N/A	Performance	Performance, Classifier Quality	Performance
Evaluation Metric(s)	**Response Time**	N/A	Response Time	Response Time, Memory Footprint	Runtime, Network Utilisation, Battery Impact

Table 3.1: ICCP in Comparison with Distributed CSP Systems

3.3 High Level Design

The aim of the Intelligent Client-Centric Personalisation framework is to enable the client to orchestrate microservices to deliver fast personalised content while maintaining user privacy.

The components involved in such a framework are a ***client-coordinator***, to orchestrate microservice interactions, and any relevant microservices, at a minimum involving a ***personalisation service*** and a ***user modelling service***. Storage of the user model, user data and the cache is performed on the client-side and is under the control of the client-coordinator. The website provider acts as a 'content server'. The ICCP framework, incorporating each of these components is shown in figure 3.1. The following section outlines the function of each component:

Client Coordinator

The client coordinator is embedded in the client's web browser. It is responsible for gathering, storing and managing both user information and web caches. The design and implementation are further detailed in section 3.5.

Personalisation Service

This is a service to perform content personalisation. The client coordinator manages interactions with the personalisation service, deciding when and how often to personalise webpages. These interactions are detailed and the implementation descibed in section 3.5.2.

User Modelling Service

In this research, the user modelling is performed by the PPS. This employs machine learning to predict the propensity of a user to perform an action. In this case the prediction is the next webpage that the user will request. The prediction made by the PPS is also used to inform the cache prefetching, its implementation is detailed fully in section 3.6. The interactions between the client coordinator and the PPS are examined in section 3.5.1.

User Model

The user model is a predictive model built and updated in the user modelling service (in this case the PPS). It is utilised to make future predictions about the user's behaviour, in this research predicting which webpage the user will request next. As the model must be stored on the client device and never by the user modelling service it is passed between the client and PPS during service requests. This is further detailed in section 3.6.2.

User Data

User data is the raw information collected about the user. For example, the interactions that they have made.

Cache

The client side cache contains both the regularly cached resources for previously viewed content as well as prefetched webpages and resources.

Content Server

The content server is a web server delivering the requested webpage and resources for the user. Its full implementation is described in section 3.4.

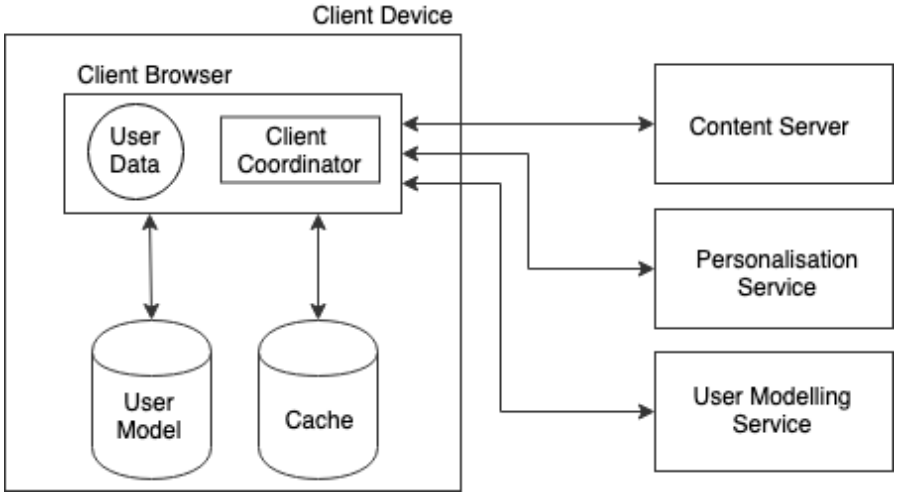


Figure 3.1: Intelligent Client-Centric Personalisation Architecture

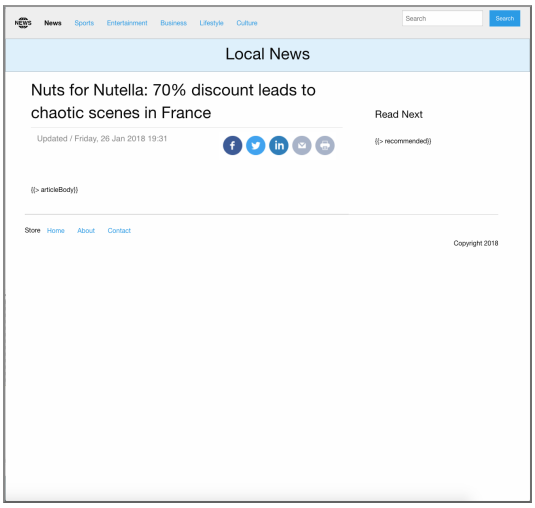
3.4 Content Server

In a traditional client-server architecture, the content server would perform the job of the client coordinator and its microservices; managing the user information and cache as well as performing user modelling and personalisation. In an ICCP framework, control is moved to the client to maintain user privacy by reducing information leakage. Thus, the job of the content server is simply to deliver the resources and tools necessary to perform personalisation to the client.

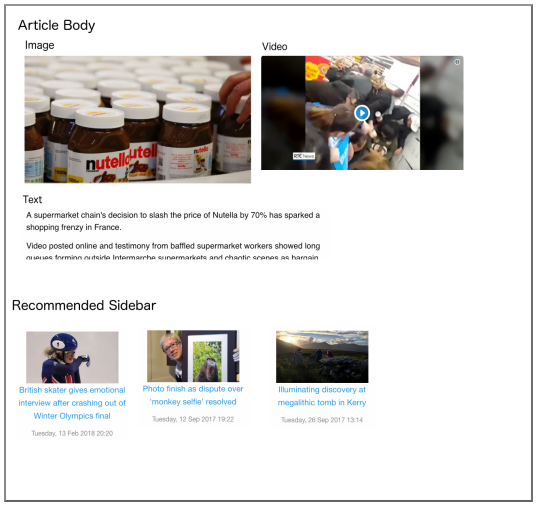
Each webpage is represented by the content server as a ***Template*** and a ***Context*** object.

The template gives the barebones outline of the webpage common to all users. Placeholders are used for sections which are personalised to each individual user. For example, a news website may have a banner bar and article headline that all user's see when accessing an article. However, the 'Read Next' sidebar which features suggestions of other interesting articles may be personalised based on the specific user's interests. In this case, the headline and banner would be included in the template but the sidebar would simply be represented by a '{sidebar}' placeholder. This is illustrated in Figure 3.2a.

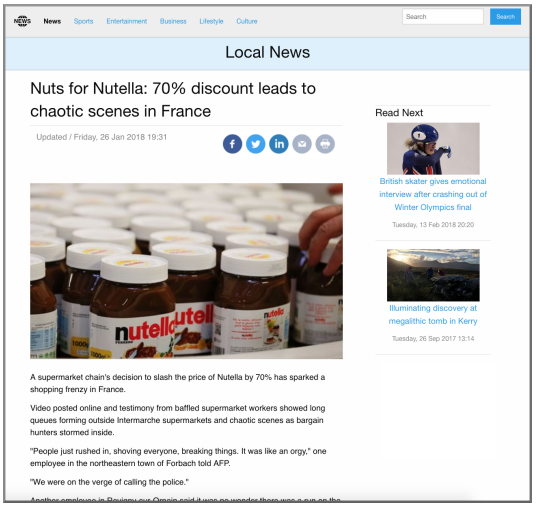
The context supplies the data that *may* be used to fill out placeholders. In the news website example, the context would include a selection of popular articles which can later be filtered on the user's interests. The personalisation service uses it's proprietary logic to compose the template and context, determining which data to use and how it should be arranged to best suit the user. It returns a fully fleshed out webpage to the client coordinator for rendering. For example, in Figure 3.2 the articles recommended for this user are selected from the available items in the context to compose a recommended *'read next'* sidebar. A sample webpage context and template are illustrated in Figure 3.2.



(a) Webpage Template



(b) Webpage Context



(c) Personalised Webpage

Figure 3.2: Sample Webpage Composition

As the context object must include a range of possible content to display, more data is being transferred than in a traditional state of the art approach. This adds to the trade-off between the privacy and the speed of the system.

Implementation The Content Server was implemented in NodeJS[1] and Express[2], allowing Javascript to be used on the server-side. Templating was achieved using *Handlebars*, a semantic templating library for JavaScript[3]. This uses a template and an input object to generate HTML. The content server composes the non-personalised elements of the webpage on the server-side and defers personalised content composition to the personalisation service.

For example, when composing the body of the article shown in Figure 3.2, the handlebars template shown in Listing 3.1 is used. This consists of HTML with some Handlebar 'Expressions' to be replaced during generation. When the template is executed by the content server, the {{articleTitle}} and {{lastUpdated}} expressions will be replaced with the input values but the {{> articleBody}} will be deferred for use by the personalisation service. This deferral was achieved through the addition of an extra 'deferral character' to personalised content, as illustrated a backslash is used as the deferral character. This causes handlebar to ignore the expression during initial execution on the content server. When the template is received by the personalisation service, deferral characters are removed in pre-processing. Handlebars is then used with the article body template (Listing 3.2) to generate the final HTML representation, filing in each page component that exists for this article.

[1] https://nodejs.org/en/

[2] https://expressjs.com/

[3] https://developer.mozilla.org/en-US/docs/Web/JavaScript

```
<!-- Article -->
<div class="article medium-8 columns">
  <div class="headline">
    <h2>{{articleTitle}}</h2>
  </div>
  <div class="metadata columns">
    <div class="medium-7 columns">
        <p class="subheader">Updated / {{lastUpdated}}</p>
    </div>
    <div class="medium-5 right columns">
        <img class="right" src="/img/socialIcons.png">
    </div>
  </div>
  <div id="columns">
    \{{> articleBody}}
  </div>
</div>
<!-- END Article -->
```

Listing 3.1: Article Template

```
{{#each pageComponents}}
   {{> (lookup . 'name') }}
 {{/each}}
```

Listing 3.2: Article Body Template

3.5 Client Coordinator

The client coordinator is responsible for gathering, storing and managing both user information and web caches. This ensures that rapid personalisation can be provided for the user while maintaining the privacy of their data. The following sections outline the main operations of the client coordinator; Privacy Enforcement, Cache Control, and Service Call Coordination.

Privacy Enforcement The client coordinator regulates access to user data through provision of an endpoint with appropriate access policies in place. External services therefore receive only the minimal data required to achieve their tasks. The user is also provided with an interface to their data to view the data being collected and set their own access policies; reducing the personalisation to a level with which they are comfortable. For this research, no privacy controls are implemented. This is discussed in depth in section 3.2.4.

Cache Control The client coordinator must also deal with management of web caches. It must know which resources to fetch and when, this is guided by the prefetching strategy employed. The web cache stores prefetched webpages in the form of the context and templates required to recompose them. After a personalisation microservice is utilised to achieve this recomposition, the personalised webpage is cached in full.

Service Call Coordination In order to provide personalisation for the user, the coordinator must be capable of enriching the data. This is achieved through interaction with trusted external microservices. For example, user data is tracked by the client coordinator as the user browses the web, this can then be transformed into an executable user model through interaction with a user modelling microservice. The client coordinator can then send the necessary data from the user model to a personalisation service to get a personalised webpage.

Ultimately the user's browser would implement these data management strategies, taking on the role of the client coordinator. In this research, the client coordinator is embedded in the browser to add this additional functionality.

3.5.1 Prefetching

The implementation of an ICCP framework requires a prefetching strategy be deployed to decrease the response times for the user. This requires a strategy which can effectively predict the webpages that a user will visit in a browsing session, and the order in which they will visit them.

The prediction of user behaviour is achieved through an external microservice, performing the necessary user modelling. As discussed earlier in this chapter, this research will utilise the user's interactions to predict their propensity to click on different elements on each webpage. The output from this ***Propensity Prediction Service (PPS)*** can then be used to inform the cache prefetching strategy. If a user is very likely to visit a webpage, it should be pre-fetched to speed up interaction.

The operation and design of the PPS is discussed further in section 3.6. The rest of this section will focus on the interaction between the client coordinator and the PPS. These interactions are controlled through the use of two thresholds, the activity threshold and the prefetch threshold.

Prefetching Activity Thresholds

The client coordinator needs to determine how often the user model should be updated. This is based on the amount the user has interacted on a webpage; updates to the user model should be made whenever it is likely the interactions have caused the user model to change significantly. Establishing significant change, and the quantity of interaction which elicits it, is domain dependent. Activity Thresholds indicate how much interaction must occur for the user model to be refreshed. This threshold varies not only per domain but also per event. For example, the user clicking is quite significant behaviour so the model may be refreshed after every click. On the other hand, a mouse movement is comparatively insignificant, thus the activity threshold for this may be 4 or 5 movements. As different events have different entropies, each event has a separate activity threshold. Whenever an activity threshold is reached, the PPS will be called to refresh the user model, producing updated predictions. Determining the optimum values for these is not in the scope of this thesis but would provide interesting future research.

Prefetch Threshold

Each time the prediction is updated, the client coordinator must also determine if any webpages should be prefetched based on the changes. If the predicted probability of a webpage being visited goes above the 'Prefetch Threshold', the webpage will be prefetched. Once prefetched, the contents of a webpage are stored in the client-side cache.

Selecting the prefetch threshold is a trade-off between the number of requested webpages that have been prefetched, and the speed of the system. A low threshold prefetches more webpages, increasing the number of webpages served from the prefetch cache, and reducing webpage load time. However, it also increases the load on each of the services and on the network, negatively impacting system performance. Optimising these thresholds is an objective of this thesis, examined and evaluated in section 6.5.4.

The size of each webpage being prefetched must also be considered. The load of a single page fetch can vary hugely based on the quantity and size of the page content. In particular, media like images and videos can immensely increase the size of the request object, therefore acting as the dominating force in increasing page latency. Instead of prefetching media content in full, smarter mechanisms can be deployed to produce a more efficient solution, for example fetching only thumbnails until the user shows some propensity to interact with the specific media or using incremental prefetching techniques [106, 107]. A stable content profile is used in this research to remove the effect of page size on latency, this is discussed in Chapter 4.

To enable this, the PPS must return the likelihood of each webpage being requested and not simply the most likely webpage to be requested next. Using this technique allows the client coordinator more control over the webpages that are prefetched. The prefetch threshold can be set so that multiple webpages are prefetched in one call. Or if the probability is not high enough, it may decide not to prefetch anything.

3.5.2 Personalisation

The ICCP framework interacts with an external personalisation service to tailor the webpage for the user. This customisation is based on the user model.

The user model represents not only information about the webpage the user will request next, but also the data which will determine the personalisation of the webpage. For this research, interaction behaviour is also utilised for the personalisation logic.

Thus, the context of the real-time webpage interactions must impact the result of the personalisation service. i.e. The user's interaction behaviour, like moving the mouse, must be one of the factors determining how the next webpage will be personalised.

To meet this requirement, the type of personalisation performed by the personalisation service is content layout. This involves rearranging the layout of the webpage to meet the user's preferences. For example, if a user always scrolls past the wall of text in an article to get to the video at the end, the video might be moved up to the top on future webpages.

As with prefetching, the client coordinator must know how often to update the user model, as well as when to perform personalisation. Again, two thresholds are used to control these interactions.

Personalisation Activity Thresholds

As with prefetching, the activity thresholds control how often the user model is updated. As personalisation is also performed based on interaction data, these thresholds are again tied to interaction events. While in this research, a single user model was used to influence both prefetching and personalisation, this would not be the case in most systems. In these cases, the separate user models would also require separate activity thresholds to determine when each should be updated.

Refresh Threshold

The refresh threshold controls how often the personalisation of a cached webpage is refreshed. This aims to determine when the personalisation of webpages is 'stale', and should be performed again. This allows personalisation to be updated based on the user's current behaviour after a webpage has been pre-fetched or cached. For example, suppose that a user is browsing a grocery website which features frequently purchased items along the side of the webpage; as they add items to the basket their current pattern reveals that they are likely planning a dinner party. Ideally, the personalisation would be reapplied and the sidebar would be updated to focus on dinner party items. Refreshing is valuable in situations where the user's recent interactions inform the prediction made. For example, when: classifying in-page actions, training a model for cold-start users or training rapidly changing user models.

3.5.3 Client Coordinator Interactions

To further explore the operation of the client coordinator, we examine the processes that occur as a user interacts with the ICCP framework. This is illustrated as an activity diagram in Figure 3.3, and outlined below:

When a user requests a webpage:

- The cache is checked to see if this webpage is already cached, and the cached webpage is still valid.
  - If it has been cached, the webpage is fetched and the personalisation checked to ensure it is up to date.
  - Otherwise, the webpage is requested from the content server which returns the webpage template and context.
- If necessary, the personalisation service is called. This uses it's proprietary logic to determine which data should be used to fill in the template as well as how it should be arranged. It returns a fully fleshed out webpage to the client.
- Finally, The personalised webpage is rendered by the client-coordinator.

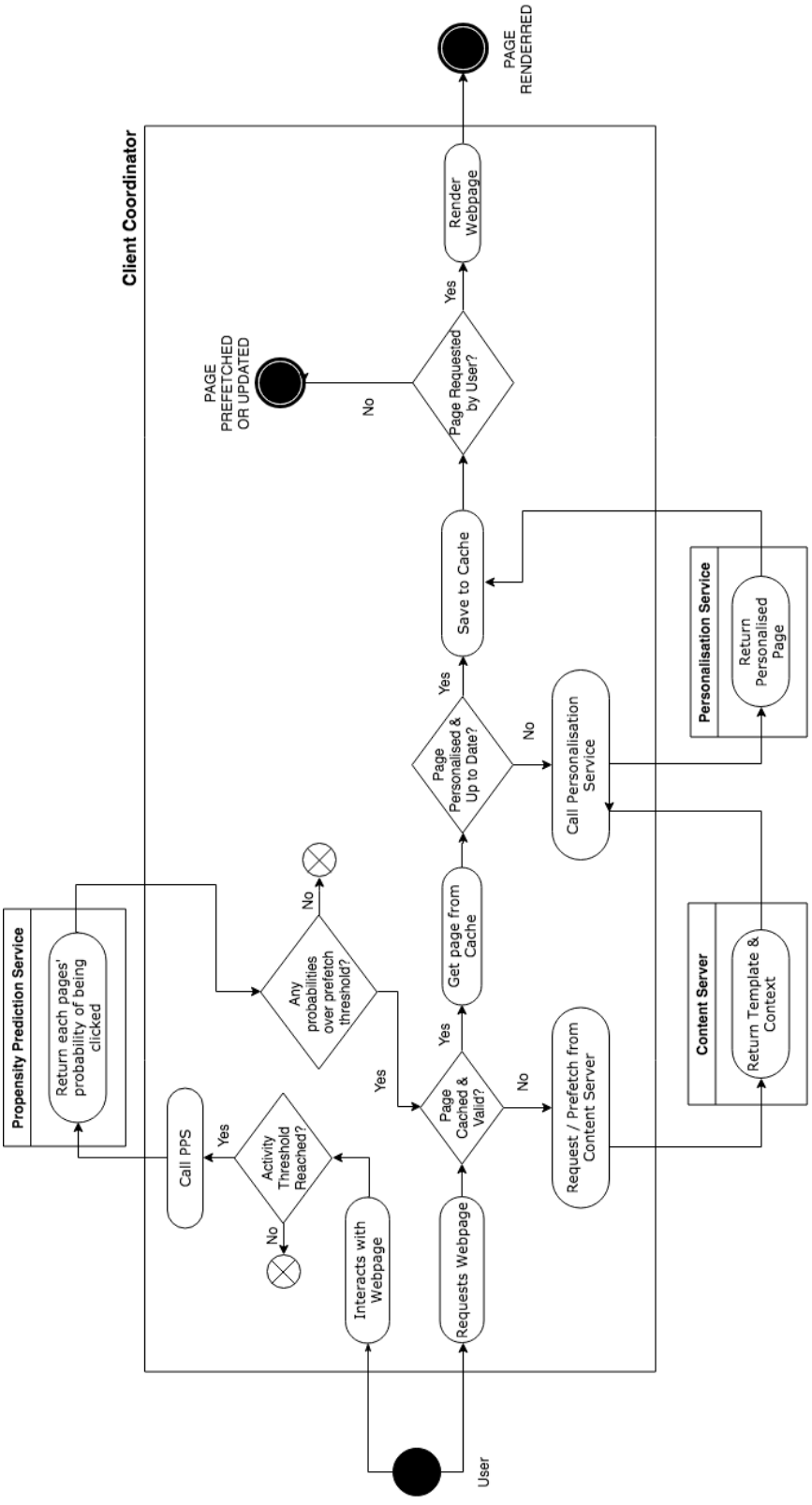


Figure 3.3: ICCP Activity Diagram

When a new user event occurs:

The messages passed between services during the page prefetching process described below is further highlighted in figure 3.4.

- The quantity of user events are compared to the activity thresholds to determine whether the PPS should be called.
- To call the PPS the new interaction data along with the user model (if it exists) is sent by the client coordinator.
- The PPS:
  - Calculates the prediction features based on the new data
  - Uses the user model to predict the probability of each link on the webpage being clicked
  - Retrains the user model if necessary
  - Returns the updated model and the prediction to the client coordinator

  Note: There is no storage of user data by the PPS.
- If the probability of any link being clicked has exceeded the prefetch threshold, the webpage is prefetched.
- To prefetch a webpage:
  - The content server is queried for the webpage template and context
  - Both the template and the context are separately cached on the client-side
  - The personalisation service is called to produce a personalised webpage
  - This personalised webpage is then cached on the client-side
- When the user requests a webpage that exists in the cache it can be served immediately

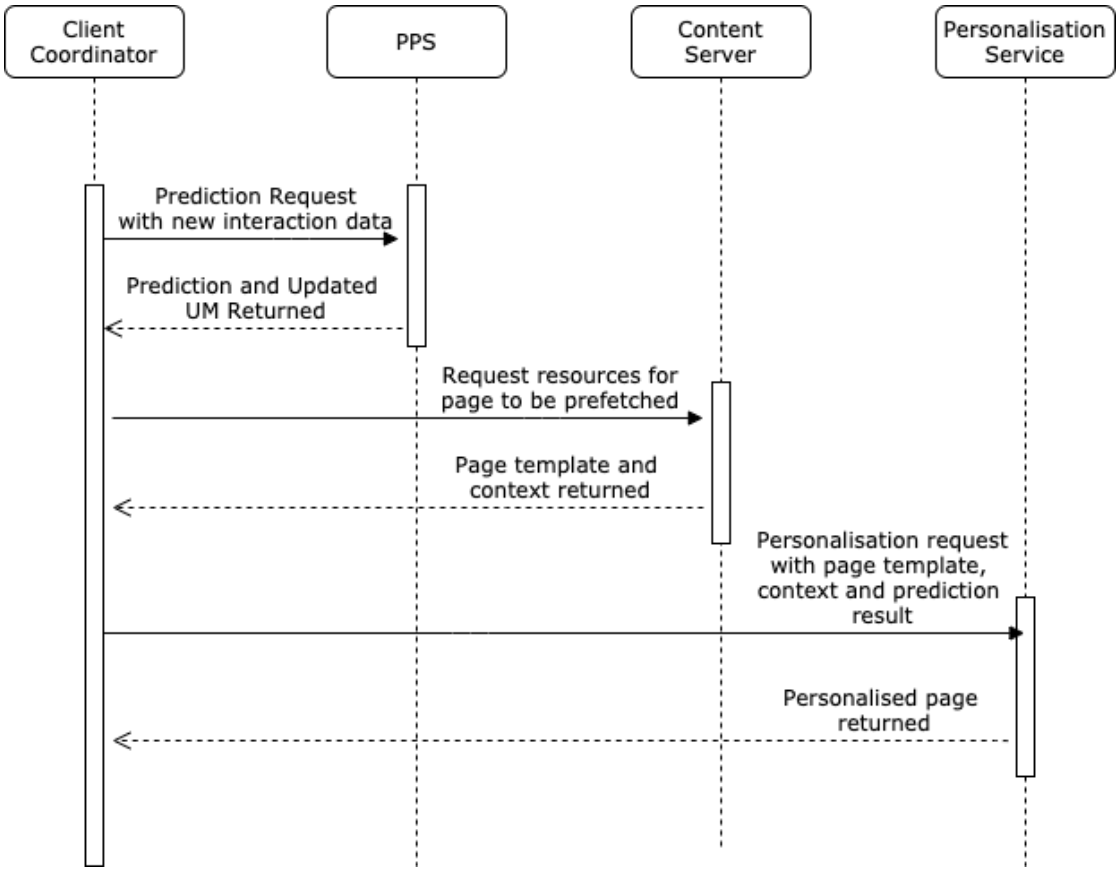


Figure 3.4: Message Sequence Diagram for Page Prefetching

3.5.4 Implementation

The Client Coordinator was implemented as a Javascript file delivered to the client by the content server. User data is held in a local webpage variable until a the activity threshold is reached. At that point, it is incorporated into the UM by calling the PPS. There are two challenges in the implementation of the coordinator as follows:

Caching

Browsers automatically cache resources for webpages that the user has previously visited. Cache rules like expiration are set on each resource by the content server. However, while prefetching is available in most modern browsers, they do not allow manipulation of the cache and these prefetched resources cannot be subsequently altered. Hence, personalising the webpage through interaction with an external service would not be possible before caching. This restricted cached resources to those fetched directly from the content server and not manipulated by the client coordinator. Thus, an alternative caching mechanism, ***locache***[4], was used allowing Client-Side Caching and giving the client-coordinator control over the cached resources. This caching framework performed client side caching in the browser using localStorage [5].

Prefetching

Prefetching requires a webpage to be requested from the content server and rendered on the client device without being served to the user. Ultimately, this should be achieved through the client browser, however, for this research an invisible iFrame in the webpage was used. The client coordinator requests a webpage to be prefetched from the content server, this is enhanced through interaction with the external services and then rendered by loading the webpage into the iFrame. When this webpage is then requested, it can be loaded straight from the iFrame without any further server interaction.

[4]https://github.com/d0ugal/locache

[5]https://developers.google.com/web/tools/chrome-devtools/storage/localstorage

In order to serve prefetched webpages, webpage clicks were interrupted and before requesting a new webpage the iFrame was checked. If the webpage existed this was served, otherwise the webpage request was sent to the content server as normal.

3.6 Propensity Prediction Service

The PPS performs the data inference, aiming to predict which webpage the user will request next. For each webpage the user is currently on, the navigation options are limited to what they can click on that webpage. Thus, each webpage has a limited number of possible outcomes for prediction. The prediction is based on the user's interaction data. Raw interaction events like mouse clicks, mouse movement, scrolling etc. are tracked. Features are then computed from this data to reason about the user's behaviour and predict their future requests. The full list of raw event data is provided in section 5.2 and the features computed from this are shown in section 7.5.

As the predicted URIs are discrete values, this is a **classification** problem. A supervised learning approach was taken, segmenting the users into train and test sets. As discussed in section 3.5.1, the PPS must return the likelihood of each webpage being requested. To achieve this, a ***probabilistic classifier*** is used, this predicts a probability distribution over the set of classes, rather than simply outputting the most likely class. The following sections outline the implementation of the PPS, detailing the user model and learning algorithm requirements, as well as discussing the API used to interface with the service.

3.6.1 Implementation

This section details the components involved in the design and implementation of a PPS in the context of the ICCP framework. These are illustrated in Figure 3.5 with the operation of each component described in the following paragraphs. The PPS was implemented as a Python Bottle Service[6].

[6]https://bottlepy.org/docs/dev/

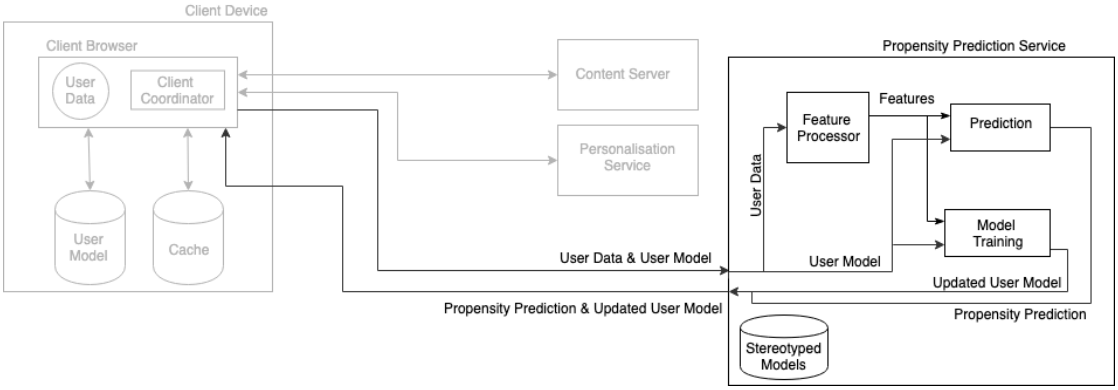


Figure 3.5: PPS Architecture

Feature Processor This takes the user data and calculates features for the learning algorithm. These can be monitored user events like click speed, or derived features which require some calculation based on the raw user data. Feature extraction is key to the success of a learning algorithm. In many modern approaches, like deep learning, feature extraction is automatic. However, for this research manual feature engineering was used. The results of the feature selection process for this research are presented in section 7.5.

Prediction / Learning Algorithm A prediction is made on the new user data by inputting the computed features into the user model. In a ICCP system, the choice of learning algorithm for the user model has several privacy-based constraints. Various algorithms which fulfil these requirements are evaluated and the best performing chosen. This was explored in section 3.6.3.

Model Training The user model is trained for a new user until sufficient data is obtained to produce good predictions. The model may also be intermittently trained to keep it up to date with the user's behaviour. Training the user model is implemented as a background task, this means that it does not effect the response time of the service. However, it does cause additional load on the PPS.

Stereotyped User Model The stereotyped user models are stored within the PPS, achieving model persistence through pickle[7], a powerful algorithm for serializing and de-serializing a Python object structure. These provide generalisations over user behaviour, providing an initial model for anonymous or new users.

3.6.2 User Model

A user model attempts to represent a user across various facets, for example, their preferences, goals, knowledge and demographics. The propensity model represents the behaviour of the user under varying conditions and is used to make predictions for the user's future behaviour.

3.6.2.1 Model Lifecycle

In this research, the user model is initiated by the PPS but is never stored by the service, protecting the privacy of the user's information. Instead, when the initial prediction request is fulfilled by the PPS, it returns a user model to the client. This model will be stored and controlled by the client coordinator. On further prediction requests to the PPS, the client will supply the user model as part of the request, also indicating whether this model should be incrementally updated using the current user data. During a prediction request, the PPS uses the model to predict the probability of each link on the webpage being clicked by the user, returning this prediction along with the updated user model.

3.6.2.2 Model Type

Models can either be **stereotyped or user-centric**, a stereotyped model is trained on many users' data so that it can then generalise about new users. A user-centric model, on the other hand, is trained only on a single user's data. User-centric models tend to achieve higher predictive accuracy, however, for new users, training data must be gathered, so predictions take some time to achieve these higher accuracies.

[7] https://docs.python.org/2/library/pickle.html

For this research, a hybrid approach is used. Stereotyped models are delivered to unknown users to make initial predictions with reasonable accuracy. These models are then retrained as user-centric data is gathered. This hybrid strategy enables both user modelling approaches to be explored in the implementation and evaluation and the relative difficulties of each to be examined.

For a stereotyped model to be deployed, user studies would be required, these would need the consent of participants for their data to be used in training a model. These stereotyped learning models would be stored within the PPS.

User-centric models introduce some issues, as these models are trained with private user data they cannot be stored in the PPS. Thus, online training and re-training must occur in the PPS, and the new model must be returned to the client-coordinator. Not all learning algorithms are suited to online training. The technical implications of implementing a user-centric model are further outlined in the next section.

3.6.3 Learning Algorithms for Privacy Conscious Frameworks

Recapping the privacy constraints for an external microservice in an ICCP, the PPS must:

- Process user data as a stream
- Update user models incrementally
- Discard both the user data and the user model

As user data cannot be stored, training of user-centric learning models needs to occur several times, whenever the data is available. Learning algorithms which can use input data to extend an existing model are called ***incremental learners***. Using these algorithms, training of the model can occur at different points in time without requiring access to the original training data. Algorithms that cannot incrementally learn, require the entire training set to be passed in at once.

The major disadvantage of this approach is that incremental learners can be slower to learn than batch methods. This may be an issue in situations where the user's recent interactions need to be incorporated into the model. For example, if it is necessary to quickly adapt to cold start users or in cases when a rapidly changing model exists. However, these use cases are niche. In this research, the use of an initial stereotyped model allows for good cold start predictions while the user-centric model is training.

Online Machine Learning uses incremental learners for cases where there is no memory, thus it must discard a sample after learning, and is not allowed to store it. Online learning is commonly used when it is computationally infeasible to train over the entire dataset, when it is necessary for the algorithm to dynamically adapt to new patterns in the data, or when the data itself is generated as a function of time, e.g., stock price prediction. For this research, the machine learning algorithm selected must be capable of online learning so that no user data needs to be stored.

The specific machine learning algorithm employed should be selected during the training phase. To find the optimal algorithm for a specific problem an empirical strategy is followed, testing a variety of techniques and observing which provides the most accurate results. During the PPS evaluation, predictions were made using each of the candidate models and cross-validation used to determine the most effective algorithm i.e. that which produces the most accurate prediction. The results of this process are shown in section 7.7.

The candidate ML algorithms for an ICCP must be both incremental learners and probabilistic classifiers. The supervised learning algorithms available in Scikit learn [8] were filtered on this criteria, leaving the following algorithms for testing during this research:

1. Multinomial NB
2. Bernoulli NB

[8] https://scikit-learn.org/stable/supervised$_l$*earning.html*

3. Perceptron
4. SGD Classifier
5. Passive Aggressive Classifier
6. MLP Classifier

Training Speed As incremental algorithms can often perform slower than batch methods, training speed must also be factored in to model selection. The most accurate model may be infeasibly slow for the specific ICCP use case. Thus, the data horizon must be considered in model selection i.e. How quickly does the most recent datapoint need to become part of the model? For an unknown user, it may be advantageous to have a fast learning model particularly if no stereotyped models are used. The exact quantification for acceptable training time will be use case dependent. One website may achieve high accuracy with a stereotyped model for new users and thus find it acceptable to take 5, 10, 20 minutes or more to train a user-centric model, achieving slightly higher predictive accuracy. Another site however may not have an initial model and perhaps require high accuracy so that they can immediately provide value to the user, in this case the site may want to train and return an updated user model in the time it takes to load the page. For this research, training time for all models was always under 1 second and completed by a background worker thread in line with the prediction completion. As a stereotyped model was used for accuracy evaluation, the accuracy of the user-centric model was not a significant gain over the initial model, this is explored further in Chapter 7.

Unseen Target Classes Most supervised machine learning models require all possible output classes to be supplied in the training set. Although some incremental learners are able to cope with unseen targets classes, many require all possible classes to be supplied during initialisation. This introduces a constraint on the system design, if an algorithm is utilised which cannot deal with unseen classes, these must be established a priori. For this research, several algorithms were tested and each was supplied a full list of target classes on initialisation. This meant that some pre-processing had to be performed to establish every possible URL on the target websites.

Imbalanced Classes Imbalanced classes are a very common issue in machine learning. The issue occurs when one output class is over-represented in the training data, the model will then tend to disproportionately assign inputs to the over-represented class. Most algorithms perform best when the instances of each class are roughly equal. In this research domain, imbalanced data is common, as there is often a large discrepancy in the number of visits to each URL on a domain. The methods employed to tackle the imbalanced data, in this thesis are described in section 7.3.3.

3.6.3.1 Propensity Prediction Service API

The design goal of the PPS API is to provide a simple, flexible interface to obtain webpage predictions for a user in real-time. The simplicity of the API is achieved by providing a set of reasonable default behaviours based on the most prevalent use cases. To attain more fine grained control, an advanced user can use the API parameters to alter these defaults.

The PPS utilises an ***event processor*** which collects data in an event-based style via a RESTful API. A website can send data to the PPS event server through its API via HTTP requests. Following the industry standards [108] the API:

- Is Exposed over HTTP
- Uses the HTTP protocol RESTfully
- Uses JSON as the data format

API interfacing can be divided into ***Training*** and ***Prediction***. Offline training of a stereotyped model can occur through the training method of the API. The real-time actions of the service provide predictions while also optionally training the model. When a prediction request is made, the current user model is supplied as an input to the service. If no such model is provided, the stereotyped model will be used to perform prediction.

API Parameters	**Description**
User Data	The recent user interactions which determine the prediction. These will also be used to update the user model if training is active.
Available Links	The webpages for which the PPS should provide probability estimates.
Current User Model (Optional)	The current model of user behaviour, the stereotyped model will be used if none is supplied.
Training	Whether or not training should be performed i.e. whether the user model should be updated. The default is False.
Outputs	**Description**
Predictions	The predicted probability of clicking on each link in the provided available links.
Updated User Model (Optional)	The user model updated to reflect the new user data.

Table 3.2: PPS API Description

Each prediction can also specify whether the model should be updated through training with the supplied user data. This may be used for a cold start user until a reasonable model has been established, or when an outdated model needs to be retrained. Table 3.2 shows the inputs and outputs to the API, illustrating its usage.

For a prediction to be made by the PPS, all events required for feature calculation must be supplied as this represents the current 'state' of the user. For example, if the prediction is based only on clicks and mouse movements, a prediction cannot be made until the mouse has been moved and at least one click has been made. While waiting for all events to occur some storage of data must therefore occur, i.e. the client

coordinator must store the most recent events in the user data. To reduce the risk of a long or even infinite waiting time, value aggregation can be employed. For example, instead of one feature representing whether a click was made and another representing whether a right click was made, a single feature with three possible values can be used i.e. no click, left click and right click. For this research, this solution was sufficient as all event types were performed by the user very soon after their first interaction. For more complex cases, a bag-of-words model could be used to simplify the feature representation further, otherwise the event that has yet to occur could be treated as missing data. Common methods in ML for dealing with missing data could be deployed, for example, deletion, mean imputation, or even applying a predictive model to infer the missing value [109].

3.6.4 Additional Use Cases

In this research, the prediction from the PPS is used both to guide prefetching and personalisation. There are several other use cases for the prediction in the context of web personalisation, these are detailed below:

Persuasive Navigation This involves choosing content or links based on their likelihood to lead a user to a desired action. For example, a political campaign website aims to raise as much funding as possible for a candidate; the desired action is that a user on the website clicks 'Donate'. Suppose a user reaches the campaign homepage and clicks on the Bio link to learn more about the candidate, The website calls the Propensity Prediction Service to predict where the user will click next if shown three alternative content layouts for the Bio webpage. For each content option the PPS reports the likelihood of the user clicking on a particular link on the webpage. The content option which gives the highest likelihood that the user will choose to donate to the campaign, is then chosen as the content to be displayed.

Adaptive Content Display Adaptive content display involved changing content based on the user's propensity. For example, suppose a user is browsing books on an eCommerce website. The website interfaces with the PPS to determine the user's current propensity to click through and purchase the items in their basket. Finding that the propensity is low, the website decides to display a one time discount coupon for this user. This is similar to the personalisation applied in this research.

Path Optimisation A website often simplifies user tasks by reducing an action normally taking 5 clicks and textual input, down to 2 clicks. This is high-risk personalisation as a mistake will leave the user very dissatisfied. As the website queries the PPS to predict the user actions, it also requests the confidence readings. High-risk personalisation of this form is only applied in cases where the confidence is above an acceptable threshold.

3.7 Chapter Summary

This chapter outlined the design requirements and challenges for an ICCP framework along with its implementation for this thesis. State of the art research, detailed in the previous chapter, established a set of considerations which influenced the final ICCP design. The privacy achieved by the ICCP framework through protection of user data was outlined, along with the predictive prefetching strategy which aimed to improve latency over the SOTA systems.

The high level design, and each of the components required in the ICCP framework were then detailed. The framework uses client-based coordination to orchestrate microservice interactions with a personalisation service and a user modelling service, gathering, storing and managing user data. The discussion of the client-coordinator along with these two supporting microservices, detailed the necessary framework configurations and their possible impact on latency. The evaluation of these configurations is described in Chapter 6.

Finally, in exploring the design and implementation of the propensity prediction service, for user modelling, the requirements for privacy conscious prediction were established (Section 3.6.3). This explored the limitations on user models and learning algorithms suitable to maintain privacy of user data in the ICCP system.

Chapter 4

Evaluation Methodology

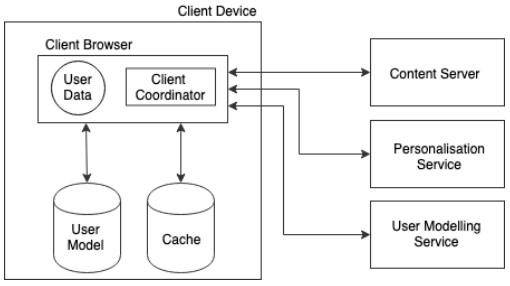
Following on from the discussion in the previous chapter, the performance and privacy of the ICCP is now examined. Specifically, when evaluating performance, the **user latency** is examined, i.e. the average time it takes for a webpage to load for a user.

The hypothesis in this research is that a client-side personalisation framework with the addition of prefetching will address the latency problem and still enhance the privacy. Thus, in order to evaluate the latency, a comparative approach is followed to specifically examine the effect of adding prefetching. This uses two frameworks as baselines with which to compare the proposed ICCP framework; a traditional content-sever based framework and a distributed client-side personalisation framework without prefetching. This comparative approach aims to establish the trade-offs between the frameworks and motivate the use cases for appropriate application of an ICCP framework. The evaluation is based on the streaming of real user data, as this allows for controllable, reproducible, and scalable experiments. Thus, a comparative, case study based approach is used, comparing the frameworks through the use of three contrasting website use cases.

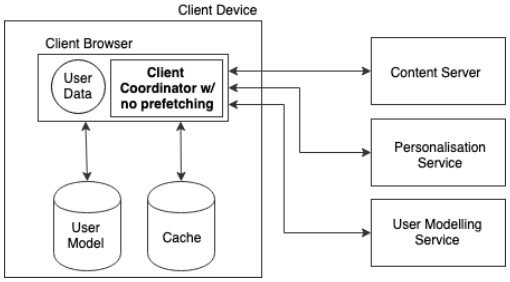
This chapter outlines the evaluation methodology used to perform this comparative evaluation, discussing the design of the comparative frameworks, the case study based methodology, and the process used to stream real user data.

4.1 Comparative Evaluation

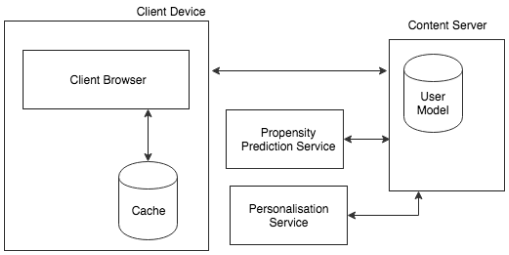
The comparative evaluation is performed across three frameworks, the ICCP, a distributed client-side personalisation (DCSP) framework without prefetching and a client-sever based framework. Comparison with the DCSP framework allows this research to examine the improvement in latency gained by the ICCP through the addition of prefetching. Measuring this latency against the traditional client-server framework then allows the trade-off between the performance and privacy of each to be examined. The implementation of these frameworks are described in the following sections with the differences summarised in Table 4.1.



(a) ICCP Architecture



(b) DCSP Architecture



(c) Client-Server based Architecture

Figure 4.1: Comparative Evaluation Architectures

	ICCP	**Client-Server Based**	**Distributed CSP**
Coordination Logic	Client-Side	Server-Side	Client-Side
Personalisation	External Microservices	Microservices	External Microservices
User Modelling	External Microservices	Microservices	External Microservices
User Data Storage	Client-Side	Server-Side	Client-Side
User Model Storage	Client-Side	Server-Side	Client-Side
Prefetching Logic	Client-Side	Server-Side	N/A
Prefetch Cache Storage	Client-Side	Server-Side	N/A
Regular Cache Storage	Client-Side	Client-Side	Client-Side

Table 4.1: Comparison of Frameworks

4.1.1 Distributed CSP Framework

In order to evaluate the effect of prefetching on the latency of a distributed CSP architecture, the prefetching mechanism of the ICCP framework is separately examined. As seen in the state of the art chapter, there is no single architecture for a distributed CSP system. However, as this research is concerned with improving the latencies of such systems through prefetching, the ICCP framework with prefetching removed can be considered representative of a distributed CSP framework. This comparison allows the effect of the prefetching on latency to be evaluated.

4.1.2 Client-Server Based Framework

The client-server framework used in the comparative evaluation represents a client-server architecture approach [110]. The personalisation and prediction are both coordinated by the content server instead of by the client. Thus, the content server harvests and stores all of the user data and the user model. Performing a comparative evaluation with this framework enables this research to examine the trade-offs between privacy (as defined

in chapter 1) and latency. The ICCP provides better privacy protections of data leakage over the client-server. This is achieved by storing the user model on the client and only allowing this to be streamed and not stored in the external services. The impact of this privacy conscious approach on the latency performance of the framework can be evaluated through the comparison with the client-server framework. This motivates the use cases for appropriate application of an ICCP framework.

The content server is implemented in a microservice style, consistent with the architecture of the ICCP. However, each of the microservices are implemented on the content server itself, meaning that no external requests are made. As discussed in section 3.2.3.1, the microservice architecture is becoming increasingly common and represents a large proportion of current personalisation systems. Thus, this framework forms a reasonable baseline of current architectures for comparison to the ICCP. The architecture of the client-server based framework is illustrated in Figure 4.1c.

The **prefetching logic** in the client-server framework lies on the content server, that is the mechanism for deciding when to prefetch a webpage based on user actions as well as which webpages should be prefetched. This logic is represented in the evaluation setup as **activity thresholds** and **prefetch threshold** values as discussed in chapter 3. Such logic would be deemed intellectual property as it is key to the performance and effectiveness of a prefetching strategy. Thus, the caching of prefetched and pre-personalised pages also occurs on the content server, this is implemented as a middleware cache. Regular resource caching, for pages or resources already seen by the client occurs on the client-side in both frameworks.

4.2 Case Study Based Methodology

Case studies are a valuable research methodology to aid understanding of a particular problem or situation in great depth [111]. Specific information-rich cases are selected to fully explore a research question. As this thesis aims to investigate the specific challenges involved in an ICCP framework, a case study based research approach was selected to enable a comprehensive study of real-world applications of this framework. These case studies consisted of real websites where an ICCP framework may be employed.

In selecting case studies, a representative sample of web traffic was desired. Zviran et al. [112] classified high-volume websites into five categories:

(i) Publish/subscribe websites which provide users information such as search engines, media websites, and newspapers;

(ii) Online shopping websites which let users browse and buy;

(iii) Customer self-service websites which let users help themselves ; such as banking at home, tracking packages, and making travel arrangements;

(iv) Trading websites which let visitors buy and sell, and

(v) Business to business (B2B) websites which let businesses buy from and sell to each other [113].

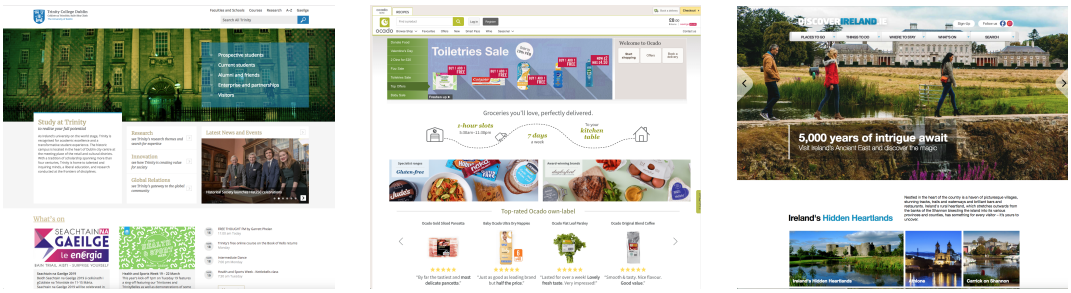
For this research the last two categories were excluded as recent surveys show these do not represent a significant proportion of Internet traffic in comparison to the former [114]. Thus, with these categories assigned shorthand names, the three website use cases selected were:

(i) **Informational:** www.TCD.ie - An Irish University website

(ii) **E-commerce:** www.Ocado.com - An online supermarket based in the UK

(iii) **Commercial:** www.DiscoverIreland.ie - The national Irish tourism information website

These websites represent high traffic sites in their respective categories, each is described in the following paragraphs with their homepages illustrated in Figure 4.2.



(a) TCD.ie (b) Ocado.com (c) DiscoverIreland.ie

Figure 4.2: Website Case Studies

TCD.ie is the main website for Trinity College Dublin, the number one research University located in Dublin, Ireland. The website offers information about the college and its courses, is in the top 50 websites in Ireland and receives over 1 million visits a month [115]. User behaviour on this website represents *information seeking behaviour*, where the user has a very specific piece of knowledge to gain and quickly traverses and skims pages.

Ocado.com is an UK based, online only supermarket. The website sees a average of 1.4M users a month with 4.8M visits in that time. Users also average 11.3 page views and 10 minutes per day on the website [116]. As the website is focused around ecommerce, the behaviour of users here is *transactional*. They are interested in fulfilling a specific goal but with some additional parameters, finding products they want with context like price.

DiscoverIreland.ie is a consumer website operated by Failte Ireland, the tourism board of the Republic of Ireland, and the Northern Ireland Tourist Board, It features information and listings for Irish accommodation, activities, events and tourist attractions. Here the user does not know the solution to their goal, therefore they engage in *explorative* behaviour while they are using the website. This involves them taking time to navigate images and text to find a solution.

4.3 Data Streaming

Web architectures are often evaluated using simulation-based experiments, this technique is particularly common in evaluating caching strategies [75, 36]. Rather than collecting real data, these emulate user behaviour based on probability models from previous user behaviour studies. For example, knowing how often webpages are typically revisited and the distribution of this value across different websites, we can create simulated user data which incorporates this probability model.

For this research, several types of data were required, the page revisitation patterns of a user over a single session, user dwell time and interaction data such as mouse movement and click patterns. As well as this, event distribution data was required which gave a time profile of these interactions on a single webpage. However, existing simulation models did not provide sufficient data, lacking in both intra-session revisitation data and event distribution data. Thus, the collection of real user data was required.

The following sections outline the existing models of user interaction behaviour, justifying the need to collect real interaction data. The method of streaming this data through the ICCP framework is then described.

4.3.1 Existing Simulation Models of User Interactions

Examining the state of the art, we find existing studies for ***page revisitation patterns, user dwell time, and click speed patterns***, these are detailed in the following sections.

Revisitation Patterns A large proportion of webpages visited online are revisits [117]. Different studies give figures from 45% [118, 119], to 80% [120]. However, these figures reflect users visiting the same webpage across different sessions. For example, a user checking the homepage of their local newspaper everyday. Models showing the short-term revisitation patterns of a user in a single session i.e. intra-session revisits, are sparse. Cockburn and McKenzie [121] found that recently consumed items are likely to be consumed again and short term visits have been found to be very common in several other studies [122]. However, none of these studies presented models estimating such revisitation patterns, and no model for intra-session web revisitation can be established from the literature.

User Dwell Time Dwell time has been found to be highly dependent on the website and context in question [123, 124]. For example, a search engine webpage will tend to have a much shorter dwell time than websites related to entertainment. However, some patterns can be established. Yi et al. [124] performed an in-depth analysis of user dwell time from a large scale yahoo web log, aiming to use this as a proxy for user interest. The study modelled user dwell time across different websites and contexts, finding that the distribution of dwell time between users follows a bell-curve. This distribution holds for different time periods and different types of devices. The study also found that the length of an article has good linear correlation with the average user dwell time. However, the correlation becomes weaker when articles are very long, with a big variance when the article is longer than 1,000 words. The device being used (e.g. mobile phone or laptop) also has an effect on user dwell time, with users spending on average less time per page on mobile or tablet devices than on desktops. The paper also presents a machine learning method to predict dwell time for articles, obtaining good results using only the article topic and the device used to read the article as features for the model. There are many further studies modelling dwell time for users on the web [125], thus, sufficient data exists for a model of dwell time to be established. This could then be used in a simulation based evaluation.

Click Speed Patterns As click data is commonly used in web research, particularly in understanding search behaviour, there exists much research recording click speed patterns. For example, one study [126] found that users have fairly unique and stable distributions in their double click speeds, with values varying between 10 to 50 ms.

However, no literature could be found regarding typical ***event distributions*** over a single webpage. That is, a time profile of user interaction events in a single webpage. Time profiling across a single webpage was essential to this research, as it controlled the quantity of server interactions and the client-coordinator overhead. Secondly, interaction behaviour based predictions could only be achieved with genuine user data.

As there are no open datasets with event distribution behaviour and no good models for intra-session webpage revisitation, real user data must be collected for this research. This data would aim to establish a full picture of user interaction behaviour.

The data collected during this research is discussed in chapter 5. This is analysed and compared to the state of the art outlined in section 5.7.

4.3.2 Streaming Real User Data

In order to run the evaluation under realistic settings, real user data is used. Thus, the frameworks are evaluated on the basis of authentic user behaviour, rather than distribution based behaviour models.

Each website used to collect data represents a varied mode of behaviour for users, and thus a different behaviour model. The three case studies introduced behavioural variations in the data, like the frequency of page requests, the prevalence of different event types, and the quantity of interactions per page. These are summarised in the next chapter, section 5.7. These behavioural variations translate to different load profiles for each of the websites. The processing burden on the services, the accuracy of the predictions and the number of network requests are a few factors changed by the specific behaviour model. Simulating these profiles allows an investigation into the effect of user behaviours on the framework performance.

The data streaming requires taking all of the data gathered and sending it through each framework as if the user actions were occurring in real time. Puppeteer[1] with Headless Chrome[2] was used to accomplish this. It provides a high-level API to control the browser, Chrome, over the DevTools Protocol[3], enabling automated testing. Almost everything that a user can do manually in a browser can also be done in an automated way with Puppeteer. For the purposes of this research the key aspects of Puppeteer that were used included, automated clicking and typing, opening tabs, interacting with forms and dropdowns etc. This allowed the same user session and actions to be replayed in the browser over and over in different testing environments as if the user was interacting with the webpage live.

4.3.2.1 Dummy Website

To test each of the behaviour profiles under the same conditions, a dummy website was created. The actions on each of the three websites were mapped to this site. This normalised the background processes for each behaviour model, standardising the number of resources being fetched per page and the quantity of data being rendered.

The dummy website had a consistent content size, with each webpage containing text, a video and images. Technically these pieces of content were identical, however the system treated each as a new resource. This maintained sensible resource caching, but kept the content load stable on each page. The full content profile for loading an unseen page is listed in Table 4.2, showing a total size of 1.9MB. This content size is approximately equal to the average size (2.05MB) of the landing pages for each of the three use case websites used in this research. Therefore, it represents a realistic real-world webpage content profile. For the other page types, some of these resources will be cached and/or prefetched changing the source of the resource and reducing the number of network requests.

[1] https://github.com/GoogleChrome/puppeteer

[2] https://developers.google.com/web/updates/2017/04/headless-chrome

[3] https://chromedevtools.github.io/devtools-protocol/

Resource Type	Source	Quantity	Size (kB)
Local Images	Content Server	3	42.8
External Images	External URL	6	183.7
Video	External URL	1	1100
Javascript Libraries	Content Server	6	518.9
CSS	Content Server	2	51.1
Page Template	Content Server	1	3.4
Context	Content Server	1	4
Personalised Page	Personalisation Service	1	7.8
Total	-	20	1911.7

Table 4.2: Content Profile for an Unseen Page

Benchmarking the framework is used to establish the performance when no background activities are ongoing. This means that no prefetching or prediction are performed by the framework. This allows the effect of these background activities to therefore be measured. Benchmarking involves repeatedly requesting pages and measuring the page load time.

The research aims to benchmark a range of different page types, for each of the frameworks. Specifically:

- Pages never seen before by the user
- Pages prefetched in the background
- Pages previously visited by the user
- Pages previously prefetched by the system but not yet seen by the user

These are described in more detail, later in this chapter (section 4.4).

Each time a request is made, the client must be reset as appropriate for the page type, for example, when requesting an unseen page the cache must be cleared on the client or server after each page request. For a request of a previously prefetched page a prefetch must be forced by setting a prefetch threshold of zero, i.e. telling the framework to fetch every possible next page. Finally, for a cached page the page must simply be visited before the request.

Mapping the data to the dummy website consists of assigning each unique URL in the data to a reciprocal URL on the dummy website. Thus, page requests can be accurately reflected and each correctly models the user's previous interaction with the page. The *page type* for each is therefore maintained.

Puppeteer is then used to simulate all of the user actions taken on each page e.g. clicking and moving the mouse. Although, these actions are not relevant in the context of the dummy website, they elicit network and server requests on each framework. The visual layout of the webpage is not relevant in any framework. The learning models used, also correctly reflect the website from which the data originated, thus only raw interaction data is required to simulate predictions.

4.4 Types of Page Request

In evaluating the response times, we establish the different types of page request that can be made by the user. These are classified based on the caching mechanism.

Unseen Page A page which has never been previously requested by the user

Prefetched Page A page which was prefetched in the background since the last page change. This means both the page and the page resources will be cached.

Revisit A page which has been previously visited by the user. Thus the resources for this page are cached.

Previously Prefetched This is a page which was prefetched on an earlier page but has not been visited by the user. As it was not requested by the user after being prefetched, it represents a prefetch cache miss. However, the resources will have been cached speeding up later retrievals of this page. The page itself is deemed 'stale' and thus must be fetched again.

The caching mechanisms for each page type is summarised in Table 4.3. A revisited page and previously prefetched page are grouped together in this research as a cached page. While these page types have different use cases, they are identical in the caching and network request profiles.

	PageType			
	Unseen Page	**Prefetched Page**	**Revisit**	**Previously Prefetched**
Resources Cached	✗	✓	✓	✓
Full Page Cached	✗	✓	✗	✗

Table 4.3: Page Type Caching

The network requests in serving each of these page types are shown in Table 4.4. To enhance performance, there is no PPS request when serving a page. Instead prediction occurs in the background while the user is interacting with the page.

	Comparative Framework		ICCP		
Page Type	Content Server	Resource Requests	Content Server	Personalisation Service	Resource Requests
Unseen	✓	✓	✓	✓	✓
Cached	✓	✗	✗	✓	✗
Prefetched	✓*	✓†	✗	✗	✗

Table 4.4: Network requests required to load each page type

* Served from the prefetch cache

† May be reduced if prefetched page has been seen

4.5 Evaluation Setup

4.5.0.1 Evaluation Environment

For the evaluation, the Content Server, PPS and Personalisation Services were deployed on a virtual machine external to the client. This server was hosted in Trinity College and utilised high speed LAN with WAN connectivity via HeaNET. The client-device was a Desktop Mac, this was connected to a dedicated LAN network over WiFi, no other devices were connected to the network. The specifications for both the server and client used are listed in Table 4.5.

4.5.0.2 Learning Models

Each of the evaluation objectives, discussed in chapter 6, assumes that a separate prediction model exists for each of the three websites. This serves as a stereotyped model on which each user can base their initial learning model. Thus, the user data is split into a training and test set with a 0.6/0.4 split. This is further detailed in section 7.3.3.

Parameter	Client Configuration	External Server Configuration
OS	Mac OS X 10.12.6	Debian GNU/Linux 8.10 (jessie)
CPU	Intel Core i5-2400 CPU @ 3.10GHz	QEMU Virtual CPU version 2.5+
CPU Cores	4	1
RAM	8GB	1GB
Network	AirPort Extreme (0x14E4, 0x843)	RTL-8100/8101L/8139 PCI Fast Ethernet Adapter
Network Speed	100Mb/s	1Gb/s
Apache Version	N/A	2.4.10-10+deb8u11
Hoxy Version	N/A	3.0
Browser	Headless Chrome 71.0.3578.98	N/A

Table 4.5: Evaluation Environment

4.5.0.3 Framework Configurations

Settings specific to the frameworks used in the evaluations are summarised in Table 4.6. This illustrates the activity thresholds chosen for each of the event types along with the proportion of those events occurring in the data collected. The data gathering process is discussed in the next chapter. As the effect of refreshing is not examined in the evaluation of this research, the refresh threshold is disabled.

4.6 Chapter Summary

This chapter outlines the methodology used to perform the comparative latency evaluation for this research. The architectures and implementation of two frameworks are described:

(i) a traditional client-sever based framework

(ii) a distributed client-side personalisation framework without prefetching.

Parameter	**Configuration**	**Proportion of Total Events**
Prefetch Threshold	0.7	
Activity Thresholds		
Click/RightClick	1	0.1
MouseMove/MouseLeave	10	0.76
Scroll	3	0.12
Typing/Paste	1	0.018
Copy	inf	0.002

Table 4.6: Framework Configuration and Event Proportions

These will be used for comparison against the ICCP framework, aiming to investigate the impact of prefetching and the latency and privacy trade-offs of the different systems. Where the traditional client-server based framework performs all operations on the server side, the ICCP utilises the client and microservices. The DCSP framework then replicates the ICCP design removing the prefetching operation in order to isolate and evaluate the effect of prefetching. Specifically, the latency impact of the framework choice and its configuration will be investigated through comparison of these three frameworks.

The case study based methodology is then discussed, outlining the use cases employed in this research. These were chosen to elicit varied interaction data from users which could then be used to test the frameworks under different load profiles.

The method used for streaming this user data through the frameworks was then outlined. This utilises a dummy website with a consistent content profile to isolate the effect of user interaction data on framework performance.

Finally, the evaluation environment and setup were given, providing context for the evaluations performed in Chapters 6 and 7.

Chapter 5

Data Gathering

As outlined in the previous chapter, the evaluation of the ICCP framework in this thesis is based on the streaming of real user data. This chapter outlines the types of data collected, as well as how the data was collected, verified and processed. It also presents some analysis over the collected user data.

5.1 Prolific Academic

To comparatively evaluate the ICCP framework, a dataset reflecting user interactions with online websites was required. The participants for this experiment were gathered through Prolific[1], a platform for crowdsourcing research participants. As it would have been difficult to realistically reflect website complexity through construction of a model website, a proxying method was used to allow users to interact with real websites, while their interaction data was tracked. Users interacted with three different websites and completed a set of given tasks, designed to produce deviation in behavioural patterns.

The three websites used were:

[1] www.prolific.co (Formerly known as Prolific Academic)

- **E-commerce** Ocado.com, An online supermarket
- **Informational** TCD.ie, University Website
- **Commercial** DiscoverIreland.ie, Irish tourism information

The selection of these websites is discussed in section 4.2 in Chapter 4.

Power Test

In order to determine the number of users required for the evaluation, power tests were performed.

For the *performance evaluation* (Chap 6), the response time was a dependent variable with the independent variable being the framework used i.e. the attribute being changed and evaluated. The statistical test that was therefore performed for this part of the evaluation was the Paired samples t-test. A power test with an effect size of 0.4, an error probability of 0.05 and a power of 0.95, requires a sample size of **70 users** for this test.

For the *prediction evaluation* (Chap 7), the guidelines [127] yield a minimum sample size as follows:

$$(\#predictors * 10) \ \ to \ \ (\#predictors * 30)$$

In this research, 14 predictors were employed, yielding a desired sample size of 140 - 420 thus settling on the midpoint of **280 users** for the experiment.

For the performance evaluation, it is necessary to evaluate the frameworks under different use cases thus all sites must gather data for over 70 users to perform this evaluation. However, the prediction evaluation did not require a comparison across use cases in order to evaluate the usefulness of mouse dynamic data. Therefore, only the e-commerce site needed to reach the threshold of 280 users for this evaluation.

	Number of Users
E-Commerce	304
Commercial	87
Informational	91

Table 5.1: Number of Users per Site

Pre-screening Some pre-screening was performed on Prolific to ensure the participants were suitable for the experiment. The users had to have previously participated in at least two Prolific experiments, as this experiment was time consuming this restriction hoped to limit the number of users not correctly performing the task described. The users must also be using a Chrome browser on a desktop, laptop or touchpad.

5.2 Proxy Setup

In order to track users on live websites, ***Hoxy***[2], was used, an existing HTTP traffic-sniffing and manipulation tool for JavaScript.

This was used as a reverse proxy, fetching webpage resources on behalf of the client from the website's origin servers. These resources were then returned to the client, appearing as if they originated from the website itself. To a user participating in the experiment, it would appear they were on the actual webpage, with the URL being the only difference. This operation is illustrated in Figure 5.1. All links clicked on the page would also be directed through the proxy.

[2]https://github.com/greim/hoxy/

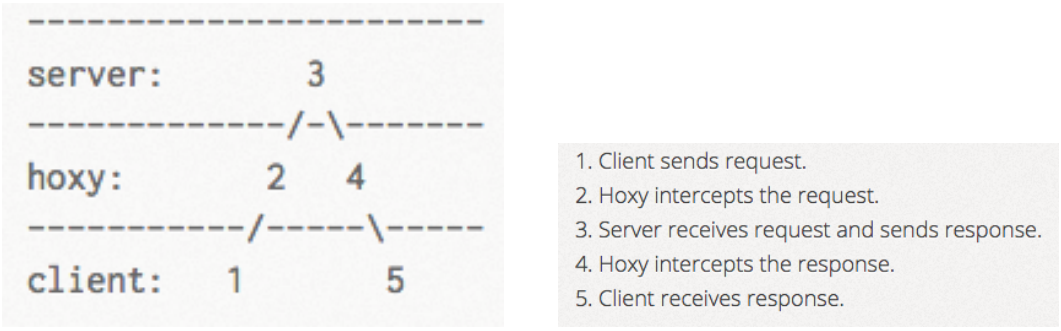


Figure 5.1: Hoxy Operation

Hoxy also enables interception and alteration of the webpage's responses. This allowed changes to be made before the page was rendered to the user. The tracking script could then be injected in the page and immediately start tracking user actions.

Other alterations to the webpages made before rendering included small behavioural changes such as redirecting the 'Checkout' button on the grocery shopping website (www.ocado.com) to complete the experiment.

To reduce the scope of the experiment, pages were also stripped of some elements and links. This controlled the navigational options and possible user actions on each page. Selection of the stripped elements was done through iterative evaluation, described in section 5.4.

Additionally, on each of the three sites some extra event listeners were added purely for error tracking purposes. These watched for mutations in certain elements which were recorded to ensure consistency in the results.

5.3 Types of Data Collected

A large quantity of interaction data was captured in the data gathering stage. This provided a sufficient number of variables to test in the predictive system later on.

To collect interaction data, users were each assigned an ID, as they interacted their events were then written to the database along with the ID. A session ID was also assigned to the user, so that should they revisit the domain in the future their data would not be saved.

A summary of the type of data gathered is listed below in Table 5.2. Each of these events included additional information such as whether a control key was being pressed.

Event Category	**Types of Event**
User Info	SystemInfo, LocationData, ScreenData
Session Data	NewUser, NewSession
Interaction Data	MouseMove, MouseLeave, Click, RightClick, Scroll, Typing, Copy, Paste

Table 5.2: Events tracked during data gathering

Mouse Movement

Recording every mouse movement is impractical, as a single mouse movement can trigger many mouse movement events, causing a huge amount of data to be collected and streamed to the web server. Thus, mouse movements were only recorded after a movement delay following the method set out in Huang2011 [94]. They found that recording cursor positions only after a 40ms pause provided a reasonable tradeoff between data quantity and granularity of the recorded events. This approach records key points of cursor movement, e.g. when the user changes direction, or at endpoints before and after a movement.

5.4 Data Verification

Before deploying the experiment on Prolific, users were recruited for data verification. This aimed to test the tracking code by comparing the data collected with real user actions, through video interviews, and ensure accurate reflection.

During the video interviews ten users were observed, in person, interacting with a webpage. As they interacted, a screen recording device was running to produce a video of the session.

A data simulation program was built which then took the raw user interaction data and played it out on a screen. The simulation was visually compared to the screen recording to investigate errors and highlight any interactions which may have been overlooked e.g. users navigating to a different tab.

The simulation program used Headless Chrome [128] to produce accurate screenshots of the page with which the user interacted. Headless chrome allows execution of the Chrome browser while controlling it programmatically. It runs without its 'head' i.e. the GUI so can be used on servers without dedicated graphics or display. JavaScript was then used to accurately simulate events in an automated browser e.g. mouse movement, clicks and scrolling behaviour.

Data verification was performed iteratively to refine the data collection for this research. As a result, the tracking code was altered ensuring all pertinent user behaviours were captured and the webpages were altered to limit the user navigations and provide error tracking.

5.5 Task Descriptions

The tasks for each website were designed iteratively. In order to assist the design of each task, a number of test runs were performed, recruiting 3-5 users at a time. Issues with the task designs and user behaviours were therefore highlighted and corrected. For example, on ocado.com, it was found that users were regularly checking out with only three items instead of the four required. This was due to the fact that one popular recipe had only three ingredients. The recipe was subsequently removed by the proxy and the issue did not recur.

Each user recruited for the experiment was assigned randomly to perform a specific task. The user performed only one task, meaning they interacted with only one of the three websites used in this experiment. This research was not focused on comparing a single user across tasks but simply building realistic interaction profiles for each site.

Only a single task existed per site, this meant that the prediction was focused on how different users achieved the same task. As outlined at the end of this chapter (Section 5.7) there was significant variation in the methods used to complete the task and the length of time each user took. Each task and the website it was conducted on were designed and selected to gather different types of usage behaviour as described in section 4.2.

5.5.1 Ocado.com

To gather *transactional* usage behaviour for the experiment, users were asked to interact with the grocery shopping website ocado.com and complete a shopping task. The following are the instructions provided to each participant:

- *Imagine you are planning a dinner party; You wish to prepare a chicken dish involving fruit for the main course.*
- *Start the process by selecting the Recipe Tab as in the image below. Browse these recipes to find an appropriate dish.*

- *Once you have chosen a dish, begin the grocery shopping. Please add 4 of the ingredients for your chosen recipe to your basket.*
- *When you have finished, click the 'Checkout' button.*

5.5.2 DiscoverIreland.ie

To gather *exploration* usage behaviour for the experiment, users were asked to plan a holiday with the following instructions:

- *Imagine you are planning a day trip to Co.Meath, Ireland; You will arrive in the morning and stay one night.*
- *Begin by selecting a Guesthouse to stay in; Take note of your selection*
- *Select an adventurous activity nearby; Note the activity name*
- *Once you have completed the task, click the 'Complete' button in the top right corner*

5.5.3 TCD.ie

To gather *information seeking* usage behaviour for the experiment, users were asked to answer a number of questions, to which the answers could be found throughout the website. The following instructions were given:

You are investigating a university which you may attend next year; Explore the website to find answers to the following questions:

- *How many places are available on the Undergraduate course 'Business studies and German'?*
- *What is the telephone number of the Business School?*

5.6 Data Validation & Post-processing

Data validation was performed during and after the data gathering process. This ensured that the participants were engaged with the experiment and their data was useful to the analysis. It involved expunging the data from any participants who did not correctly complete the task or who performed random actions during it.

5.6.1 Ocado.com

During the experiment, a participant's submission was rejected if they were clearly not engaged with the process. The participant's submission was rejected after an attempted checkout if they:

- Did not view any recipe
- Did not add four items to their basket

The participant was offered a second chance to complete the experiment if the number of items in their basket equalled three, as this could have been a genuine mistake.

After the experiment, a post-processing script was applied to the data to determine any other submissions that may not be useful. As a result of this process, participant submissions were rejected if they:

- Had less than one successful checkout
- Did not view a recipe page at all
- Did not view a recipe page before a failed checkout
- Had less than 4 trolley events
- Did not add at least 3 items to their trolley before attempting to checkout
- Did not add any items to their basket matching the viewed recipes

5.6.2 DiscoverIreland.ie

During the experiment, a participant's submission was rejected if they:

- Did not view any Guesthouses
- Did not view any activity pages

After the experiment, a participant's submission was rejected if they:

- Did not record the activity and guesthouse
- Did not view pages correlating to the activity and guesthouse names reported
- Did not constrain their search to the location given i.e. Co.Meath

5.6.3 TCD.ie

During the experiment, a participant's submission was rejected if they:

- Visited less than two pages before entering answers

After the experiment, a participant's submission was rejected if they:

- Did not visit a page containing the answer to the questions asked
- Incorrectly answered the telephone number question. The answer to the first question turned out to have some ambiguity so ensuring they visited a page which contained the answer was deemed sufficient.

5.7 Data Exploration

An initial, exploratory evaluation was performed on the data gathered. This aimed to explore the variation in behaviour on the different website types as well as comparing the behaviour models to those found in the state of the art (section 4.3.1). The behaviours examined include typical time spent on page i.e. dwell time, intra-session page revisitation, and click speed patterns.

5.7.1 Dwell Time

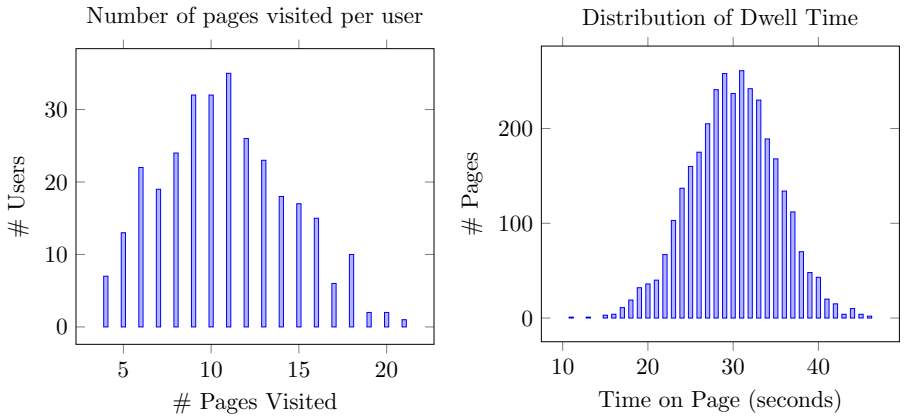
Data from 482 users was gathered for the evaluations. This was analysed to investigate the number of pages visited by each user in a single session and the average time each spent on a page. Table 5.3 gives the average number of pages visited per user across the different use cases. This shows that the average number of pages across sites was quite different, with the average for the commercial site almost three times greater than the e-commerce site. This was due to the specific task assigned on each site with some taking longer or requiring more exploration to complete. The distribution in this average number of pages is illustrated in Figure 5.2.

The dwell time per user across the use cases is then analysed. Again, the results are shown in Table 5.3 with the distribution illustrated in 5.2. On the E-commerce site, users tend to spend much longer per page as the task requires more content consumption than the other tasks, the length of time is also fairly consistent between pages. In contrast on the Informational site, users spend much less time as they are engaged in an information seeking task requiring only scanning of the page content.

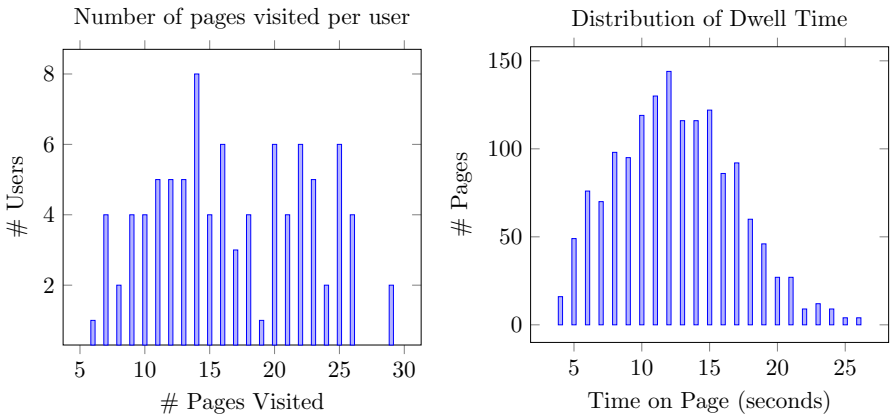
These distributions show that the different sites and different tasks assigned elicited different behaviour profiles with which to preform the performance evaluation.

Statistic	**E-commerce**	**Informational**	**Commercial**
# Users	304	91	87
Avg. # of pages visited	10.8	16.78	32.15
Std.Dev # of pages	3.65	5.89	9.34
Avg. Dwell Time	30.06	12.51	17.65
Std.Dev Dwell Time	5.05	4.38	4.27

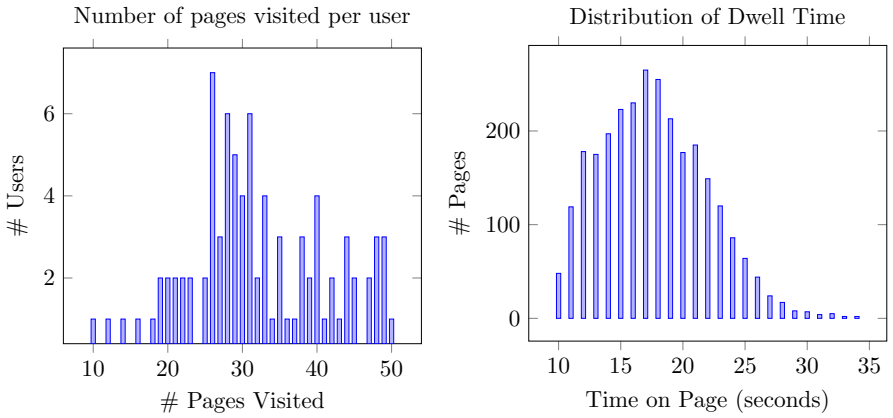
Table 5.3: Statistics Comparing Three Use Cases



(a) **E-Commerce Website - Ocado.com**



(b) **Informational Website - TCD.ie**



(c) **Commercial Website - DiscoverIreland.ie**

Figure 5.2: Page Visitation Statistics Per Use Case

5.7.2 Page Revisitation

Figure 5.3 shows the average percentage of revisited pages versus unseen pages in the user data. The quantity of revisited pages is unique to each website and task. For the e-commerce site we see a lower proportion of revisited pages, likely due to the sequential nature of the task assigned. Whereas with a more exploratory task like in the commercial webpage, a greater quantity of revisits is seen.

The state of the art revisitation statistics explored in Section 4.3, showed that across multiple sessions revisitation to the same website accounted for between 45% to 80% of page requests. Comparing this to the intra-session statistics shown here there is a large difference. This is expected as in a single session the contents of pages is unlikely to change reducing the need for revisitation.

The quantity of revisited pages varies between use cases. For the e-commerce website Ocado.com, we see a lower proportion of revisited pages, likely due to the sequential nature of the task assigned. Whereas with more exploratory task like in the commercial webpage, a greater quantity of revisits is seen.

Click Speed Patterns Examining the click speed for users across the data set shows consistent averages across the use cases. This is an expected result, matching the state of the art previously outlined, which shows that click speed follows a unique and stable distributions per user.

Statistic	E-commerce	Informational	Commercial
Average Click Speed (ms)	101.10	100.82	101.93

Table 5.4: Average Click Speed Across Use Cases

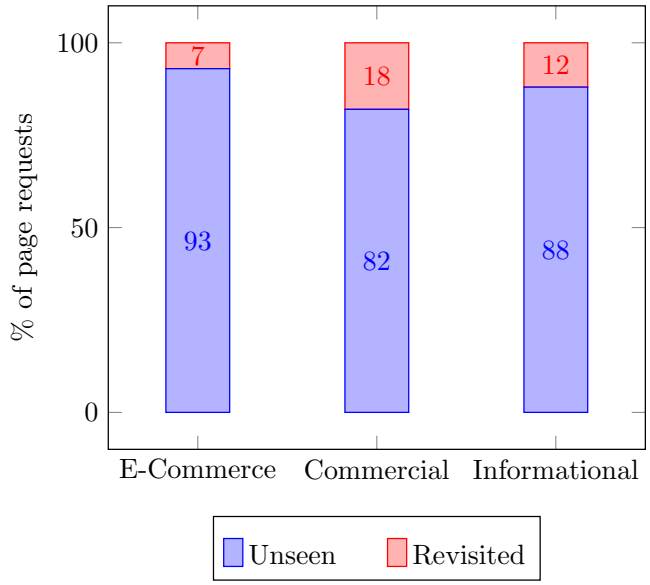


Figure 5.3: Revisits vs Unseen Pages Across Use Cases

5.7.3 Data Exploration Summary

This section performed a brief exploration of the data gathered for use in this research. The exploration showed that the interaction profile of users varied between use cases. This allows the evaluation to be performed under different load distributions.

5.8 Chapter Summary

This chapter outlined the collection of realisitc user data for this research.

Firstly, tracking code was implemented and verified through video interviews. This ensured that data was being accurately collected by comparing a screen recording of participant interactions to a simulation based on the data collected during these interactions. These interviews were also monitored in person to ensure that all relevant data was collected by the tracking code.

Large scale participant data was then collected through Prolific using a reverse proxying method to serve live webpages with this tracking code injected. Three use cases were used to produce three datasets exploring varying user interaction data. The task defined for each use case was outlined in Section 5.5.

The chapter also outlined the types of raw data that were collected for the research (Section 5.3) as well as the post processing and validation of this data (Section 5.6).

Finally, some analysis over the collected user data was presented, showing the different user interaction patterns between the frameworks.

Chapter 6

ICCP Evaluation

The objective of this research, derived from the research question in Chapter 1, is to investigate the design challenges, as well as the performance and privacy trade-offs, for an Intelligent Client-Centric Personalisation (ICCP) framework. The ICCP framework aims to allow both the user and content provider to enjoy the performance benefits of predictive prefetching along with the privacy offered by a distributed client-side personalisation approach.

This chapter presents a comparative performance evaluation of the ICCP framework, against (i) a traditional client-sever based framework, and (ii) a distributed client-side personalisation framework without prefetching. The evaluation uses well established performance tests focusing on user latency.

Ultimately, the chapter will investigate, through comparison, ***the impact of the framework choice and configuration on user latency***. It will examine the effect of two elements:

- The effect of the prefetching strategy on user latency
- The effect of the framework configurations on user latency.

Following the methodology outlined in Chapter 4, a benchmarking evaluation of the frameworks is initially performed, with a subsequent streaming based evaluation using three case-studies.

The conclusion of the chapter highlights the differences between the frameworks, outlining the circumstances under which an ICCP framework provides marked benefits over the alternative frameworks. A methodology for evaluating these benefits given a specific use case is also established. Ultimately, the benefits, challenges and trade-offs of each approach are provided as guidelines for determining when an ICCP framework may offer a practical, more privacy conscious alternative.

6.1 Evaluation Objectives

The success of most online services rely upon user satisfaction, and evaluations of such systems should employ user-oriented metrics. Therefore, in evaluating the performance trade-offs for each framework, a user-oriented view is taken, examining the *user latency*. User latency measures the time taken from a user requesting a webpage to that page being fully served. Throughout this thesis, the terms 'User Latency', 'Response Time' and 'Webpage Load Time' are used interchangeably to refer to the delay seen by the user. It should be noted however, that these terms are often attributed inconsistent nuances in the literature differentiating them from each other. The term 'Page Latency' is used in the thesis to refer to the user latency for a specific page type.

User latency is impacted by several factors:

(i) The page caching i.e. whether the webpage and resources are cached as well as where these are cached

(ii) The prefetching strategy

(iii) The network load and network speed

(iv) The processing power of the client

(v) The processing power of the services (including the content server)

(vi) The service load and the scalability of the services (including the content server)

Given the experimental setup described in Chapter 4, the processing ability of both the client and services along with the service scalabilities are consistent between the frameworks. The network speed while not stable, should have consistent variation while testing each framework. Where the frameworks differ is in **page caching, prefetching, network load**, and **service loads**. Therefore, these are the variations which should be evaluated to determine the impact of the framework choice on webpage load time i.e. user latency.

The **page caching** and **prefetching** for each framework is indicated by the 'page type', as outlined in section 4.4. Each page type represents variations in whether a page was cached and/or prefetched. Where this caching occurred was dependant on the framework, as shown in Chapter 4 (Table 4.4). All frameworks performed client-side cache storage for regular resources, but each varied in where prefetched pages where cached. The distributed CSP did not perform any prefetching, the client-server framework stored prefetched pages on the server, and the ICCP stored them on the client. Evaluation of the caching and prefetching is achieved through investigating the response time, i.e. page latency, for each of the page types.

Both **network load** and **service load** are dependent on the number of calls made to each of the services. This is controlled by the framework thresholds. Thus, the evaluation aims to investigate how the values of these thresholds affect the network and service loads as well as how these can be optimally chosen in an ICCP system.

In summary, to evaluate the impact of the framework choice on webpage load time, we establish the following evaluation objectives:

1. Evaluate the effect of caching and prefetching on user latency

    1.1. Page latency benchmarking across different page types

    1.2. Exploring the impact of background processes on page latency

2. Examine the effect of ICCP framework configurations on the network and service loads

    2.1. Exploring the effect of the prefetching configuration

These objectives are expanded upon in the following sections.

Objective 1.1: Page Latency Benchmarking

To establish page latencies, benchmarking must be performed against each of the three frameworks. This assesses the performance when page requests are made with no background activities ongoing. This means that the content server and microservices should not be dealing with latent requests when the page is fetched. Thus, prefetching and prediction are suspended. Benchmarking tests the best case scenario for each framework, and as such gives the best latency metrics possible. Objective 1.1. also investigates the effect of regular resource caching, comparing the response time of the frameworks when resources are, and are not cached. As discussed in Chapter 4 (section 4.1.1), the architecture of the ICCP framework is very similar to that of the distributed CSP it is compared with. However, the distributed CSP does not employ prefetching. For the benchmarking experiment, the latencies of page requests in the ICCP and the DCSP would be identical. However, prefetched and previously prefetched pages do not exist in the DCSP.

Objective 1.2: Page latency Streaming

When background processes are active, resources are used on the client and server, these may then be locked up as a new page request is made. The aim of objective 1.2 is to explore how prefetching pages and resources impacts the performance of each framework. i.e. does the background process of prefetching impact the framework response times. The methodology discussed in section 4.3 to standardise and stream the behaviour models is followed. The evaluation objective will also analyse the traffic increase seen by the services due to prefetching. Finally, this objective compares the latency effects across the use cases, to investigate the effect of the specific user interaction profiles on latency.

Objective 2.1: Exploring the effect of the prefetch configuration
The prefetching configuration in the ICCP affects both the network load and the load on the services. The configuration consists of two thresholds, the **prefetch threshold** and the **activity thresholds**. This evaluation aims to discuss threshold optimisation along with investigating the impact of different thresholds in this research.

6.2 Evaluation Settings

This section outlines the framework settings for each of the evaluation objectives outlined above. These settings vary in the data streaming method, whether background activities were active and the resource caching. Table 6.1 shows which parameters apply to each objective.

Parameter	**Configuration**	**Objective**		
		1.1 Benchmarking	1.2 Streaming	2.1 Prefetching
Data Streaming Method	Page Requests	✗		
	Data Streaming		✗	✗
Prefetching Active	False	✗		
	True		✗	✗
Prediction Active	False	✗		
	True		✗	✗
Client-Side Caching	False	✗		
	True	✗	✗	✗

Table 6.1: Evaluation Objective Settings

6.3 Metrics

6.3.1 Performance Evaluation

The evaluation in this chapter aims to compare the performance of each framework. Latency is a widely used measure to assess framework and service performance [129, 130], evaluating the efficiency and reliability of a system [131, 132]. It is defined as the delay from input into a system to desired outcome. As discussed in Chapter 2, section 2.1.4, each of the state of the art systems examined as part of this research, use response time as a performance measure, this is also referred to in this thesis as user latency or webpage load time. To further quantify the user latency, a second latency metric, service latency is also examined.

As the evaluation also examines the effect of prefetching on latency, metrics for evaluating prefetching will be utilised, specifically the latency per page ratio and the traffic increase.

The metrics used in the performance evaluation are defined in the following paragraphs.

User Latency The User Latency gives the average of the response times observed by users during a test. Where the response time can be defined as the delay from a user click to full rendering of the next webpage or resource i.e. the page load time. A high response time indicates poor performance. This incorporates the network transfer time, the response time of the micro-services, as well as the time taken for the client coordinator tasks. To further analyse the contribution of microservices to the latency observed by the user, we also examine the service latency.

Service Latency The service response time is characterised by both the network time and the processing time. As the infrastructure used for the evaluation is not reflective of that which would be utilised in a large-scale deployment, the service efficiency is far from optimum. In a large-scale deployment the services would, for example, have a lot more compute power and memory available. Microservices built on high performance servers can vastly reduce processing time, however, the network time is unlikely to be affected. Thus, this is an important metric in evaluation.

Latency Per Page Ratio The latency per page ratio is an evaluation metric for prefetching established in section 2.2.2.2. This is the ratio of the latency that prefetching achieves to the latency with no prefetching. In this evaluation, this is established by comparing the user latency for the ICCP framework which includes prefetching to the distributed CSP framework which does not. The client-server framework is not compared using this metric as the architecture is different than that of the other frameworks, thus the effect of prefetching would not be tested in isolation.

Traffic Increase This metric allows the overhead of prefetching to be analysed from the service provider's perspective. This will analyse the increase in network requests on each of the services.

6.3.2 Prediction Evaluation

A full evaluation of the predictive accuracy of the Propensity Prediction Service (PPS) developed is provided in Chapter 7. In this chapter, the predictive accuracy is evaluated in terms of its effect on prefetching. The Prefetch Hit Ratio and the Precision, are the metrics used as discussed in section 2.2.2.2. In the ICCP framework the outcome for these metrics is based upon the framework configurations. The metrics are summarised again below.

Recall i.e. Hit Ratio

This is the number of successfully prefetched pages over the total number of requested pages; see equation 6.1. The metric measures the effectiveness of the prefetching strategy and thus, the prefetching configuration.

$$PrefetchingHitRatio = \frac{\#PrefetchHits}{\#Requests} \tag{6.1}$$

$$\#Requests = \#PrefetchHits + \#PrefetchMisses \tag{6.2}$$

Precision

The ratio of prefetch hits to the total number of objects prefetched.

$$P_c = \frac{\#PrefetchHits}{\#Prefetches} \tag{6.3}$$

Where a PrefetchHit is a page that is requested, and has been prefetched since the last page request.

6.4 Limitations

The following aspects of the experimental setup potentially limit the conclusions that can be drawn from the streaming study in comparison to a realistic test environment.

Network Jitter Network latency introduces an unpredictable effect in the evaluation of the system due to Packet Delay Variation (PDV) or network jitter. This is defined as the variation in latency as measured in the variability over time of the end-to-end delay across a network. PDV is expressed as an average of the deviation from the network mean delay. To remove this effect, evaluation would need to be performed at scale. Ensuring a quality network connection, enough bandwidth, and a powerful client, can help reduce network jitter. Studies have shown that webpage response times typically

model as a right-skewed long-tail distribution [133]. To mitigate a possible external change in network load the frameworks were compared simultaneously. It is possible that this had the adverse effect of contributing to the jitter seen in the experiments as the machines competed for the same resources.

Testing Environment The frameworks were compared under only one environment. To expand the results of these tests and verify the consistency of the results, the tests should be expanded to multiple browsers and operating systems with varying performance capabilities.

Participant Behaviour The study assumes that users on Prolific act as they would on any other website. In reality, it is likely their behaviour is faster and less considered. The effect is minimised through data verification as outlined in section 5.4. Furthermore, any speed up in behaviour will cause the simulation frameworks to be put under more load than would be expected in real browsing. As these deviations are consistent while testing each framework, the data allows for an accurate comparison of the frameworks to be made.

6.5 Results and Analysis

This section presents the results and analysis of the comparative study. Its structure follows the evaluation objectives. The first two subsections examine the latency of the frameworks across different page types, beginning with benchmarking these page latencies (6.5.1), then exploring the impact of the prefetching process (6.5.2). The final subsection then studies the impact of framework configurations on the network and service loads, namely the prefetch threshold (6.5.4). Each subsection first presents and analyses the outcome of the evaluation metrics, and then summarises how the results address the evaluation objective.

6.5.1 Objective 1.1: Page Latency Benchmarking

The first set of results aims to evaluate the latency of each framework across the three different page types, outlined in section 4.3. This will establish and compare the average time a prefetched page, an unseen page, and a cached page take to load for a user.

This evaluation *benchmarks* these latencies when no background activities are ongoing in the framework. This means that the content server and microservices are not dealing with latent requests while a page is being fetched. Thus, prefetching and prediction are suspended. For each of the three page types (prefetched, unseen and cached), 1000 page requests are made with Puppeteer as outlined in section 4.3.2.1. For this objective only two metrics are utilised, the user latency and the service latency. With no prefetching active, the remaining metrics are not relevant to the evaluation.

The benchmarking results can then be compared to the results in section 6.5.2 to examine the effect of these background activities.

6.5.1.1 User Latency

Each page type varies in its caching; both whether the page is cached as well as where it is cached. This is summarised in Table 4.3 and Table 4.4. The user latency is affected by this page caching as well as the number of service calls made to serve a requested page. For the ICCP, page loading can involve service calls to the Content Server and the Personalisation Service. For the client-server Framework only a call to the Content Server is ever made. Both frameworks may also make several resource requests when loading the page. The requests made by each framework are again summarised in Table 4.4.

Unseen Page

The user latency observed for an unseen page is summarised in Table 6.2. Comparing the frameworks, we see that the average response time of the ICCP is 10.6ms or 5.88% higher than that of the client-server framework. An Independent-samples t-test shows a statistically significant difference between these response times ($p < .05$).With the given number of tests this detects an effect size of 0.25 with a statistical power 0.95%.

Metric	Client-Server	ICCP
Average Response Time (ms)	180.42	191.02
Std.Dev	22.02	21.84
Min/Max (ms)	138/258	153.4/278.2

Table 6.2: User Latency for Unseen Page

There are two contributing factors to this response time difference. In the ICCP framework, there are more service requests and thus network latencies, in addition there is more load on the client-device to coordinate these calls and the results. In section 6.5.1.2 the contribution of service calls to this result is examined. This shows that the increase in network latencies contributes roughly 6ms of this time.

The difference in the standard deviation of 0.18ms, has a small significance and as such suggests some difference in the frameworks. For example, in the variability of the client-coordinator or content server processing times. This is again addressed in section 6.5.1.2. Figure 6.1 illustrates this further, showing the variability between response times across tests. The high peak outliers here are caused by the network jitter as discussed in section 6.4. This jitter is reflected again in the service latency evaluation. For both frameworks, an uptick in average latency is seen at the tail end of runs, this would indicate an external increase in network load as the evaluations are performed simultaneously. This was performed on different machines with identical specifications on the same network.

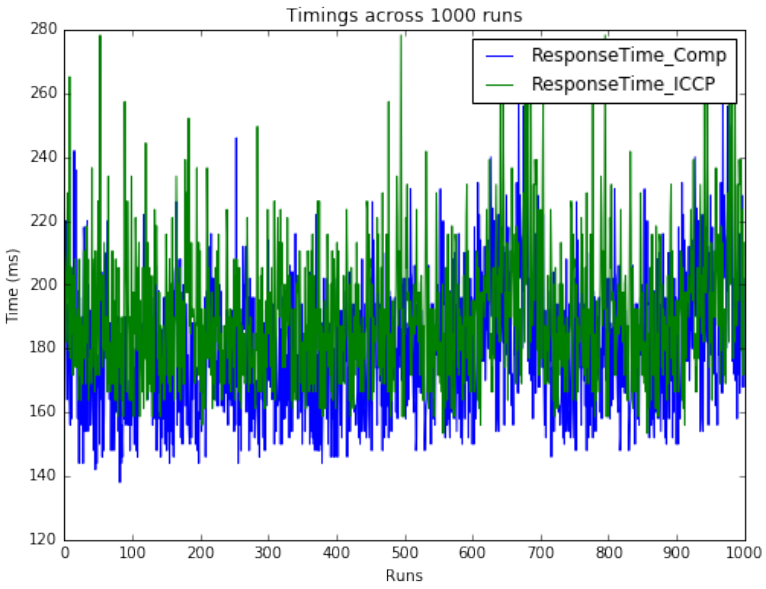


Figure 6.1: Response times for an Unseen Page

Cached Page

With a cached page, resources like images and JavaScript have been cached as the user has previously seen them. However, the personalisation of the page itself is deemed out of date, therefore personalisation will need to be performed at request time in both frameworks.

The user latencies for fetching a cached page are shown in Table 6.3. Comparing the load time for an unseen page to a cached page, shows the effect of client-side caching of page resources. Caching resources reduces the number of network requests made, thus reducing the average latency for both frameworks. For the content profile used, the network load is reduced by 11 requests from the 20 requests made for an unseen page. These are outlined in Table 4.2. Comparing the client-server framework and the ICCP, this effect is larger for the ICCP, the average response times reducing by 15.46% and 17.79% respectively. The ICCP has a further reduction in network requests for a cached page. There is no request sent to the content server as the page template and context are taken from the cache to be sent to the personalisation service.

If the personalisation of the page changes so that different, uncached resources are now included in the page, the ICCP could perform better than the client-server framework. As the ICCP loads context which it may not need during personalisation, the new resources may still be cached by the ICCP, but would not be cached in the client-server framework. However, in this research, a limited content profile was used, i.e. there were a limited number of resource choices available for personalisation and the personalisation rarely changed to include a new resource. Further research would be required to determine whether this effect would be significant with a larger content profile with more text, image and video samples available for use in building personalised content.

Again, jitter is seen in the response time as shown in Figure 6.2. The difference in the standard deviation for each framework is relatively consistent from an unseen page at 0.16. Again, this suggests service latencies or the latency in the client coordinator as the source of variability.

Metric	**Client-Server**	**ICCP**
Average Response Time (ms)	152.53	157.04
Std.Dev	21.51	21.35
Min/Max (ms)	115/209	127/225

Table 6.3: User Latency for Cached Page

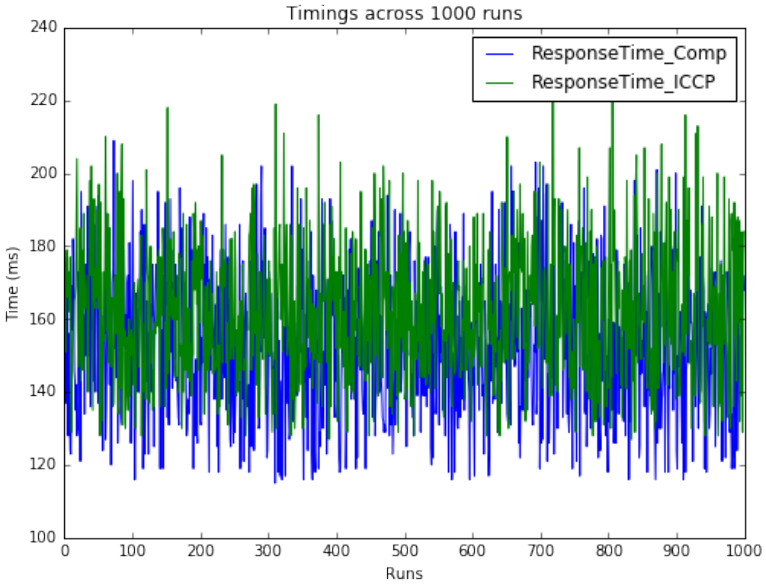


Figure 6.2: Response Times for a Cached Page

Prefetched Page

Finally, the latencies for a prefetched page are examined, these results are shown in Table 6.4, and in Figure 6.3. Comparing the frameworks, we see that the response time of the ICCP framework is significantly faster than the client-server framework when serving a prefetched page, with an average speedup of 79.65%.

This is due to the location of the prefetch cache as well as the rendering of the page. In the client-server framework, the prefetch cache lies on the content server and so a network request is required before a page is served, an additional overhead is then required to render the page. However, for the ICCP framework, the prefetched page exists on the client-device and no network requests are required when the page is requested. Thus, it is served pre-rendered with minimal latency. Requiring no network requests to serve a prefetched page also explains the very low standard deviation for the ICCP framework serving a prefetched page.

Metric	Client-Server	ICCP
Average Response Time (ms)	144.19	29.35
Std.Dev	20.37	5.9
Min/Max (ms)	109/216	19/40

Table 6.4: User Latency for Prefetched Page

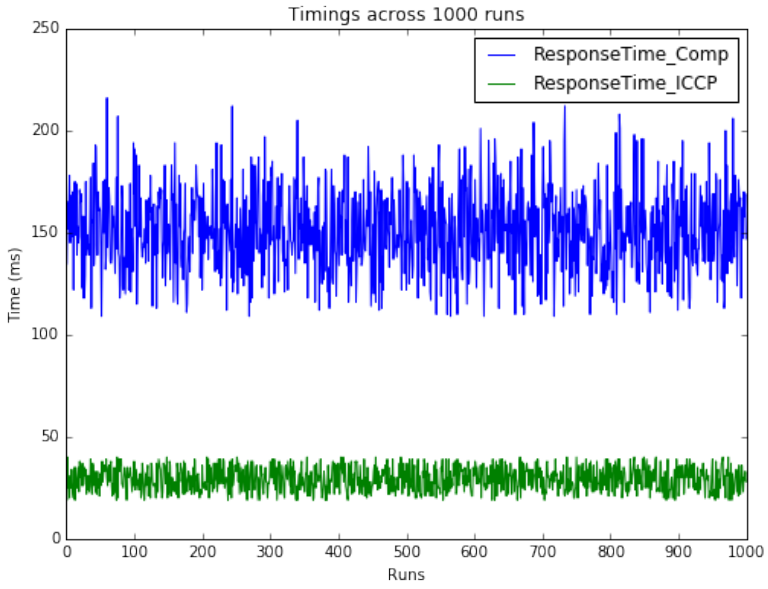


Figure 6.3: Response Times for a Prefetched Page

Comparing the latency for a prefetched page against an unseen or cached page in the client-server framework, a 13-20% speedup is observed. This is due to the reduced service processing time in the content server. During prefetching, the response has been prepared by the content server and placed in the prefetch cache so that a request can be quickly fulfilled.

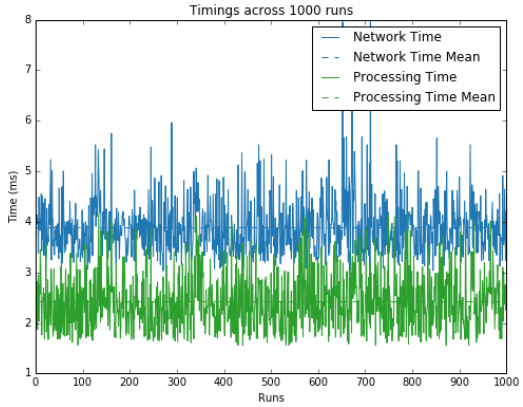
6.5.1.2 Service Latency

The next metric investigates the user latency in more detail, aiming to separate the contribution of network requests and service latency to the overall page response time. This section will establish the service latency benchmarks, i.e. the response times of each service with no additional load.

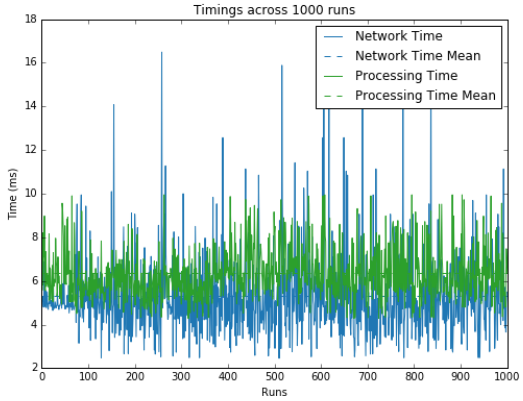
Unseen Page

For an unseen page, the ICCP makes a call to the content server and then to the personalisation service. The client-server framework makes a single call to the content server, which also performs the personalisation. Table 6.5 and Figure 6.5 summarise the average service latencies across 1000 calls when an unseen page is requested. Comparing the network time for the personalisation service and the client-server content server we see similar figures. However, the content server in the ICCP has a higher network time. This is because the network time incorporates the time it takes for the client to receive the full response, while in an ICCP the context object creates a larger response. Comparing the processing times of the services, shows that the content server in the client-server framework is approximately equivalent to that of the individual services in the ICCP. This is expected as the tasks being undertaken are almost the same. However, it confirms that the distribution of load across multiple services does not impact the processing time.

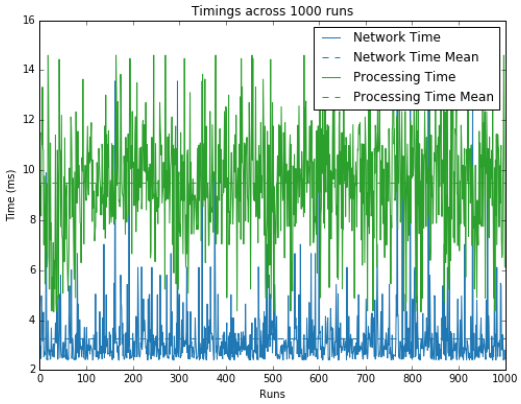
The variability in these values is further examined in figure 6.4 a, b and c. The network time for each service is relatively uniform with random peaks due again to the network jitter. While studying the user latency for an unseen page, a higher standard deviation was seen for the client-server framework against the ICCP. It was suggested that this may be due to variability in service processing times. Comparing the standard deviation in the content servers for each framework we see a difference of 0.25 which accounts for this change.



(a) Personalisation Service in ICCP



(b) Content Server in ICCP



(c) Content Server in Client-Server Framework

Figure 6.4: Service Latencies for an Unseen Page

	Network Time		***Processing Time***	
	Mean	Std.Dev.	Mean	Std.Dev
ICCP				
Personalisation	3.9	0.55	2.41	0.55
Content Server	5.37	1.46	6.32	1.1
Client-Server Framework				
Content Server	3.24	1.21	9.37	1.67

Table 6.5: Service Latencies for an Unseen Page

Cached Page

For a cached page, only a repersonalisation request needs to be made by the ICCP, thus there is no interaction with the content server. The service latencies for a cached page are shown in Table 6.6. As would be expected, these latencies are relatively unchanged from the unseen page.

	Network Time		***Processing Time***	
	Mean	Std.Dev.	Mean	Std.Dev
ICCP				
Personalisation	3.89	0.55	2.11	0.5
Client-Server Framework				
Content Server	3.24	1.24	9.36	1.67

Table 6.6: Service Latencies for an Cached Page

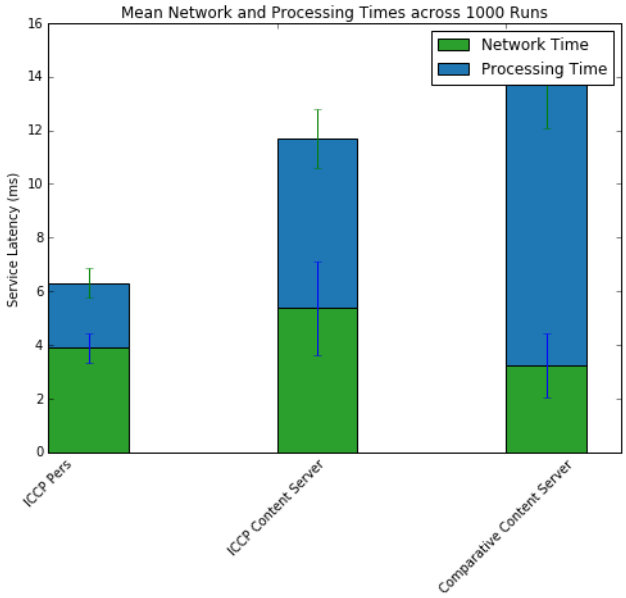


Figure 6.5: Service Latency Summary for an Unseen Page

Prefetched Page

In the ICCP framework, requesting a prefetched page involves no service latency. Thus, no call will be made to either the content server or the personalisation service. The content server response time in the client-server framework is shown in Table 6.7. Comparing this to the average response time for the other page types, there is a 74% reduction in processing time. This is the effect of the prefetch cache, as the content server immediately serves the page on request with no processing. The network time is unchanged as the amount of content being transferred is stable.

	Network Time		***Processing Time***	
	Mean	Std.Dev.	Mean	Std.Dev
Client-Server Framework				
Content Server	3.01	1.19	2.4	0.43

Table 6.7: Service Latencies for a Prefetched Page

6.5.1.3 Objective 1.1 Summary

The benchmarking objective aimed to evaluate the latency of each framework across the three different page types, with the goal of evaluating the effect of caching and prefetching on latency across the frameworks.

In the preceding sections, the ICCP and client-server framework latencies were examined for each page type. The distributed CSP approach with which we wish to compare the ICCP has the same architecture and thus the same latencies but involves only two page types as no prefetching exists.

Table 6.8 summarises the difference in user latency between the frameworks for each page type.

	Client-Server	ICCP	Distributed CSP
Unseen Page	180.42ms	+5.88%	+5.88%
Cached Page	152.53ms	+2.96%	+2.96%
Prefetched Page	144.19ms	-79.65%	N/A

Table 6.8: Comparison of Average User Latency between Frameworks When Benchmarking

For an unseen or cached page, the ICCP is slower than the client-server framework by an average of 5.88% and 2.96% respectively. However, when serving a prefetched page, there was a speedup of 79.65%. The majority of the user latency seen in the frameworks was due to rendering and fetching of resources. Investigating the contribution of service latency to the overall user latency, found that this represented roughly half of the slow down seen in the ICCP compared to the client-server framework. The additional load on the client-coordinator in the ICCP in dealing with these requests was responsible for the remainder. The ICCP is able to serve a prefetched page significantly faster than the client-server framework as the page already exists on the client-device and no network requests are required. When the page is requested it is served pre-rendered with minimal latency.

Comparing the frameworks, it is clear that the distribution of page types is essential to the overall latency in each framework. While the distributed CSP is always slower than a client-server approach, the addition of prefetching in the ICCP means that this framework can provide a lower overall latency given a high enough ratio of prefetched pages. Exactly how this can be achieved and the impact of the framework configurations is discussed in objective 2.1.

Objective 1.1 also addressed the effect of regular resource caching on user latency. It was found that the the average response times reduced by 15.46% for the client-server framework and and 17.79% for the ICCP framework when a cached page was requested.

6.5.2 Objective 1.2: Page Latency Streaming

After benchmarking the page response times, the latency during full system operation was evaluated. In this evaluation, background activities are operational, i.e. prefetching and prediction. Thus, the content server, the client coordinator and the microservices have increased load. Puppeteer is now used to stream the real user data collected during data gathering, as outlined in section 4.3.

This objective evaluates a realistic scenario for each framework and measures the latency impact of the background processes caused by user interaction. It also evaluates the increased load on each of the services, through measuring the traffic increase due to prefetching. Ultimately, with limited resources, this would affect the response time of the services. The objective does not utilise predictive metrics in the evaluation as this analysis is left to objective 2.1.

Finally, this objective compares these latency effects across the use cases, aiming to observe the effect of the user interaction profiles on the latency of the frameworks.

6.5.2.1 User Latency

When using streamed data, the process of serving a requested page is identical to that described in the benchmarking objective (1.1). The page caching and the quantity of network requests caused by the page request are the same. However, as the user has been interacting with the page, their behaviour has invoked additional processing by the client-coordinator. Predictions are made, updating the user model based on the user's behaviour, and as a result prefetching and personalisation are being performed in the background. This additional processing increases the load on all components and the network requests between them.

In the ICCP, the number of network requests caused by the background processes is predictable based on the value of each of the framework thresholds. To recap the summary in Chapter 3:

Activity Thresholds These control the number of calls made to the PPS, i.e. how often the user model is updated.

Prefetch Threshold This controls when a page qualifies for prefetching, and therefore how many pages are prefetched. Each prefetched page requires several network requests. Firstly, to the content server to fetch a page, secondly to the personalisation service and finally to fetch any page resources needed to render the page.

For the client-server framework, the processing is coordinated by the content server. The background processes are again controlled by thresholds, though in the case of prefetching the thresholds only increase the load on the content server itself. There is no activity threshold in the client-server framework as there is no client-coordinator and events are simply streamed straight to the content server for processing. Thus, prediction in a client-server framework causes a greater number of network requests than in an ICCP. On the other hand, both prefetching and caching have reduced network effect in comparison to ICCP.

However, if the client and the services were perfectly scalable, i.e. the response time was consistent regardless of how many requests were being processed, these background processes would not impact the user latency. In this case, only the latency due to the client-coordinator and the network requests would need to be considered.

	eCommerce		***Commercial***		***Informational***	
	Mean	Std.Dev.	Mean	Std.Dev	Mean	Std.Dev
ICCP						
Unseen	193.11	20.84	192.89	21.82	192.31	22.00
Cached	158.74	21.35	158.40	21.90	157.96	21.74
Prefetched	29.67	5.90	29.61	5.90	29.53	5.90
Client-Server						
Unseen	180.42	22.00	180.4	22.11	180.42	21.90
Cached	152.53	21.51	152.53	21.51	152.53	21.51
Prefetched	144.19	20.37	144.19	20.37	144.19	20.37
DCSP						
Unseen	192.03	21.79	191.90	21.93	191.76	20.96
Cached	157.80	21.47	157.81	21.70	157.66	21.34

Table 6.9: User Latencies for Streaming Evaluation

Table 6.9 shows the page load time distribution for each of the three case studies across the three page types.

For the ICCP, the use cases each show latencies very close to those seen during benchmarking in objective 1.1. However, there is a small latency increase for each use case from the benchmark value. Examining this across the page types shows that this is a consistent increase, meaning that the increase caused by the streaming methodology was not related to page type. As the latency of the client-server framework did not increase significantly for any page type, this suggests that again the overhead of the client coordinator adds to the user latency during streaming of real data. The average latency increase across these page types is summarised in Table 6.10.

	Average Latency Increase	
	ICCP	DCSP
E-Commerce	+1.12%	+0.5%
Commercial	+0.89%	+0.47%
Informational	+0.69%	+0.39%

Table 6.10: Increase in the latency from the benchmark when streaming real user data for the ICCP

Comparing the three use cases, we see that the e-commerce site has the largest latency increase. As this site also has the greatest number of interactions per page, as discussed in section 5.7, it seems that the ongoing background activities do impact the client-coordinator during a page request. Looking at the other two use cases, shows a relationship between the interaction profile and the latency of the client-coordinator.

6.5.2.2 Service Latency

For the service latency evaluation the average processing time and network time is examined across all websites. As the framework configurations for each are the same, the average load per service request will also be the same.

The average processing times for the service are shown in Table 6.11. These have stayed stable compared to objective 1.1. This shows that the number of requests sent to each service were within the scalability limits of each service, and thus did not affect the response time.

	Network Time		***Processing Time***	
	Mean	Std.Dev.	Mean	Std.Dev
ICCP				
Personalisation	3.95	0.57	2.34	0.5
Content Server	5.41	1.44	6.32	1.0
PPS	6.42	1.71	12.87	2.6
Client-Server Framework				
Content Server	3.17	1.20	6.08	1.09

Table 6.11: Average Service Latencies Across All Calls

6.5.2.3 Traffic Increase due to Prefetching

The previous section shows that during this evaluation, the services could handle the increased number of requests caused by prefetching. However, at scale this increase would cause a large overhead on the services, eventually slowing down the response time when the scalability limits of the service were reached. The decrease in user latency achieved through prefetching has an accompanied trade-off in increased service and network load.

This section examines the trade off for this evaluation by detailing the increase in service requests caused by the addition of prefetching. As this section aims only to examine the effect of prefetching, the ICCP is compared only to the distributed CSP framework without prefetching. This removes the effect of the framework design, isolating the effect of adding prefetching. The traffic increase due to prefetching for each service is discussed in the following sections.

Propensity Prediction Service

In this research, the PPS is used to make predictions which influence both the personalisation and the prefetching strategy. In reality, two separate prediction services would likely be employed to perform personalisation and prefetching. As the activity thresholds for both personalisation and prefetching were also identical, every call to the PPS served both purposes. Thus, the number of calls to the PPS arising from the process of prefetching can be taken to be every call. Therefore, the increase in load on the PPS due to prefetching is 100% as this service is utilised specifically for prefetching. The increase in network load depends upon the number of calls made to the PPS.

Website	# Calls	
	Mean	Std.Dev
Ecommerce	17.19	6.22
Informational	26.11	9.51
Commercial	47.0	15.34

Table 6.12: Average Number of Requests to the PPS Per User

The average number of calls made to the PPS by a user over a single session is summarised in Table 6.12. It shows that the load on the predictive service is hugely domain dependant, with the commercial site having more than double the number of calls than the E-commerce site. This load is entirely due to the specific interaction profile of users on a domain.

How this activity threshold can be determined to balance the load on the PPS for each specific domain is discussed in section 6.5.4.

Personalisation Service

The traffic increase for the personalisation service due to prefetching is caused by pages which are prefetched, and therefore personalised, but never requested i.e. prefetch misses. Thus, the additional load on the service depends on the predictive accuracy of the prefetching strategy and the prefetch threshold, as discussed in section 6.5.4.

For this research, the increase per threshold is shown in Table 6.13. This shows that when the prefetch threshold is 1, the load on the personalisation service is the same as when no prefetching is performed with a 1:1 ratio. When every page possible is prefetched for the use cases provided, the load on the personalisation service is more than tripled.

	Prefetch Threshold										
Site	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Ecommerce	3.27	2.41	1.76	1.47	1.43	1.37	1.41	1.36	1.33	1.25	1.0
Informational	3.29	2.51	1.97	1.72	1.55	1.61	1.55	1.55	1.46	1.23	1.0
Commercial	3.49	2.63	1.96	1.63	1.6	1.53	1.57	1.44	1.29	1.19	1.0

Table 6.13: Ratio of load on the personalisation service when prefetching vs not prefetching

Content Server

The traffic increase on the content server caused by the addition of prefetching can also be entirely accounted for by prefetch misses. Table 6.13 displays the traffic increase due to prefetching for the content server.

For the client-server framework, there is an additional load on the content server as the user interaction data is constantly streamed to the service for processing.

6.5.2.4 Objective 1.2 Summary

The objective of this evaluation context was to explore how prefetching pages and resources impacted framework performance.

Examining the user latency found that while streaming data, the response time of the client-server was unchanged from the benchmark. However, the ICCP framework displayed an increased latency for each use case, this was due to the overhead on the client-coordinator. As this latency was consistent for each page type, but varied between the use cases, the difference in the interaction profiles is the cause.

The evaluation objective also analysed the traffic increase seen by the services due to prefetching. This value was the same for both the content server and the personalisation service in the ICCP, and is dependant on the framework configurations. For the illustrative configurations used in this research, the additional load on these services was a maximum of three times that when no prefetching was employed.

6.5.3 Overall Analysis of Objective 1

Objective 1 aimed to investigate the latency of the frameworks across the different page types. The experiments were designed to investigate the impact of the framework differences, specifically the page caching strategy, the network load, and the service load created in each. While objective 1.1 gave benchmarking results, objective 1.2 tested the frameworks with real users and realistic settings.

The results showed that for an unseen and cached page the ICCP performed 5.88% slower than the client-server framework. This was due to the extra burden on the client device. For a prefetched page, the ICCP performed approximately 80% faster as the page had already been rendered and no network requests were made. Therefore in order for the ICCP framework to perform optimally, providing equivalent or better latency than the traditional client-server framework, the number of prefetched pages should be maximised, while the number of unseen and cached pages should be minimised.

Streaming real user data then showed that additional load on the client coordinator from background processes only slightly increased the response time. The amount of this increase is use case dependent. The network loads for the framework also increase as an additional microservice is employed to perform the user modelling for prefetching. Along with that, there is an increase seen in the network and service loads for the personalisation service and the content server. The increase depends upon the predictive accuracy of the prefetching strategy and the framework configuration.

Average Load time per user

These experiments allow the estimation of the average load time per user based on the response times for each page type. For each page type, multiplying the average number of pages of this type by the response time for that page, gives the average response time for a user. This assumes that that all services are perfectly scalable, i.e. the response times are stable and are not impacted by the number of simultaneous server requests.

$$\begin{aligned} Ave.ResponseTime \;=\; & (AvePrefetchedPageRT * \#PrefetchedPages) \\ & +(AveUnseenPageRT * \#UnseenPages) \\ & +(AveCachedPageRT * (\#RevisitedPages + \\ & +\#PrevPrefetchedPages)) \end{aligned}$$

Where RT is the response time of each page type

The number of pages prefetched is dependent on both the accuracy of the PPS as well as the prefetch threshold. Fetching multiple pages can make up for a poor predictive ability, however a balance must be struck. The client device has limited storage and the content-server has a scalability cap. If too many requests are received, the content-server will begin to perform poorly. This trade-off is explored in the second objective.

6.5.4 Objective 2.1: Effect of the Prefetching Configuration

The second evaluation objective, aims to investigate the impact of framework configurations on the latency. Specifically looking at the network and service loads in the frameworks. The effect of the prefetching configuration is evaluated and discussed. This configuration consists of two groups of thresholds, the activity thresholds and the prefetch threshold. Ultimately, the evaluation aims to discuss a method of optimising these threshold values as well as evaluate their impact in this research.

6.5.4.1 Prefetch Threshold

The prefetch threshold controls when a page is prefetched from the content server. The probability of being clicked that the PPS calculates for each page must reach this threshold before it will be prefetched.

In order to optimise this threshold, we need to examine the trade-offs between the number of pages prefetched and the prefetch hit ratio. If the prefetching strategy performed no prediction and simply prefetched every possible next page, the next page the user requests will always be a prefetch hit. Thus, the user latency would be hugely reduced, however this would cause considerably larger network and service loads. In addition, the cache space could also be filled very quickly, making this a completely infeasible solution. Hence, prediction is used to try and obtain cache hits while minimising the number of pages fetched. A poor prediction model or a difficult prediction can be compensated for through the use of a prefetch threshold, allowing more than one page to be prefetched. Selecting the prefetch threshold therefore depends on the predictive accuracy of the prefetching model for a website.

Figure 6.6 graphs these values for the training sets in this research. It shows the variation in threshold effect across the three different use cases. The e-commerce site has the highest accuracy as the number of interactions per page are greater. The graph shows that a prefetch threshold of 0.4 would achieve a 100% hit ratio. Whereas for the other two sites the ratio would have to be 0 and 0.2 to achieve the same hit ratio.

In selecting the prefetch threshold, the aim is to minimise the number of pages while maximising the hit ratio. This is therefore a ***multi objective optimization*** problem. Balancing the number of items cached and the hit ratio is a common problem in caching theory [134]. To be cost-effective and to enable efficient use of data, caches must be relatively small. Solutions seek to improve the efficacy of the overall system by reducing the data cost in terms of network utilization and processor demand. To find solutions to such a problem, weights must be assigned to the two variables, establishing the relative importance of each factor. The optimum threshold can then be determined using multi objective optimisation techniques such as Normal Boundary Intersection (NBI), Successive Pareto Optimization (SPO), Directed Search Domain (DSD) etc [135].

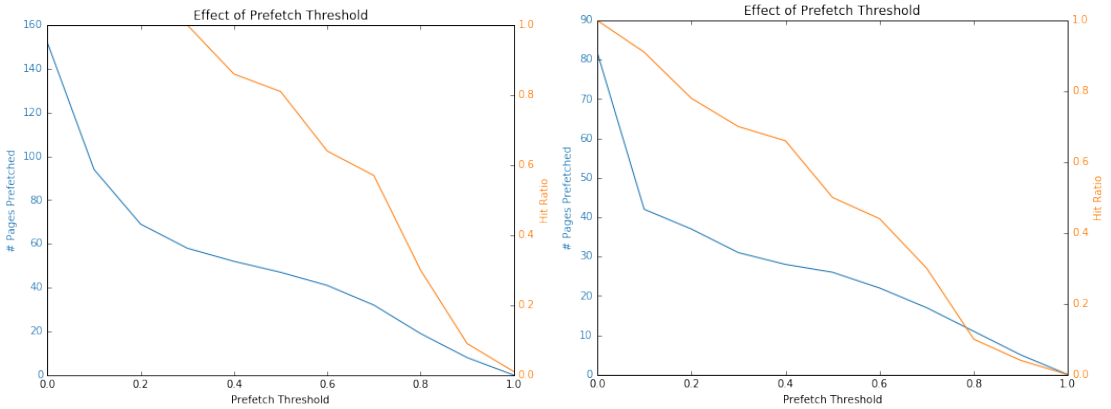
Determining the relative weights of the two variables, involves two further considerations:

- How much storage does the client device have?

- How scalable is the content server?

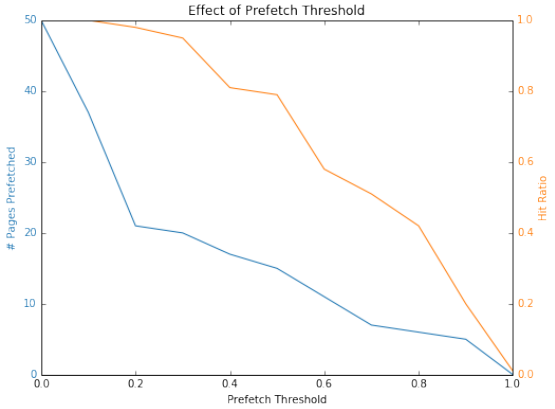
Each of these imposes constraints on the optimisation problem, introducing an upper limit on the number of pages that can be prefetched.

Setting a prefetch threshold is a use-case specific issue which is dependant on the limitations of the system, and the relative importance of the cache size against the hit ratio.



(a) E-Commerce

(b) Commercial



(c) Informational

Figure 6.6: Effect of the Prefetch Threshold: the number of pages prefetched vs prefetch hit ratio across website type

6.5.4.2 Activity Threshold

The activity thresholds for prefetching control how much a user interacts with a site before the user model is updated. The number of calls made to the PPS over a user session depends entirely on the activity threshold per event.

$$nPPSCalls = \sum_{e=0}^{N} \frac{nEvents^{(e)}}{activityThreshold^{(e)}} \tag{6.4}$$

where N = Number of event types

The activity thresholds should cause a call to the PPS only when the prefetching of pages will change based on the new predictions. That is, they aim to minimise the number of unnecessary calls to the PPS. Optimising this involves examining the frequency of prefetch requests against the frequency of prediction requests. Ideally, after a prediction is made, the result causes a new page to be prefetched.

In order to determine a value for this threshold, we need to consider:

- How often the propensity prediction changes based on the user interaction events
- What impact this change has on the prefetching
- How scalable the PPS is

Table 6.14 shows the proportion of PPS calls which resulted in a prefetch for each of the sites.

Again, the scalability of the PPS would enforce an upper bound on the activity threshold. Too many requests would slow down the service processing time in dealing with requests.

Given a fixed prefetch threshold, an appropriate activity threshold could be selected by taking the average number of each event occurring between PPS calls resulting in a prefetch. However, there may be many factors contributing to whether a call results in a prefetch besides the absolute number of events. For example, the features used in the

	Prefetch Threshold										
Site	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Ecommerce	56	68	64	61	56	55	50	44	32	17	0
Informational	56	71	66	60	59	56	51	45	31	15	0
Commercial	57	68	66	62	59	56	51	43	31	17	0

Table 6.14: Percentage of PPS Calls resulting in at least one prefetch

prediction, which feature values changed, and how much they changed by. Depending on the complexity of the system, dynamic thresholds could provide the biggest benefit. Determining these would be well modelled as another machine learning task, open for future research.

6.6 Chapter Summary

This chapter aimed to evaluate the performance of the ICCP framework against a comparative, client-server framework and a Distributed CSP without prefetching. Specifically, investigating the impact of framework choice on webpage response time, looking at:

1. The system latencies of the framework across different page types
2. The effect of framework configurations on the service loads

The first experiment benchmarked the response times for each of the page types. This found that for an unseen page and a cached page, the ICCP framework response time was 5.88% and 2.96% slower than that of the client-server framework. This increase was due to the overhead of the client coordinator. However, for a prefetched page, the ICCP provided a 79.65% increase in speed over the client-server framework. The average load

time per user seen in an ICCP framework is always lower than the distributed CSP without prefetching. It is also lower than the client-server framework when a high hit ratio can be achieved, maximising the number of page requests which serve a prefetched page.

The trade-off in network and service load against this hit ratio must also be considered when comparing the client-server framework and the ICCP. The exact profile of this trade-off is use case dependant and thus, when determining the applicability of the ICCP an analysis of the use case must be performed. The metrics and evaluation outlined in this chapter can be followed to determine the trade-off profile between the client-server framework and the ICCP for a specific use case.

Chapter 7

Propensity Prediction Evaluation

Evaluating the Propensity Prediction Service (PPS) aims to examine how well a user's next page request can be predicted by their interactions. The previous chapter investigated the prefetch hit ratio from the results of the PPS. However, this metric was dependent on the configuration of the framework. Thus, it explored the effect of different predictive accuracies and the absolute accuracy achieved by the PPS did not impact the evaluation.

This chapter examines metrics for evaluating the PPS predictions independently from the framework configuration. The investigation of the ability for mouse movement to predict a user's navigation in a stereotyped model forms a minor contribution of this research. The factors influencing the predictive accuracy in this research are also discussed.

7.1 Prediction Data

The data gathered during this research was outlined in Chapter 5. This was collected by recruiting participants through Prolific to interact with three website use cases. The raw data collected is recapped in Table 7.1.

Event Category	Types of Event
User Info	SystemInfo, LocationData, ScreenData
Session Data	NewUser, NewSession
Interaction Data	MouseMove, MouseLeave, Click, RightClick, Scroll, Typing, Copy, Paste

Table 7.1: Events tracked during data gathering

Thus, three separate data sets were established which were used in the simulation experiments in the previous chapter as well as the predictions outlined in this chapter.

7.2 Predictive Models

This research, examined a hybrid approach to building user models as discussed in Section 3.6.2. A stereotyped model was first established offline to be used for new users to the system. These models were then retrained as user-centric data was processed during the data streaming. This hybrid strategy enabled both user modelling approaches to be explored in this research.

7.2.1 User Centric Model

The ability to train user-centric models was an important consideration in the design of the framework and the system latency evaluation. However, due to the limited volume of data available, evaluating the predictive power of these user centric models was inconclusive. The results can be seen in table 7.2, there is very little loss reduction ob-

served compared with the stereotyped model and in some cases it has actually worsened the model. The predictions made by these models did not improve over the stereotyped model significantly as they did not have sufficient train time. However, incorporating the user-centric model in the framework evaluation was important to allow the performance impact to be evaluated.

Model	E-Commerce	Informational	Commercial
Multinomial NB	+0	-0.01	-0.03
Bernoulli NB	+0.02	-0.02	+0.01
Perceptron	+0	-0.02	-0.09
SGB Classifier	+0.01	-0.02	-0.07
Passive Aggressive Classifier	-0.05	-0.03	-0.07
MLP Classifier	-0.03	-0.01	-0.14

Table 7.2: Change in Log Loss Observed with the User-Centric Model

The evaluation of these user centric models based on mouse dynamic data offers an interesting avenue for future research. Particularly in examining the effect on predictive accuracy of the incremental training necessary to protect user data.

The remainder of this chapter discusses the offline training and predictive power of the stereotyped model only.

7.3 Data Pre-processing

The predictions made by the PPS are based on raw user interaction data like mouse movements and scrolling. In order to make predictions over this data some pre-processing must first be performed. Specifically, this section looks at strategies for dealing with missing data, normalising the data and dividing it between the training and test sets.

7.3.1 Missing Data

Coping with missing data is a common step in data pre-processing. Strategies need to be carefully considered, to avoid the introduction of bias into the model. There are a variety of techniques used in the research to replace missing data values; parameter estimation using statistics over the dataset, imputation to replace data with plausible values or simply discarding erroneous data [109]. To reduce the likelihood of missing data occurring, validation and verification of the data was performed during data gathering, this is outlined in chapter 5. Performing in person monitoring and video interviews minimised the risk of user events occurring which were not tracked by the system. Using a simulation program to then replay actions and compare them to screen recordings verified the veracity of the data that was collected.

Performing data exploration during this research shows that missing data is not a significant issue. Only two cases could be identified which produced 'null' values in the data. Firstly, in typing event records, the character code for the symbol typed was not reliably present. However, this is redundant information and as such it's omission was inconsequential. The second case where missing data was encountered was in measuring click duration for a right click made by the user. The cause of this missing data is not known, however, it represents only 2% of right clicks, which are themselves rare in the data. A total of 5 out of 212 right clicks recorded in the data (both in test and training data) were missing duration information. Furthermore, examining the correlation matrix suggests that right click data is not a large contributing feature in the model. Thus, a decision is made to discard the missing data in this instance i.e. Any right click events with missing click duration were discarded from the data set.

7.3.2 Data Normalisation

Data Normalisation is applied to the raw data collected, to ensure all features share a common scale and range. This prevents the model from over-weighting features with large scales, for example the length of a mouse click may be 100ms whereas the number of clicks made would be 1.

7.3.3 Train Test Split for Stereotyped Model

To produce a stereotyped model for each of the three website case studies, the collected data had to be split into a training and a test set. This model was batch trained as it was not subjected to the same privacy concerns as a user-centric model which must be trained incrementally.

Each user's data occurred either in the training set or the test set but never in both. The data was divided on a **per user basis**. This meant that when evaluating the predictions with the test set, the user was completely unknown.

An iterative process was followed to establish the proportion of users to assign to each set, ultimately establishing a 0.6/0.4 train test split. This means that 60% of users were assigned to the training set and 40% to the test set. This resulted in both good predictive accuracy and sufficient test data for performance testing. The number of users in the test and training set for each website use case is shown in Table 7.3. The results from the evaluation of these stereotyped models are presented later in this chapter (Section 7.7).

	Test Set	Training Set
E-Commerce	182	122
Commercial	52	35
Informational	55	36

Table 7.3: Number of Users in the training and test sets

7.3.4 Imbalanced Data

Assigning users to either the training or test set can be achieved through random sampling. However, with classification problems this can lead to issues caused by **imbalanced classes**, as discussed in section 3.6.3. Most machine learning algorithms work best when the number of samples in each class are about equal. Unbalanced class distribution across the sets can result in very poor predictive ability. In this research, the class, i.e. what we are trying to predict, is the URL which the user next visits. At a minimum, each class, or visited URL, must exist in the training data.

If some classes are underrepresented in the data, random sampling may cause these to be highly unlikely to be predicted. Stratified sampling is used to yield a different target class distribution which is more attuned to the minority classes. For cases in which the distribution of classes is close to uniform, stratified sampling is equivalent to random sampling. Exploratory analysis showed that there was in fact a class imbalance issue in the data. Some webpages were visited by almost every user, whereas others may have only be visited by one. Thus, stratified sampling was used to distribute the users between test and training set. This ensures that each fold in the cross-validation was representative of the whole.

7.4 Prediction Granularity

The granularity of the prediction being made must be defined i.e. when should a URL be classified as unique? For example, is a URL different due to its query parameters? The URL classes were determined manually through visual inspection aiming to determine when the objective of the user was significantly different. In establishing these URL classes three rules were set out:

1. All query parameters were excluded from the URL

2. The URL was stripped of terms to find the earliest 'grouping subdomain'. This was a subdomain which always produced the same page visually, differing only in context.

3. The root of a grouping subdomain is considered its own class

For example, for DiscoverIreland a grouping subdomain was '/destinations/'. The pages *'https://www.discoverireland.com/destinations/republic-of-ireland/dublin/dublin-city/'* and *'https://www.discoverireland.com/destinations/northern-ireland/county-antrim/belfast/'* are identical visually but the exact text and images for each are different. According to rule three *'https://www.discoverireland.com/destinations/'* is a separate class to that above as it is visually different.

The number of unique URLs in the data based on this classification are shown in table 7.4. For the E-commerce and commercial sites, these numbers are relatively low as the classification granularity is high. For example, the PPS did not attempt to predict exactly which product page a user would view but only that they would next view a product page. However, for the informational site, there existed no obvious page groupings, and thus a much lower classification granularity was applied. This allowed the effect of classification granularity on the prediction to be observed. Increasing the granularity of the prediction while maintaining predictive accuracy would require some contextual information to be introduced.

Website	No. Classes
E-Commerce	17
Informational	47
Commercial	11

Table 7.4: Number of Unique Classes per Use Case

7.5 Predictive Features

The PPS derives predictive features from the raw user interaction data, outlined in section 5.3. Manual feature engineering was used on this raw data to explore features which may be predictive of the next URL. The features tested in the system for predictive accuracy were influenced by the state of the art, described in section 2.3.

The features used in the final iteration of the PPS are as follows:

- Current Page
- Mouse Position i.e. Distance from links
- Features of Stroke [136, 137]
  - Angle of Move
  - Speed
  - Degree of backtrack
  - Steadiness – How far from a straight line
  - Vertical Range
- Speed of Click
- Average Time between Actions
- Action Prediction [138]
  - Reading: Horizontal Move
  - Skipping: Fast Vertical Move & Hesitating
- Boolean Action Occurrence
  - Double Click
  - Right-Click
  - Keyboard Event

The current page the user is on is a significant factor in prediction as it reduces the solution space significantly. There are a limited number of next pages available to navigate to on each individual page.

7.6 Metrics

In evaluating classifiers, evaluation metrics can be categorised into three groups, threshold metrics, ranking metrics and probability metrics [139]. Each of these metrics have different goals in evaluating the classifier.

Threshold metrics include accuracy, the most used evaluation metric, as well as precision, recall, sensitivity and specificity. While these metrics are easy to both compute and understand they also produce less distinctive and less discriminable values. The metrics consider only the class or classes that were assigned by the model i.e. the class(es) with a probability over the designated threshold. In the context of this research, these metrics results would be dependent on the prefetch threshold as they would only evaluate the prediction based upon which pages were prefetched by the system, ignoring the probability assigned to each page. For this evaluation we aim to evaluate these probability values, making this category of metrics unsuitable.

Ranking metrics, for example the area under the ROC curve, are independent of the threshold. These evaluate the model based on the order of the results rather than the absolute probability assigned to each. While these are more useful in the context of this research the absolute probability is more informative than the ranking as there are not a fixed number of pages being fetched. Thus, a probability metric is used in this research, this evaluates the the accuracy of probabilistic predictions accounting for the exact probability assigned to each class, it is also known as a scoring rule.

There are two metrics commonly used as scoring rules to evaluate probabilistic predictors [140]: The Brier score and the log loss. These are outlined in the following sections.

7.6.1 Brier Score

A common method for evaluating a probabilistic classifier is using the **Brier Score**. For each output class, it measures the mean squared difference between the predicted probability, f_t, and the actual outcome o_t:

$$BS = \frac{1}{N}\sum_{t=1}^{N}\sum_{i=1}^{R}(f_t - o_t)^2 \tag{7.1}$$

Where, R is the number of possible classes and N is the number of classifications A lower Brier score indicates a better classifier.

7.6.2 Log Loss

Log loss measures how much the predicted probabilities differ from the actual label. A larger gap between these values leads to a higher log loss. A perfectly predictive model would have a log loss of 0. The log loss is defined as follows:

$$LogLoss = -\frac{1}{N}\sum_{i=1}^{N}(y_i).log(p(y_i)) + (1 - y_i).log(1 - p(y_i)) \tag{7.2}$$

The log loss score heavily penalises predicted probabilities far away from their expected value. The Brier score is gentler than log loss but still penalises proportional to the distance from the expected value. Like the average log loss, the average Brier score will present optimistic scores on an imbalanced dataset, rewarding small prediction values that reduce error on the majority class.

7.6.3 Conclusion

For the evaluation of the PPS, the log loss is used as the predicted probabilities far away from the true values represent an additional load on the framework and thus should be heavily penalised.

7.7 Stereotyped Model Results

To contextualise these results we compare them to a naive baseline model. Typically, in evaluating a probabilistic predictor, the intercept-only model is used for comparison. This is a naive model which has no independent variables, thus, it simply predicts the average value for the sample. In the case of classification, this means always predicting the majority class i.e. always predicting that the most frequently visited URL will be visited next. The Power of the intercept-only model is 0 percent. However, in the context of this research the intercept-only model represents a poor baseline. The most frequently visited URL across the dataset may not be available to be navigated to from the current page. Given the low proportion of revisited pages, this page is also likely to be correct for only one of the predictions made per user. Instead, a model predicting the next url based on the frequency of visited URLs in the training data *from the current page* was used as a baseline. Another advantage of using this baseline is that the power of the 'current page' feature is normalised across the models, thus the mouse dynamic data can be evaluated in isolation.

For each website, the best performing model was automatically chosen through cross validation over the training set. Theses results are presented in Table 7.5. In each case, the MLP Classifier produced the highest accuracy. The log loss results for each website are shown with the baseline in Table 7.6.

These results show that in each case a small gain was made over the baseline using mouse dynamic data. The increase was largest for the E-Commerce site, likely due to the larger quantity of available data. For the informational site, the increase achieved over the baseline was smaller, this was due to the higher granularity of the prediction being made. In determining the significance of probabilistic classification results the literature [141] concludes that a paired t-test should be used. The results for this are also shown in Table 7.6, indicating the p-value for each website. For both the e-commerce and commercial sites the results can be considered statistically significant. However, for the informational site the results do not indicate any significance.

Model	E-Commerce	Informational	Commercial
Multinomial NB	0.46	0.77	0.73
Bernoulli NB	0.51	0.8	0.73
Perceptron	0.42	0.74	0.59
SGB Classifier	0.42	0.75	0.62
Passive Aggressive Classifier	0.42	0.75	0.64
MLP Classifier	0.3	0.71	0.48

Table 7.5: Comparison of Model's Average Log Loss

	Baseline	**PPS Stereotyped Model**	
Website	Mean log loss	Mean log loss	p-value
E-Commerce	0.57	0.3	0.0047
Informational	0.77	0.71	0.5292
Commercial	0.71	0.48	0.0159

Table 7.6: Average Log Loss for the Stereotyped Predictions Across Website Use Cases

7.8 Chapter Summary

This section aimed to evaluate the predictive ability of the PPS based only on mouse dynamic interaction data. The results for the evaluation of the stereotypical model show that with a high class granularity, predictions above a baseline model can be achieved. However, in cases where there is a low class granularity, the predictions do not achieve a significant improvement over this baseline. High class granularity can be seen in websites where some contextual grouping of pages exists, for example, on an estate agent website you may group pages which describe three bedroom houses in a specific postcode.

This evaluation shows that mouse dynamic interaction data, without any contextual information, holds predictive ability about the user's navigational intentions. As discussed in Chapter 2, no current literature uses mouse dynamic interaction data to predict a user's navigation. Using this interaction data to augment current predictive techniques utilising contextual information provides an interesting avenue for future research.

The prediction strategy for this research was deliberately chosen to evaluate the framework under a heavy load. Unlike data such as the user's interests or browsing history, interaction data provides a near constant stream of information to a prediction system. As the user browses a website they produce a wealth of data by clicking, scrolling, typing and moving their mouse. In addition, the research employed machine learning techniques which created more complex user model training. Utilising data in the ICCP with a smaller footprint or using a less complex predictive model would result in reduced service and network loads than those seen in chapter 6. This would further improve the ICCP performance over the traditional client-server framework.

In addition, the latency of all frameworks can be improved through the framework configurations, namely the activity and prefetch thresholds discussed in chapter 6. A poor prediction model or a difficult prediction can be compensated for by lowering the prefetch threshold, allowing more pages to be prefetched. This has to be balanced against the additional service and networks loads it would result in as well as the cache memory usage to store the quantity of prefetched pages.

The absolute accuracy achieved from the PPS did not impact the evaluation in chapter 6 giving this research the flexibility to form a minor contribution illustrating the ability of mouse dynamic data to predict which webpage the user will request next.

Chapter 8

Conclusion

This chapter summarises the thesis and its achievements. It discusses the trade-offs encountered in the design of the ICCP framework, discusses the results of the comparative performance evaluation of the framework and highlights potential areas for future work.

8.1 Thesis Summary

The question which this thesis sought to answer was:

> *"What are the design challenges, privacy and performance trade-offs for a privacy-conscious web architecture using a distributed client-side personalisation approach augmented with predictive prefetching?"*

The design challenges and privacy trade-offs of an Intelligent Client-Centric Personalisation framework were examined throughout Chapters 2 and 3. Four state of the art systems were contrasted and the trade off of different design approaches discussed. Ultimately, the design for an ICCP framework was proposed. This theorised that incorporating predictive prefetching with a distributed CSP architecture, could realise the privacy offered by a client-side personalisation approach while enhancing the performance of existing techniques.

A quantitative evaluation of the ICCP framework's performance was detailed in Chapter 6 and 7. Specifically, focusing on the user latency of the ICCP in comparison with a distributed CSP with no prefetching as well as with a traditional client-server framework. The evaluation addressed both the effect of the prefetching strategy on user latency as well as the effect of the framework configurations. It was found that, with sufficient prefetching, the ICCP framework could perform better than the traditional client-server model. However, the trade-off for this performance was an increase in network and service load on both the content server and the microservices. The chapter concluded by motivating the use cases for appropriate application of an ICCP framework.

8.2 Contributions

8.2.1 Major Contribution

The major contribution from this thesis to the current research in client-side personalisation [11, 13, 52, 62] is:

- The novel introduction of prefetching into a Distributed CSP architecture and the comparative evaluation and analysis of the effects of this prefetching on user latency. The contribution also includes the comparative analysis of DCSP with prefetching to the traditional client-server framework.

The ICCP framework has the potential to significantly improve the privacy of personal data for individual users, while allowing website providers to offer personalised services.

Personalisation is used to improve the services of an increasing number of websites. It has been shown to improve user engagement, satisfaction, and increase sales. Some of the biggest web companies have adopted personalisation and now consider it essential in increasingly competitive markets.

However, there is also increased awareness of the amount of personal data which many companies have on individual customers. This awareness has been slowly building, especially due to recent unauthorised data breaches and social media scandals. There is also increasing awareness of the value and private nature of the less visible data consumers leave on non social media websites, such as their search history, browsing habits, online shopping habits, and advertisement clicks.

For websites to personalise their services to users, they need personal data. Users report having increased usefulness satisfaction, and engagement from personalised platforms. Yet, they are also more aware of the dangers of sharing their personal data. This has become known as the personalisation-privacy paradox.

Existing privacy conscious frameworks such as Client-Side Personalisation have several disadvantages. For example, IP concerns (where proprietary personalisation and ML algorithms have to run on the client device), limited inference data, large data overheads, restrictive client processing power and an inability to easily sync user profiles across devices. While Distributed CSP went some way in addressing these issues, they have issues with scalability and performance.

The ICCP framework builds on these existing frameworks, especially Distributed CSP, and addresses this issue of performance. Due to its microservice design, which employs trusted third party services and maintains storage of personal data on the user's device, it is a responsive, lightweight, and privacy conscious framework.

The effect on the domain is twofold. Firstly, the ICCP framework can help to alleviate key privacy concerns among users of personalised websites and services, consistently outperforming the distributed CSP framework with no prefetching. Secondly, the ICCP framework allows companies to continue offering personalised services to better serve the needs of their customers.

8.2.2 Minor Contributions

There are two minor contributions from this research to the state of the art.

- A novel combination of evaluation techniques and metric specification which focus on user latency across multiple frameworks
- The examination of the ability of mouse dynamic data to predict which webpage a user will request next and its improvement over a comparative baseline

8.2.2.1 A novel combination of evaluation techniques and metric specification which focus on user latency across multiple frameworks

The evaluation approach used in this research formed a novel technique for assessing the latency of a web framework. The combination of techniques allowed for a framework to be compared to existing systems under realistic user loads, isolating the impact of user actions on this latency. This involved the *collection* of real user data and the *streaming* of this data through multiple frameworks to *compare* the latency of each.

A dataset reflecting user interactions with online websites was required to test the frameworks under realistic loads. Constructing a bespoke website specifically for research purposes would not have realistically reflected website complexity, thus data from a real website was required. Existing methods for collection of user data in research include web scraping and utilising plugins. Scraping a website, involves copying all of the resources from a site and recreating it. However, this method rarely produces a functional website due to the complexity of the underlying code. Requiring participants to install a plugin, is more frequently used. This injects code into specific websites when vis-

ited by the participant. However, there are several issues with this, firstly it creates an additional burden on the participant, increasing the probability of user error, the user must also uninstall the plugin after use. More importantly, injected code will run after the site has rendered, thus causing issues with timing. For example, any actions taken by the user before the site has fully rendered will not be recorded. Additionally, this limits the ability to alter the site before presenting it to the participant, for example, by removing links to external websites. During this research, both of these methods were tested before establishing the proxying method ultimately utilised. This was technically challenging due to the various security measures implemented in modern browsers. The proxying method used in this research allows the participant to simply click a URL where an altered version of a live site will be immediately rendered and realistic data can be tracked. This method cannot be found in the state of the art and collection of research data through a reverse proxy forms a novel contribution to the literature.

To more rigorously evaluate the frameworks, several use cases were employed. This allowed the framework to be evaluated across different user interaction profiles so that the results were not website specific. However, if these websites were then used in evaluating the frameworks, the effect of the interaction profiles on latency could not be evaluated as the content of each website would introduce variation in the network and service loads. To mitigate this effect, the content profile was standardised through a dummy website. The data was mapped to this site and streamed through each framework. Performing this offline, standardised evaluation meant that the effect of interaction behaviour was isolated, several frameworks could be compared and the experiments could be repeated.

Finally the metrics for this comparative evaluation were outlined, namely, user latency, service latency, latency per page ratio and traffic increase. The latter two metrics allow frameworks to be compared focusing specifically on the effect of a single architectural change.

This combination of evaluation techniques allowed the latency of a framework to be compared to existing systems using realistic user data while isolating the effect of user behaviour on latency. This is a technique which could be applied in any research wishing to compare web framework latencies under realistic settings.

8.2.2.2 The examination of the ability of mouse dynamic data to predict which webpage a user will request next and its improvement over a comparative baseline

The second minor contribution from this thesis was an examination of the ability of mouse dynamic behaviour to predict user navigation. In the course of this pursuit, a significant gap in the current body of knowledge was identified. Although mouse dynamic information is used in search optimisation and advertising, the majority of personalisation research looks only at click history. Currently no work exists which attempts to predict user clicks based solely on mouse dynamics, see section 2.3.

This thesis found that mouse dynamic behaviour could be utilised to predict user navigation patterns in use cases with high granularity classes. Although in isolation, these gave only a small increase over the baseline, they could be integrated with existing prediction techniques to improve accuracy.

The identification of this novel and unexplored area for personalisation forms a minor contribution of this thesis.

8.3 Research Limitations

This thesis forms a proof of concept for the applicability and benefits of the ICCP framework. However, there are many questions around the research still to be explored and answered before it is viable for industry use.

Some of the limitations to the conclusions drawn from the evaluation are addressed in section 6.4. While every effort was made to elicit reliable data from the users, realistic behaviour is impossible to reproduce when the user goals are prompted. This research would be very strongly improved with real industry data.

The evaluation of the system is also limited, a single machine profile is used to test the framework. In addition only one browser and one operating system are involved in the evaluation. Before deploying an ICCP framework more expansive evaluation would be required.

For the ICCP framework to be useful in practice, a prediction service must exist which can predict the user's next URL quickly and accurately enough to reduce page load latency to that of a traditional content-server approach. This thesis simply posits that such a service exists without explicitly building it.

The next section touches on future work that could be done to address some further limitations of this research.

8.4 Future Work

Many of the ideas and research challenges surrounding this thesis could be addressed in more depth through future work. Included below are suggestions for this future work, discussing the next steps required to explore these.

8.4.1 Activity Thresholds

For this research, fixed activity thresholds were used for both prefetching and personalisation. The activity thresholds control how often the user model should be updated based on how much a user has interacted with a page. If the user model is updated and there is no resulting change in the personalisation or prefetching then an extra load

on the framework has been introduced to no benefit. Ideally, the models would only be updated when sufficient activity has occurred to impact the outcome. Determining the optimum value for these activity thresholds would prove an interesting area for future work.

8.4.2 User Behaviour Models

As discussed in chapter 4, section 4.3, there were no existing behaviour models of user interactions that provided sufficient information to be utilised in this research.

To protect the privacy of the participants involved in the data gathering for this research, an open dataset could not be released.

However, aggregate statistics over this data, like those outlined in section 5.7, could provide a useful resource for future research. These would enable other researchers to create accurate simulation models of user behaviour. Additional privacy measures could be applied to the data in producing these aggregate statistics. For example, differential privacy or homomorphic encryption (Discussed in Section 2.1.2.1).

8.4.3 User Centric Models

The evaluation of the predictions made in this research looked only at a stereotyped, batch trained model. The low volume of data available meant that the user centric models did not provide a substantial gain over this baseline. However, with more data research could be performed on user centric models, examining two facets: (i) the predictive ability of mouse dynamic data in user centric models and (ii) the effect of incremental training on the predictive ability of such a model.

8.4.4 Latency Impact of Refreshing Personalisation

Section 3.5.2 introduces the refresh threshold in the ICCP framework. This allows the personalisation of a page to be updated in the background while the user is interacting. Refreshing is valuable in situations where the user's recent interactions inform the prediction made. For example, when:

- Classifying in-page actions
- Training a model for cold-start users
- Training rapidly changing User Models

Refreshing this personalisation would require additional calls to the personalisation service, increasing the network and service loads. An interesting area of research would be in establishing (i) the value of refreshed personalisation and (ii) the impact of this on the page latency.

8.4.5 Mouse Dynamics for Personalisation

The ability for mouse dynamic information to predict user navigation was established in this thesis. A promising avenue for future research looks at integrating this data with existing predictive strategies, for example, based on the user click history, their interests or the context of the page. This could augment existing techniques to give gains in predictive accuracy.

Bibliography

[1] Sarabjot Singh Anand and Bamshad Mobasher. Intelligent Techniques for Web Personalization. pages 1–36. 2005.

[2] Alfred Kobsa, Bart P. Knijnenburg, and Benjamin Livshits. Let's do it at my place instead? *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI '14*, pages 81–90, 2014.

[3] Eduardo B Andrade, Velitchka Kaltcheva, and Barton Weitz. Self-Disclosure on the Web: The Impact of Privacy Policy, Reward, and Company Reputation. *Advances in Consumer Research*, 29(1):350–353, 2002.

[4] Scott Lyon and Ronald Camhi. What Is the California Consumer Privacy Act ? *Risk Management*, (October 2018), 2018.

[5] California Act. California Consumer Privacy Act, 2018.

[6] CIPP/US Emily Leach, CIPP/E. 2019 Global Legislative Predictions. *International Association of Privacy Professsionals iapp.org*, 2018.

[7] Chung Hun Lee and David A. Cranage. Personalisation-privacy paradox: The effects of personalisation and privacy assurance on customer responses to travel Web sites. *Tourism Management*, 32(5):987–994, 2011.

[8] Naveen Farag Awad and M Krishnan. The Personalization Privacy Paradox: An Empirical Evaluation of Information Transparency and the Willingness to be Profiled Online for Personalization. *MIS Quarterly*, 30:13–28, 2006.

[9] Avi Goldfarb and Catherine Tucker. Shifts in privacy concerns. *American Economic Review*, 102(3):349–353, 2012.

[10] Lillian Cassel and Ursula Wolz. Client side personalization. *DELOS-NSF workshop on personalization and recommender systems in digital libraries*, page 57, 2001.

[11] Anupriya Ankolekar and Denny Vrandečić. Kalpana - enabling client-side web personalization. *Proceedings of the nineteenth ACM conference on Hypertext and hypermedia - HT '08*, page 21, 2008.

[12] Eran Toch, Yang Wang, and Lorrie Faith Cranor. Personalization and privacy: A survey of privacy risks and remedies in personalization-based systems. *User Modeling and User-Adapted Interaction*, 22(1-2):203–220, 2012.

[13] S. Gerber, M. Fry, J. Kay, B. Kummerfeld, G. Pink, and R. Wasinger. PersonisJ: Mobile, client-side user modelling. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6075 LNCS:111–122, 2010.

[14] Josep Domènech, Ana Pont-Sanjuán, Julio Sahuquillo, and José A. Gil. Evaluation, analysis and adaptation of web prefetching techniques in current web. *Advanced Information and Knowledge Processing*, 46(March):239–271, 2015.

[15] Utpal Acharjee. Personalized and artificial intelligence web caching and prefetching.pdf, 2006.

[16] Gartner Marketing. Top Marketing Predictions for 2020. 2019.

[17] Adeneye Olarewaju Awofala. Effect of Personalisation of Instruction on Students' Motivation to learn Mathematics Word Problems in Nigeria. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, 7(3):486–486, 2016.

[18] Johannes N. Blumenberg and Manuela S. Blumenberg. The Kretschmann Effect: Personalisation and the March 2016 Länder Elections. *German Politics*, 27(3):359–379, 2018.

[19] Ramnath K. Chellappa and Raymond G. Sin. Personalization versus Privacy: An Empirical Examination of the Online Consumer's Dilemma. *Information Technology and Management*, 6(2-3):181–202, apr 2005.

[20] PATRICIA A. NORBERG, DANIEL R. HORNE, and DAVID A. HORNE. The Privacy Paradox: Personal Information Disclosure Intentions versus Behaviors. *Journal of Consumer Affairs*, 41(1):100–126, jun 2007.

[21] Joseph Phelps, Glen Nowak, and Elizabeth Ferrell. Privacy Concerns and Consumer Willingness to Provide Personal Information. *Journal of Public Policy & Marketing*, 19(1):27–41, 2000.

[22] Kim Sheehan and Mariea Hoy. Dimensions of Privacy Concern Among Online Consumers. *Journal of Public Policy & Marketing - J PUBLIC POLICY MARKETING*, 19:62–73, 2000.

[23] H. Jeff Smith, Sandra J. Milberg, and Sandra J. Burke. Information Privacy: Measuring Individuals' Concerns about Organizational Practices. *MIS Quarterly*, 20(2):167, 1996.

[24] Kathy A. Stewart and Albert H. Segars. An Empirical Examination of the Concern for Information Privacy Instrument. *Information Systems Research*, 13(1):36–49, mar 2002.

[25] Sam Corbett-Davies and Sharad Goel. The Measure and Mismeasure of Fairness: A Critical Review of Fair Machine Learning. (Ec), jul 2018.

[26] Nicholas K Taylor, Elizabeth Papadopoulou, Sarah Gallacher, and M Howard. *Information Systems Development.* Number January. Springer New York, New York, NY, 2013.

[27] Hanbyul Choi, Jonghwa Park, and Yoonhyuk Jung. The role of privacy fatigue in online privacy behavior. *Computers in Human Behavior*, 81:42–51, 2018.

[28] Leslie K. John, Alessandro Acquisti, and George Loewenstein. Strangers on a Plane: Context-Dependent Willingness to Divulge Sensitive Information. *Journal of Consumer Research*, 37(5):858–873, feb 2011.

[29] Francesco Feri, Caterina Giannetti, and Nicola Jentzsch. Disclosure of personal information under risk of privacy shocks. *Journal of Economic Behavior and Organization*, 123:138–148, 2016.

[30] Helia Marreiros, Mirco Tonin, Michael Vlassopoulos, and M. C. Schraefel. "Now that you mention it": A survey experiment on information, inattention and online privacy. *Journal of Economic Behavior and Organization*, 140:1–17, 2017.

[31] Regulation (EU) 2016/679 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data (General Data Protection Regulation – GDPR). In *International and European Labour Law*, volume 2014, pages 958–981. Nomos Verlagsgesellschaft mbH & Co. KG, 2018.

[32] Bart P. Knijnenburg and Shlomo Berkovsky. Privacy for Recommender Systems. *Proceedings of the Eleventh ACM Conference on Recommender Systems - RecSys '17*, pages 394–395, 2017.

[33] Alfred Kobsa and Jörg Schreck. Privacy through pseudonymity in user-adaptive systems. *ACM Transactions on Internet Technology*, 3(2):149–183, 2003.

[34] Jörg Schreck. *Security and Privacy in User Modeling*, volume 2 of *Human-Computer Interaction Series*. Springer Netherlands, Dordrecht, 2003.

[35] Carter Jernigan and Behram F.T. Mistree. Gaydar: Facebook friendships expose sexual orientation. *First Monday*, 14(10), sep 2009.

[36] Matt Duckham and Lars Kulik. A Formal Model of Obfuscation and Negotiation for Location Privacy. In *Pervasive computing. Springer Berlin Heidelberg*, pages 152–170. 2005.

[37] Thomas Neubauer and Johannes Heurix. A methodology for the pseudonymization of medical data. *International Journal of Medical Informatics*, 80(3):190–204, 2011.

[38] Samson Yoseph Esayas. The role of anonymisation and pseudonymisation under the EU data privacy rules : beyond the 'all or nothing' approach. *European Journal of Law and Technology*, 6(2):1–28, 2015.

[39] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–487, 2013.

[40] J. Canny. Collaborative filtering with privacy. *Proceedings - IEEE Symposium on Security and Privacy*, 2002-Janua:45–57, 2002.

[41] Eugenia Politou, Efthimios Alepis, and Constantinos Patsakis. Forgetting personal data and revoking consent under the GDPR: Challenges and proposed solutions. *Journal of Cybersecurity*, 4(1):1–20, 2018.

[42] WC3. Proposal for an Open Profiling Standard, 1997.

[43] Susan Bull and Judy Kay. Student Models that Invite the Learner In: The SMILI:() Open Learner Modelling Framework. *International Journal of Artificial Intelligence in Education*, 17:89–120, 2007.

[44] Eran Toch, Norman M. Sadeh, and Jason Hong. Generating default privacy policies for online social networks. *Conference on Human Factors in Computing Systems - Proceedings*, pages 4243–4248, 2010.

[45] Umesh Shankar and Chris Karlof. Doppelganger: Better browser privacy without the bother. *Proceedings of the ACM Conference on Computer and Communications Security*, pages 154–167, 2006.

[46] Hans Löhr, Ahmad Reza Sadeghi, Claire Vishik, and Marcel Winandy. Trusted privacy domains - Challenges for trusted computing in privacy-protecting information sharing. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5451 LNCS:396–407, 2009.

[47] George O.M. Yee. An automatic privacy policy agreement checker for e-services. *Proceedings - International Conference on Availability, Reliability and Security, ARES 2009*, pages 307–315, 2009.

[48] Ramón Compañó and Wainer Lusoli. The Policy Maker's Anguish: Regulating Personal Data Behavior Between Paradoxes and Dilemmas. In *Economics of Information Security and Privacy*, number June, pages 169–185. Springer US, Boston, MA, 2010.

[49] Michelle Madejski, Maritza Johnson, and Steven M. Bellovin. A study of privacy settings errors in an online social network. *2012 IEEE International Conference on Pervasive Computing and Communications Workshops, PERCOM Workshops 2012*, (March):340–345, 2012.

[50] Bart P. Knijnenburg, Alfred Kobsa, and Hongxia Jin. Preference-based location sharing: Are more privacy options really better? *Conference on Human Factors in Computing Systems - Proceedings*, pages 2667–2676, 2013.

[51] A. Acquisti and J. Grossklags. Privacy and rationality in individual decision making. *IEEE Security and Privacy Magazine*, 3(1):26–33, jan 2005.

[52] Matthew Fredrikson and Benjamin Livshits. REPRIV: Re-imagining content personalization and in-browser privacy. *Proceedings - IEEE Symposium on Security and Privacy*, pages 131–146, 2011.

[53] Kevin Koidl, Owen Conlan, and Vincent Wade. Cross-site personalization. pages 66–76, 2014.

[54] Ari Juels. Targeted advertising... and privacy too. *Topics in Cryptology — CTRSA 2001*, 2020:408–424, 2001.

[55] Jesús López Miján, Irene Garrigós, and Sergio Firmenich. Supporting Personalization in Legacy Web Sites Through Client-Side Adaptation. In *Web Engineering*, volume 4607, pages 588–592. 2016.

[56] Jiaxi Tang and Ke Wang. Ranking distillation: Learning compact ranking models with high performance for recommender system. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2289–2298, 2018.

[57] Joan Serrà and Alexandros Karatzoglou. Getting deep recommenders fit: Bloom embeddings for sparse binary input/output networks. *RecSys 2017 - Proceedings of the 11th ACM Conference on Recommender Systems*, pages 279–287, 2017.

[58] JScrambler, 2016.

[59] Clemens Kolbitsch, Benjamin Livshits, Benjamin Zorn, and Christian Seifert. Rozzle: De-cloaking Internet malware. *Proceedings - IEEE Symposium on Security and Privacy*, pages 443–457, 2012.

[60] Saikat Guha, Alexey Reznichenko, Kevin Tang, Hamed Haddadi, and Paul Francis. Serving Ads from localhost for Performance, Privacy, and Profit. *HotNetsVIII Proceedings of the Eighth ACM Workshop on Hot Topics in Networks*, 2009.

[61] Darya Tarasowa, Ali Khalili, S Auer, and J Unbehauen. CrowdLearn: Crowdsourcing the Creation of Highly-structured E-Learning Content. *5th International Conference on Computer Supported Education*, pages 33–42, 2013.

[62] Drew Davidson, Matt Fredrikson, and Benjamin Livshits. MoRePriv: mobile {OS} support for application personalization and privacy. *ACSAC '14 Proceedings of the 30th Annual Computer Security Applications Conference*, pages 236–245, 2014.

[63] Louise Barkhuus and Valerie E. Polichar. Empowerment through seamfulness: Smart phones in everyday life. *Personal and Ubiquitous Computing*, 15(6):629–639, 2011.

[64] Markus Zanker, Laurens Rook, and Dietmar Jannach. Measuring the impact of online personalisation: Past, present and future. *International Journal of Human-Computer Studies*, 131(June):160–168, 2019.

[65] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys*, 52(1):1–35, 2019.

[66] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. Wide & Deep Learning for Recommender Systems. pages 1–4, 2016.

[67] Arvind Narayanan, Vincent Toubiana, Solon Barocas, Helen Nissenbaum, and Dan Boneh. A Critical Look at Decentralized Personal Data Architectures. 2012.

[68] Benjamin Heitmann and Conor Hayes. user profile portability on the Web of While most personalised services collect profile data from Data.

[69] Irene C.L. Ng. The Market for Person-controlled Personal Data with the Hub-of-all-Things (HAT). pages 1–26, 2018.

[70] Toly Chen. Obtaining the optimal cache document replacement policy for the caching system of an EC website. *European Journal of Operational Research*, 181(2):828–841, 2007.

[71] Waleed Ali, Siti Mariyam Shamsuddin, and Abdul Samad Ismail. A Survey of Web Caching and Prefetching. 3(1):1–27, 2011.

[72] Chetan Kumar and John B. Norris. A new approach for a proxy-level web caching mechanism. *Decision Support Systems*, 46(1):52–60, 2008.

[73] I Robert Chiang, Paulo B Goes, and Zhongju Zhang. Periodic Cache Replacement Policy for Dynamic Content at Application Server. *Decis. Support Syst.*, 43(2):336–348, mar 2007.

[74] Timo Koskela, Jukka Heikkonen, and Kimmo Kaski. Web cache optimization with nonlinear model using object features. *Computer Networks*, 43(6):805–817, 2003.

[75] Mudashiru Busari and Carey Williamson. ProWGen: A synthetic workload generation tool for simulation evaluation of web proxy caches. *Computer Networks*, 38(6):779–794, 2002.

[76] Heung Ki Lee, Baik Song An, and Eun Jung Kim. Adaptive prefetching scheme using web log mining in Cluster-based web systems. *2009 IEEE International Conference on Web Services, ICWS 2009*, pages 903–910, 2009.

[77] Jianhui Lin, Tianshu Huang, and Chao Yang. Research on WEB Cache prediction recommend mechanism based on usage pattern. *Proceedings - 1st International Workshop on Knowledge Discovery and Data Mining, WKDD*, pages 473–476, 2008.

[78] Abdolreza Abhari, Sivarama P. Dandamudi, and Shikharesh Majumdar. Web object-based storage management in proxy caches. *Future Generation Computer Systems*, 22(1-2):16–31, 2006.

[79] Thomas M Kroeger, Darrell D E Long, and Jeffrey C Mogul. Exploring the Bounds of Web Latency Reduction from Caching and Prefetching. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems on USENIX Symposium on Internet Technologies and Systems*, USITS'97, page 2, Berkeley, CA, USA, 1997. USENIX Association.

[80] Yin Fu Huang and Jhao Min Hsu. Mining web logs to improve hit ratios of prefetching and caching. *Knowledge-Based Systems*, 21(1):62–69, 2008.

[81] George Pallis, Athena Vakali, and Jaroslav Pokorny. A clustering-based prefetching scheme on a Web cache environment. *Computers and Electrical Engineering*, 34(4):309–323, 2008.

[82] Cheng Zhong Xu and Tamer I. Ibrahim. A keyword-based semantic prefetching approach in internet news services. *IEEE Transactions on Knowledge and Data Engineering*, 16(5):601–611, 2004.

[83] Jin Yu Ban Zhijie, Gu Zhimin. A Survey of Web Prefetching.

[84] Qinghui Liu. *Web Latency Reduction with Prefetching.* PhD thesis, Ont., Canada, Canada, 2009.

[85] Zhicheng Dou, Ruihua Song, and Ji-Rong Wen. A large-scale evaluation and analysis of personalized search strategies. *Proceedings of the 16th international conference on World Wide Web*, pages 581–590, 2007.

[86] Pawel J Kalczynski, Sylvain Senecal, and Jacques Nantel. Predicting On-Line Task Completion with Clickstream Complexity Measures: A Graph-Based Approach. *International Journal of Electronic Commerce / Spring*, 10(3):121–141, 2006.

[87] Dh Kim, V Atluri, Michael Bieber, and Nabil Adam. A clickstream-based collaborative filtering personalization model: towards a better performance. *6th annual ACM international workshop on Web information and data management*, pages 88–95, 2004.

[88] Zhong Su, Qiang Yang, and Hong-Jiang Zhang. A Prediction System for Multimedia Pre-fetching in Internet. *Proceedings of the Eighth ACM International Conference on Multimedia*, pages 3–11, 2000.

[89] Aleksandr Chuklin, Ilya Markov, and Maarten de Rijke. *Click Models for Web Search*, volume 7. jul 2015.

[90] Zhiwei Guan and Edward Cutrell. An eye tracking study of the effect of target rank on web search. *Proceedings of the 25th SIGCHI Conference on Human Factors in Computing Systems*, pages 417–420, 2007.

[91] Yuchen Zhang, Weizhu Chen, Dong Wang, and Qiang Yang. User-click modeling for understanding and predicting search-behavior. *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '11*, page 1388, 2011.

[92] Chao Wang, Yiqun Liu, Min Zhang, Shaoping Ma, Meihong Zheng, Jing Qian, and Kuo Zhang. Incorporating vertical results into search click models. *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval - SIGIR '13*, page 503, 2013.

[93] Q Guo and Eugene Agichtein. Exploring mouse movements for inferring query intent. *31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 707–708, 2008.

[94] Jeff Huang, Ryen W White, and Susan Dumais. No clicks, no problem: Using cursor movements to understand and improve search. *Proceedings of the 29th SIGCHI Conference on Human Factors in Computing Systems*, page 1225, 2011.

[95] Qi Guo and Eugene Agichtein. Towards predicting web searcher gaze position from mouse movements. *Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems - CHI EA '10*, page 3601, 2010.

[96] Kerry Rodden, Xin Fu, Anne Aula, and Ian Spiro. Eye-mouse coordination patterns on web search results pages. *CHI '08 Extended Abstracts on Human Factors in Computing Systems*, pages 2997–3002, 2008.

[97] Fernando Diaz, Ryen White, Georg Buscher, and Dan Liebling. Robust models of mouse movement on dynamic web search results pages. *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management - CIKM '13*, pages 1451–1460, 2013.

[98] Ernesto Arroyo, Ted Selker, and Willy Wei. Usability tool for analysis of web designs using mouse tracks. *CHI 06 extended abstracts on Human factors in computing systems CHI 06*, Montr\éal,(3):484, 2006.

[99] Vidhya Navalpakkam and Elizabeth Churchill. Mouse tracking: measuring and predicting users' experience of web-based content. *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems - CHI '12*, page 2963, 2012.

[100] Ahmed Awad E. Ahmed and Issa Traore. A New Biometric Technology Based on Mouse Dynamics. *IEEE Transactions on Dependable and Secure Computing*, 4(3):165–179, 2007.

[101] Cheng Jung Tsai, Ting Yi Chang, Yu Ju Yang, Meng Sung Wu, and Yu Chiang Li. An approach for user authentication on non-keyboard devices using mouse click characteristics and statistical-based classification. *International Journal of Innovative Computing, Information and Control*, 8(11):7875–7886, 2012.

[102] Agata Kolakowska, Agnieszka Landowska, Pawel Jarmolkowicz, Michal Jarmolkowicz, and Krzysztof Sobota. Automatic recognition of males and females among web browser users based on behavioural patterns of peripherals usage. *Dynamic Factor Models*, 35:317–360, 2016.

[103] Daniela Chudá and Peter Krátky. Usage of computer mouse characteristics for identification in web browsing. *Proceedings of the 15th International Conference on Computer Systems and Technologies*, pages 218–225, 2014.

[104] Arun Ranganathan. Revitalizing Caching, 2010.

[105] Kai Cheng and Yahiko Kambayashi. LRU-SP: A size-adjusted and popularity-aware LRU replacement algorithm for web caching. *Proceedings - IEEE Computer Society's International Computer Software and Applications Conference*, pages 48–53, 2000.

[106] Shiow Yang Wu, Jungchu Hsu, and Chieh Ming Chen. Headlight prefetching for mobile media streaming. *International Workshop on Data Engineering for Wireless and Mobile Access*, pages 67–74, 2007.

[107] Yue Cao, Ning Wang, Celimuge Wu, Xu Zhang, and Chakkaphong Suthaputchakun. Enhancing Video QoE over High-Speed Train Using Segment-Based Prefetching and Caching. *IEEE Multimedia*, 26(4):55–66, 2019.

[108] ProgrammableWeb - APIs, Mashups and the Web as Platform.

[109] Gustavo E. A. P. A. Batista and Maria Carolina Monard. An analysis of four missing data treatment methods for supervised learning. *Applied Artificial Intelligence*, 17(5-6):519–533, may 2003.

[110] Mario Villamizar, Oscar Garces, Harold Castro, Mauricio Verano, Lorena Salamanca, Rubby Casallas, and Santiago Gil. Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud. *2015 10th Colombian Computing Conference, 10CCC 2015*, pages 583–590, 2015.

[111] Khairul Baharein Mohd Noor. Case study: A strategic research methodology. *American Journal of Applied Sciences*, 5(11):1602–1604, 2008.

[112] Moshe Zviran, Chanan Glezer, and Itay Avni. User satisfaction from commercial web sites: The effect of design and use. *Information and Management*, 43(2):157–178, 2006.

[113] Selcuk Cebi. Determining importance degrees of website design parameters based on interactions and types of websites. *Decision Support Systems*, 54(2):1030–1043, 2013.

[114] Alexa.com. https://www.alexa.com/, 2019.

[115] Alexa.com. https://www.alexa.com/siteinfo/tcd.ie, 2019.

[116] Alexa.com. https://www.alexa.com/siteinfo/ocado.com, 2019.

[117] Eytan Adar, Jaime Teevan, and Susan T. Dumais. Resonance on the web. page 1381, 2009.

[118] George Papadakis, Ricardo Kawase, Eelco Herder, and Wolfgang Nejdl. Methods for web revisitation prediction: survey and experimentation. *User Modeling and User-Adapted Interaction*, 25(4):331–369, 2015.

[119] Hartmut Obendorf, Harald Weinreich, Eelco Herder, and Matthias Mayer. Web page revisitation revisited: implications of a long-term click-stream study of browser usage. *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 597–606, 2007.

[120] Andy Cockburn and Bruce McKenzie. What do web users do? An empirical analysis of web use. *International Journal of Human Computer Studies*, 54(6):903–922, 2001.

[121] Ashton Anderson, Ravi Kumar, Andrew Tomkins, and Sergei Vassilvitskii. The dynamics of repeat consumption. pages 419–430, 2014.

[122] Matthias Mayer. Web History Tools and Revisitation Support: A Survey of Existing Approaches and Directions. *Foundations and Trends® in Human–Computer Interaction*, 2(3):173–278, 2007.

[123] Janette Lehmann, Mounia Lalmas, Elad Yom-Tov, and Georges Dupret. Models of user engagement. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7379 LNCS:164–175, 2012.

[124] Xing Yi, Liangjie Hong, Erheng Zhong, Nathan Nan Liu, and Suju Rajan. Beyond clicks: Dwell time for personalization. *RecSys 2014 - Proceedings of the 8th ACM Conference on Recommender Systems*, pages 113–120, 2014.

[125] Patrick Mair and Marcus Hudec. Analysis of Dwell Times in Web Usage Mining. pages 593–600. 2008.

[126] Chee-hyung Yoon and Daniel Donghyun Kim. User Authentication Based On Behavioral Mouse Dynamics Biometrics. *Computer*, pages 0–4, 2005.

[127] Edgar Erdfelder, Franz FAul, Axel Buchner, and Albert Georg Lang. Statistical power analyses using G*Power 3.1: Tests for correlation and regression analyses. *Behavior Research Methods*, 41(4):1149–1160, 2009.

[128] Getting Started with Headless Chrome.

[129] Andreas Gizas, Sotiris Christodoulou, and Theodore Papatheodorou. Comparative evaluation of javascript frameworks. *Proceedings of the 21st international conference companion on World Wide Web - WWW '12 Companion*, (Cc):513, 2012.

[130] Benjamin Yen, Paul Jen Hwa Hu, and May Wang. Toward an analytical approach for effective Web site design: A framework for modeling, evaluation and enhancement. *Electronic Commerce Research and Applications*, 6(2):159–170, 2007.

[131] Jae Yoo Lee Jae Yoo Lee, Jung Woo Lee Jung Woo Lee, Du Wan Cheun Du Wan Cheun, and Soo Dong Kim Soo Dong Kim. A Quality Model for Evaluating Software-as-a-Service in Cloud Computing. *2009 Seventh ACIS International Conference on Software Engineering Research, Management and Applications*, pages 261–266, 2009.

[132] Zheng Li, Liam O'Brien, He Zhang, and Rainbow Cai. On a catalogue of metrics for evaluating commercial cloud services. *Proceedings - IEEE/ACM International Workshop on Grid Computing*, pages 164–173, 2012.

[133] David M Ciemiewicz. What Do You 'Mean': Revisiting Statistics for Web Response Time Measurements. *Computer Measurement Group Conf.*, pages 385–396, 2001.

[134] Charu Aggarwal, Joel L. Wolf, and Philip S. Yu. Caching on the World Wide Web. *IEEE Transactions on Knowledge and Data Engineering*, 11(1):94–107, 1999.

[135] Michael T.M. Emmerich and André H. Deutz. A tutorial on multiobjective optimization: fundamentals and evolutionary methods. *Natural Computing*, 17(3):585–609, 2018.

[136] Maja Pusara and Carla E. Brodley. User re-authentication via mouse movements. *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security VizSECDMSEC 04*, pages 1–8, 2004.

[137] Qi Guo and Eugene Agichtein. Ready to buy or just browsing?: detecting web searcher goals from interaction data. *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 130–137, 2010.

[138] Xia Chen and Huaqing Min. Improving click model by combining mouse movements with click-through data. *Proceedings of the IEEE International Conference on Software Engineering and Service Sciences, ICSESS*, 2015-Novem:183–187, 2015.

[139] Hossin M and Sulaiman M.N. A Review on Evaluation Metrics for Data Classification Evaluations. *International Journal of Data Mining & Knowledge Management Process*, 5(2):01–11, 2015.

[140] C. Ferri, J. Hernández-Orallo, and R. Modroiu. An experimental comparison of performance measures for classification. *Pattern Recognition Letters*, 30(1):27–38, 2008.

[141] Jennifer Neville, Brian Gallagher, and Tina Eliassi-rad. Statistical Tests for Network Classifier Evaluation. (1998).