International® Baccalaureate

Extended essay cover

Candidates must complete this page and then give this cover and their final version of the extended essay to their supervisor.			
Candidate session number			
Candidate name			
School number			
School name			
Examination session (May or November)	MAY	Year	2013

Diploma Programme subject in which this extended essay is registered: Computer Science

(For an extended essay in the area of languages, state the language and whether it is group 1 or group 2.)

Title of the extended essay: The viability of WebSockets as a replacement for AJAX in Large scale Web applications.

Candidate's declaration

This declaration must be signed by the candidate; otherwise a grade may not be issued.

The extended essay I am submitting is my own work (apart from guidance allowed by the International Baccalaureate).

I have acknowledged each use of the words, graphics or ideas of another person, whether written, oral or visual.

I am aware that the word limit for all extended essays is 4000 words and that examiners are not required to read beyond this limit.

This is the final version of my extended essay.

Candidate's signature: Date:

Supervisor's report and declaration

The supervisor must complete this report, sign the declaration and then give the final version of the extended essay, with this cover attached, to the Diploma Programme coordinator.

Name of supervisor (CAPITAL letters)

Please comment, as appropriate, on the candidate's performance, the context in which the candidate undertook the research for the extended essay, any difficulties encountered and how these were overcome (see page 13 of the extended essay guide). The concluding interview (viva voce) may provide useful information. These comments can help the examiner award a level for criterion K (holistic judgment). Do not comment on any adverse personal circumstances that may have affected the candidate. If the amount of time spent with the candidate was zero, you must explain this, in particular how it was then possible to authenticate the essay as the candidate's own work. You may attach an additional sheet if there is insufficient space here.

From brainstorming to completion, demonstrated the highest levels of commitment, interest, curiosity, focus, polish and passion for the process and outcome of his extended essay. The sources of information and material researched enhanced his previous knowledge and understanding of the topic, which developed paying close attention to the assessment criteria and requirements for the task. He attended every meeting, worked independently and reflected carefully on supervisor feedback. His extended essay addresses the research question as throughly as possible, using sound arguments and first hand testing of the technologies compared in his essay.

This declaration must be signed by the supervisor; otherwise a grade may not be issued.

I have read the final version of the extended essay that will be submitted to the examiner.

To the best of my knowledge, the extended essay is the authentic work of the candidate.

I spent [3] hours with the candidate discussing the progress of the extended essay.

Supervisor's signature: Date:

Assessment form (for examiner use only)

Criteria	Achievement level				
	Examiner 1	maximum	Examiner 2	maximum	Examiner 3
A research question	2	2		2	
B introduction	2	2		2	
C investigation	4	4		4	
D knowledge and understanding	3	4		4	
E reasoned argument	3	4		4	
F analysis and evaluation	3	4		4	
G use of subject language	4	4		4	
H conclusion	2	2		2	
I formal presentation	3	4		4	
J abstract	2	2		2	
K holistic judgment	3	4		4	
Total out of 36	31				

International Baccalaureate Extended Essay

Computer Science

The viability of WebSockets as a replacement for AJAX in large scale web applications

---

Session: May 2013

Candidate:

Word Count: 3958

Author:

Supervisor:

October 17, 2012

Abstract

Web based applications are becoming more advanced and are requiring greater amounts of data to be sent and received from servers. Traditional methods of data synchronization like Asynchronous JavaScript and XML (AJAX) between client and server are no longer sufficient to satisfy this need for instant information updates and low overhead communication. WebSockets are a new type of web based browser-server communication that features server side data pushing and low overhead communication.

This investigation into *The viability of WebSockets as a replacement for AJAX in large scale production web applications* will allow a fair comparison to be undertaken. This investigation is done in segments, examining each protocol in terms of design and implementation, speed and reliability, and practical considerations. The protocols are then compared to each other using these criteria to determine their suitability.

This investigation has found that WebSockets are more efficient in data transmission and utilize resources more efficiently to achieve a richer, and faster user experience. Protocol and developer programming overheads are lower with WebSockets than they were with AJAX as the API is more elegantly designed. However it is seen that AJAX will remain widely used until browser support and documentation for WebSockets has significantly improved. Overall, technically the WebSocket protocol is ready for a large scale implementation but the supporting elements will still take time to mature. In the future it is very likely that WebSockets will be used as a replacement for AJAX requests in large scale web applications.

Contents

1 Introduction

Over the past decade the internet has become a prominent aspect of human communication, and the protocols which it is based upon have evolved along with it. Web based applications have been around for many years and have changed drastically with the creation of newer client-server technologies. In particular Asynchronous Javascript and XML (AJAX) has become a very popular choice for client-server communication in web applications. This technology is in use by large scale web applications such as but not limited to Google, Facebook, and Wordpress, and these handle very high traffic loads in their data centers. (Harold)

AJAX is a web programming technique by which the client running JavaScript code in a web browser is able to poll a server, through which new data can be send and received asynchronously. This makes it possible for the web application to fetch data without the need to disrupt the workflow of the user by the action of reloading the page. Each AJAX call is a separate HTTP request with its own headers and connection tracking. In this sense, with respect to the underlying protocols, it is the same as loading a webpage. In large web applications where many of these requests are generated in short time intervals, a large overhead is created and this becomes inefficient (Garrett)

WebSockets, as defined in RFC 6455, is a new web protocol that has been proposed by the World Wide Web Consortium (W3C) and the Internet Engineering Task Force (IETF). This new specification defines a protocol which enables full duplex communication, through a TCP stream that stays open until the user closes the application (Hickson). This allows for efficiency, speed and an improved user experience.

An investigation into the practicality of using WebSockets instead of the current AJAX technologies would be beneficial in reducing resources that are currently needed to run and maintain current large scale application deployments. Google currently serves its web applications through its own purpose built data centers, which consume 260 Megawatts of electricity. A switch to a more lean and efficient protocol would free up a lot of processing that is currently required by protocols such as AJAX, and therefore save time and energy.

This investigation into the viability of WebSockets as opposed to AJAX requests in large scale web applications will first define a set of requirements that are essential to a protocols used in web application, then proceed to an in depth exploration of both protocols, followed by a comparison. The outcome of this investigation will determine how realistic it would be to implement WebSockets in a large web application at this point in time.

2 Methodology of Evaluation

The aim of this research paper is to evaluate the viability of WebSockets as a client-server transport medium for large scale web applications. To achieve this aim a testing framework must be defined to allow for a fair and unbiased evaluation. This testing framework is a qualitative way of measuring the suitability of the protocol for the tasks at hand.

2.1 General Criteria

These general criteria will be applied to the specific criteria (listed in the next section) for success to yield a measure of each protocol's success.

Level	Description
3	The objective of this criteria is fully met and satisfied to make this suitable for large scale web applications. The implementation meets all standards described in the specific criteria for success section and lends itself to a strong and robust protocol for information interchange within this context.
2	The objectives outlined are met to an extent that allows for adequate functionality in the context of large scale web applications. Only minor problems exist that can be worked around and still allow for the specific criteria to be met.
1	There are lapses in the protocol that so not allow for parts of the specific criteria to be met. There are still some aspect of this criterion that are satisfied and benefit large scale web applications.
0	This criteria is not met in any way by the protocol under evaluation. Significant lapses in this area have an effect on its use for large scale web applications.

Figure 1: Grading criteria for aspects of the protocols to be examined

2.2 Specific criteria for success

2.2.1 Design and Implementation

This criterion looks at the design of the protocol and how well this is suitable for the purposes of large scale web applications.

Criterion	Description
Design of the Application Programming Interface (API)	When developing large web applications, the programming interface must be logical and easy to understand in order to facilitate development.
Programming overheads related to implementing the protocol	During the developmental process it is important to be able to use the basic functionality of the protocol with the least amount of extraneous code possible. This means that if a developer wants to use the protocol in its most basic form , then they should be able to implement this without needing to know about advanced options that do not relate to the task they want to achieve.
Documentation available for the protocol	For new web developers and those who want to gain an in depth understanding of the protocol without reading the source code documentation is important, this criteria looks at the availability , completeness and availability of materials documenting the protocols.

Figure 2: Criteria in Design and Implementation category

2.2.2 Speed and Reliability

This criterion looks at the protocols in terms of their real world performance and what this means for large scale web applications.

Criterion	Description
Request speed and protocol overhead	This aspect is concerned with the way in which data is sent across the network, and weather it is appropriate for sending large amounts of data frequently as is often the case with modern large web applications.
Data integrity and data loss prevention	This relates to any measures that are in place to prevent data corruption while being transmitted using the protocol, and the effectiveness of safeguards that are in place to handle or prevent such incidents.
Server side processing	regards the overheads on the server side for processing requests made using the protocol. This aspect would include architectural considerations

Figure 3: Criteria in Speed and Reliability category

2.2.3 Practical considerations

This criterion is targeted to evaluate the possible problems that may be evident in a real life implementation of either protocol in a large scale deployment.

Criterion	Description
Browser compatibility	This aspect relates to the extent to which the protocol is supported in browsers that end users use at the current time of writing.
Compatibility with other network elements	Concerns how well the protocol is treated by elements common in computer networks today, such as but not limited to proxy servers, firewalls, and antivirus programs.

Figure 4: Criteria in Practical considerations category

3 AJAX

Asynchronous JavaScript and XML is a collective term which refer to technologies which allow web applications to make independent requests to a web server without page reloads. A standard HTTP request is used as the vector for information transmission, JavaScript is used to initiate, monitor and process the results of the request and XML encapsulates transmitted data.

AJAX is used in modern web applications to load information asynchronously without disrupting the user. Such solutions are critical where response time is key. Live web based collaborative software and online chat applications will require near instant response speed as the user experience is defined by response speed.

3.1 Protocol design

This protocol is based upon raw HTTP requests that are initiated by JavaScript code rather than the user clicking a link, or loading a resource for a page. In terms of client—server communication protocols it is identical to normal HTTP, however in its technical implementation, AJAX calls are made through the XMLHttpRequest API.

3.2 Anatomy of an AJAX request

There are distinct aspects that make up an AJAX request, and these are shown in figure 5

Stage	Description
Initialization	JavaScript code prepares the request and inserts data into the request such as: URL, request type, and data,
Request execution	The web browser registers the request and processes the request using its low level networking handlers, This includes 1. Formation of HTTP packet — Data is collected and inserted into an HTTP packet, 2. DNS lookup — The hostname of the web server is looked up either in the DNS cache (locally) or on a remote DNS server, 3. Establishing the connection — A TCP connection is established between server/client, exact details of this process are beyond the scope of this essay, and the relevant packets are sent,
Request Monitoring	While the request is being executed the browser will allow the JavaScript code to monitor the status of the request through the HTTPRequest.status property.
Result and closing of the connection	The result of the request is received by the browser, or an error code, and this is available to the JavaScript code for processing.

Figure 5: The constituent parts of an AJAX request

These stages are performed in each AJAX request, and some steps are redundant when making many requests.

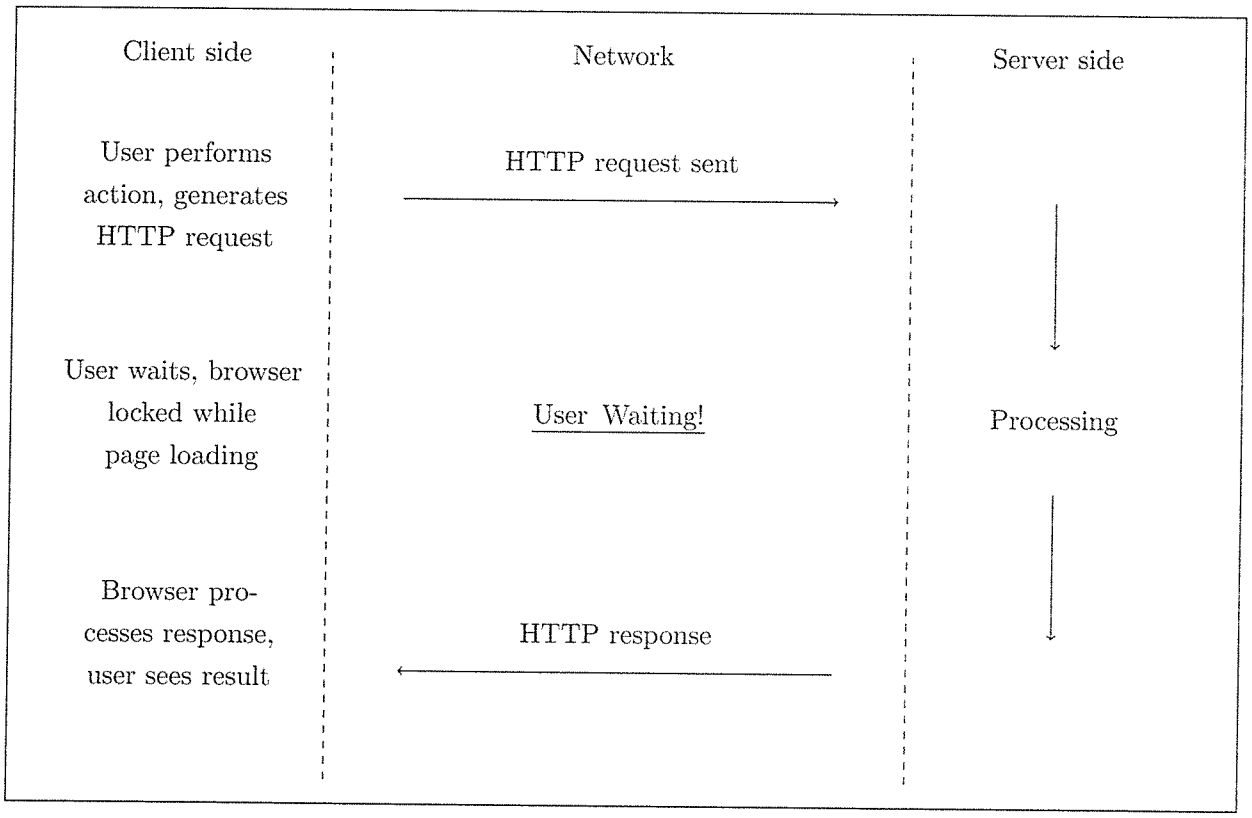

Figure 6: Illustration of traditional synchronous HTTP requests

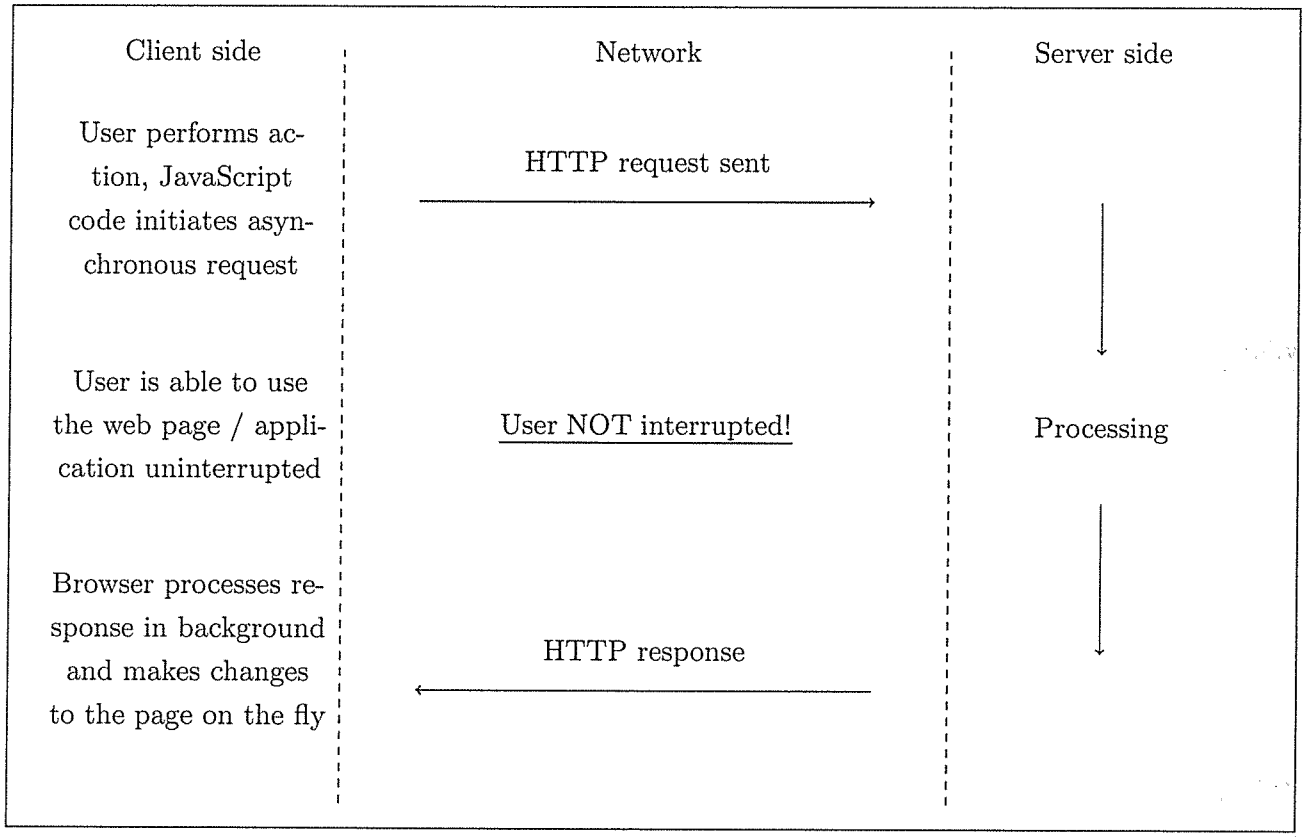

Figure 7: Illustration of Asynchronous HTTP requests as implemented in AJAX

Figure 6 and Figure 7 above show the difference between normal HTTP requests and those made through AJAX calls.

Each AJAX request that is made must go through this process and multiple connections can be made to the same server. In large web applications multiple requests may need to me made at the same time, and in rapid succession. This means that there is inefficiency in that a new connection needs to be established for each request, often to the same server.

3.3 Use in large scale web applications

AJAX requests are the current standard way of getting up to date information in a web browser without the need of page reloading. AJAX requests can be utilized by web applications to fetch resources based on the client's needs (i.e. dynamically based on user actions), or to fetch updates from the server. The latter is more widely used and concerns this research topic more, thus it will be focused upon.

The main limitation of AJAX as it is implemented in web applications today is that the client does not know when new content will be available, and needs to constantly poll the server. This is highly inefficient as computer networks are inherently two way, but HTTP requests only allow for server—client data pushing if the client initiated the request first. Constant polling consumes resources continually

despite being no new information to be sent. (Jquery Foundation)

Workarounds to this problem have been found with AJAX, for example long polling is used to emulate server pushes. This works by creating HTTP request to the web server that is locked until new content is available. This forces the connection to remain open, when new data is received a new connection must be established. Moreover these are unstable as HTTP connections were not designed to be so long lived and can time-out.

3.4 Analysis with respect to criteria

3.4.1 Design and Implementation

API Design

AJAX requests are made using the *XMLHttpRequest* object that is available in modern browsers (See section on browser compatibility), and this object has methods and attributes used to initiate, configure, send, and monitor the status of the request. The main methods / attributes are displayed in the table below.

Method	Description	Use
open	Initializes the AJAX request with parameters such as URL and request type	Used to initialize an AJAX request before sending it.
send	Executes the AJAX request	Used to initiate a request after the parameters have been specified.
readyState	Contains the current HTTP status code of the request	Client side code can use this to monitor the progress of the request as well as knowing if the request was successful or a failure.
responseText	Contains the result of the AJAX call	Used to fetch the result of the request
responseType	The content type of the result	Used by the client side code to know the data type of the response and how to parse it appropriately.
upload	Provides a representation of how much an upload is complete	Used by client side code to indicate to the user how much of an upload is complete.

Figure 8: Commonly used attributes of the XMLHttpRequest Object

The design of the API is simple to use and develop with, however some methods are not intuitive to use. For example the readyState attribute could have been replaced by a simpler one to indicate if the connection was ready or not. This criterion scores 2 points.

Programming Overheads

The understanding of the protocol and functioning of HTTP is required to use AJAX requests. The developer must be able to understand what HTTP status codes are to be able to handle them properly. For example, when monitoring the status of an AJAX request to see weather or not it has completed, the "readyState" value will need to be compared to the value for a successful request [1]. (Mozilla Foundation)

This introduces extra knowledge and concepts that the programmer will need to understand to implement AJAX, the score in this criterion is 2.

Documentation

AJAX is a technique that has existed fro six years in mainstream web programming. It was first seen in 2006 with the introduction of Google maps (Mozilla Foundation) and the term was coined by Jesse James Garrett of Adaptive Path (Garrett). There are numerous websites, books and tutorials available; in addition to the official documentation that exists for the XMLHttpRequest object on the W3C working group website (Van Kesteren)

This criterion scores 3 points.

3.4.2 Speed and Reliability

Request speed and protocol overhead

AJAX requests are HTTP requests that have been initiated through the use of the XMLHttpRequest object, and are therefore identical to requests made when loading a page. HTTP requests were originally designed to be only used to load webpages and their dependencies, however now as they are being used to dynamically load content several parts of its design have become redundant. (Van Kesteren)

For example the stages of DNS lookup and establishing a new connection to the server are redundant if many connections are to be made such as in the case of web applications where data is polled from the same server very frequently. Many steps are repeated due to the design of the protocol only being able to support one request per connection.

The protocol is grossly inefficient with regards to its use in web applications, however it still is able to provide some functionality. This criterion's score is 1.

Data integrity

There are several levels of safeguards in place to catch errors that may occur in an AJAX request, these safeguards are on different levels depending on the type of error to be caught.

On a high level, HTTP status codes are used to verify the status of a request made to the server. These are implemented by the developer and are caught in the client code itself. When this type of error is detected the code may take an action such as displaying an error to the user or retrying to poll the sever.

[1] The value for a successful HTTP request is "200 OK"

On a lower level there exists built in safeguards to protect against the mis-transmission of information. On the TCP level, CRC checksums are employed to verify that the packets have been transmitted correctly between the client and server. These errors are picked up by network equipment and compensated for.

The data corruption prevention methods are comprehensive and sufficient to catch any errors. The score in this criterion is 3.

Server side processing

On the server side, the HTTP requests will need to be processed as separate entities as there is no link between two consecutive requests from the same client apart from data *within* the request itself.

In web applications, session cookies are used to ensure that the user is logged and permit the application to know which user created the request. This applies to AJAX calls in particular as a session cookie is the only way for the application to know which user created the request.

When an AJAX request is received on the server side it needs to be processed as any other HTTP request would, the stages of these are shown in the diagram below.

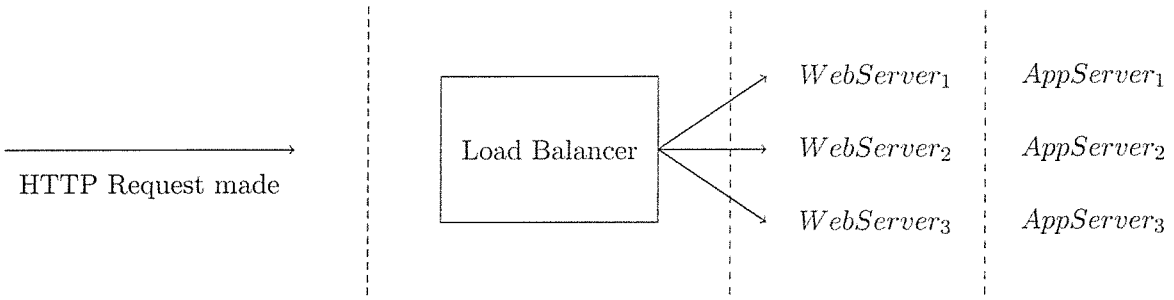

Figure 9: The typical request processing architecture of a web application

Since all the web servers are equally capable of handling the requests, a load balancer distributes the request across an array of servers to distribute the load. However despite being able to share the load over many servers, each server must still parse the contents of each packet and match the cookie to a user in the system. Due to the separate nature of these requests, the user's credentials must be checked at each request, thus utilizing resources. The score in this criterion is 1 for reasons of inefficiency.

3.4.3 Practical considerations

Browser compatibility

The XMLHttpRequest API has long been part of the official W3C specification for JavaScript (MacCaw) and is well supported by most browsers that have full JavaScript support. There are certain specific cases in which AJAX requests will not work, such as cross domain requests, and the use of browser security extensions that disable certain features for security reasons [2] The following table shows browser compatibility.

[2] An example of this would be the NoScript extension for Mozilla Firefox which disables scripts on unknown websites by default and enforces additional restrictions on web pages. `http://noscript.net/`

Web browser	Current Version	Support for XMLHttpRequests	Other notes
Mozilla Firefox	15	Full support	-
Google Chrome	22	Full support	-
Apple Safart (WebKit)	6	Most - Missing JSON support	-
Microsoft Internet Explorer	9	Full support	Older versions may have different API's

Figure 10: Browser compatibility. (Source: Microsoft Developer Network, Mozilla Foundation, Opera Software,)

As all major browsers support commonly used features of the XMLHttpRequest object, this criterion score is 3.

Compatibility with other network elements

Since HTTP is such a well established and well known protocol, many other pieces of software and hardware have been designed to support its transmission through networks that make up the internet. Devices such as layer 4 switches and transparent proxies know and understand how HTTP works and support its transport through the internet. There is no problem with the transmission of AJAX requests through these networks and devices as everything that supports HTTP will also support AJAX requests.

This criterion scores 3 points.

4 WebSockets

WebSockets are a new web communication standard proposed by the W3C (Hickson) and has been standardized by the Internet Engineering Task Force (IETF) in RFC 6455. WebSockets allow for web browsers and servers to have a full duplex communication channel for clients to send requests and servers to push new information to clients (Grigorik). In the past such two way communication was not possible without the use of inefficient or dangerous workarounds that did not work under all circumstances. This protocol allows for a TCP connection to be established after completing a handshake which is initiated in an HTTP request.

4.1 Protocol Design

A websocket can be broken down into three main parts, establishing a connection through a "HTTP upgrade" request, sending messages between hosts, and the closing of the connection.

```
GET /Application HTTP/1.1
Host: websocket.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: I83Hus92bXBsAIRub83PRQ==
Origin: http://example.com
Sec-WebSocket-Protocol: WSProtocol
Sec-WebSocket-Version: 13
```

Figure 11: An example WebSocket client upgrade request

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: WSProtocol
```

Figure 12: An example WebSocket client upgrade server handshake

Figure 11 and 12 show the initialization handshakes that occur between the client and the server when a HTTP connection is upgrading to a WebSocket one. Once this is established there is a two way communication channel that can be used by the server to push information and by the client to upload data. When the connection is being closed specialized status codes are sent to indicate the reason for closure. (MacCaw)

4.2 Use in large scale web applications

WebSockets are a new technology and have not been implemented in large scale web applications yet because of compatibility concerns. Despite this, the benefits that the technology could bring to the web application industry are vast and substantial. Currently there is no way for web servers to push new information to clients, clients must request new information from servers. This process is not only time and resource consuming, but also defeats the purpose of having the HTTP protocol. The protocol which underlays HTTP: TCP is by nature a two way protocol that allows for a connection to be kept open and information sent in full duplex mode (Two way communication). HTTP, which is used ubiquitously for web pages and applications only allows for requests to be made by web browsers. To achieve "two way"-like communication , AJAX polling and long polling are used by web developers to be able to have up to date information in the browser. (Hansa)

4.3 Evaluation with respect to criteria

4.3.1 Design and implementation

Design of the API

The WebSocket API is defined in RFC 4655 and the working draft by the W3C, the relevant methods that will be used in an implementation of WebSockets in a web application are covered here.

Method / Attribute	Description	Use
WebSocket	This is the constructor of the WebSocket object and accepts an argument specifying the URL of the WebSocket endpoint. When this method is invoked, an attempt is made to establish a WebSocket connection to the server.	This method is used in applications to establish connections before further operations can be carried out.
onOpen / onClose	These methods are used to perform callback functions when the connection is opened or closed respectively.	This is used in web applications to make it easier for the developer to monitor for such events. The developer does not need to know the underlying protocols which comprise the WebSocket protocol, nor do they need to understand the status codes. These methods automatically perform a callback when the connection is opened or closed.
onMessage	This method is automatically called when there is a new message form the server and is used to parse such messages	In applications this will be method that parses the message from the server and performs an appropriate action as defined by the developer.
send	This method is used to send data and requests from the server to the client.	This is used in applications to request for a certain action to be done or new user generated data to be sent to the server
onError	Used for error handling	In web applications this would be used to take appropriate actions should an unexpected situation arise. For example this could tell the user that an error occurred and to refresh the page.

Programming overheads

The API for the WebSocket protocol is very clear with its method naming and no extra knowledge of underlying protocols. To create a simple implementation of the WebSocket protocol in an application, only basic methods such as WebSocket(...) onMessage(...) and send(...) are required. Score: 3 points in this criteria.

Documentation and frameworks

The WebSocket protocol is one that has been recently proposed as a web standard and not fully finalized. For this reason the amount of documentation available for this protocol as compared to other more established ones is rather limited. Websockets are covered in books by technical publishers such as O'Reilly (MacCaw) , but not in great depth. On the internet one can find tutorials, example implementations and official specifications.

Completeness and accuracy are hard to achieve in the case when a new technology is being discussed as the standards are always changing. In the case of WebSockets there is a description of the API available at the latest RFC page (Fette), as well as on the developer page for specific web browsers (Mozilla Foundation)

4.3.2 Speed and reliability

Request speed and protocol overheads

WebSockets are separate from HTTP in the sense that they are an independent TCP connection between the browser and the server. This connection does not close when the request is over as it does for traditional HTTP requests, but rather it remains open for the entire lifetime of the webpage (i.e. when the webpage is closed the connection closes). This is an efficient and elegant solution as the handshake and connection establishment only needs to happen once, and all subsequent requests are made through this one connection. This eliminates the need of establishing multiple connections and the redundant steps associated with that. For this reason the WebSocket protocol is fast and efficient. (Holdener)

Furthermore since it is a two way communication channel, the server is able to push new information to the client without it being explicitly requested for. This allows for near instant updates to information that has changed and is in direct contrast to previous techniques that involve the client periodically polling the server for new information (such as long polling).

This method is appropriate for sending / receiving large amounts of data in discrete events, such as receiving a new email or sending a status update to a server.

Data integrity

There are mechanisms in place which have been defined in the WebSocket protocol to handle accidental data corruption Or unexpected results such as receiving data of the wrong encoding. Each WebSocket packet which is sent across the network contains an opcode which is used to indicate the type of data being sent in that packet, and this can be used to verify that the data received is of the correct type.

Error code / range	Description
1000	Connection closed normally
1002	Protocol error
1003	Connection closed due to unexpected data being received
1004	Too much data received
1006	Abnormal circumstances, for use in cases where the connection was suddenly dropped without proper notification of closure

Figure 13: Error codes in WebSocket communication. Source: (Autobahn WebSockets)

Moreover, the TCP packets themselves contain checksums which can be used to detect any corruption in data. If any corruption is detected at this level, lower level networking equipment will automatically ask for the data to be re-sent.

In addition to these lower level safeguards, WebSocket server endpoints enforce certain measures in cases where they have received invalid data. In such cases, the server can send back special error codes (see Figure 13) which correspond to a certain anomaly or misunderstanding. When such a code is received it is dealt with accordingly by the server or client.

Section score: 3.

Server side processing

The only server side processing that needs to be done is generating the handshake and listening for new events. The connection is always open and additional processing does not need to be done for each request that the client generates.

Section score 3 points.

4.3.3 Practical considerations

Browser compatibility

As this is a new technology and has not been finalized as of yet, many browsers have preliminary support for the protocol but do not have comprehensive implementations due to the constant changes being made.

Web browser	Current Version	Support for WebSockets
Mozilla Firefox	15	Most features supported in draft specification
Google Chrome	22	Most features supported in draft specification
Apple Safart (WebKit)	6	No Support for Latest WebSocket specification
Microsoft Internet Explorer	9	No Support for Latest WebSocket specification

Figure 14: Browser compatibility. (Source: Stack Overflow)

Note that some browsers have disabled WebSockets on the grounds of security concerns. It can be seen that there is not widespread support for a common protocol across browsers. For this reason the score in this criteria is 1.

Compatibility with network elements

The WebSocket protocol encounters some challenges when it comes to compatibility with existing network equipment, particularly because the protocol is not HTTP based. When a WebSocket connection is being established, the client sends an upgrade offer (refer to protocol design) which asks the server to establish a WebSocket connection with the desired endpoints.

Network elements such as proxy servers and level 4+ switches need to understand the traffic that passes through them. If the WebSocket connection needs to pass through one of these elements then the connection may be dropped, errors may be returned, or other unexpected consequences may result. (Hickson)

However in unrestricted networks where computers can make connections to ant port this is not a great concern. Proxy servers and firewalls likely to block these connections are found in corporate environments.

The score in this criterion is 2.

5 Comparison

Aspect	Ajax	WebSocket
DESIGN AND IMPLEMENTATION		
API Design	2	3
Programming Overheads	2	3
Documentation	3	1
SPEED AND RELIABILITY		
Request speed and Protocol Overhead	1	3
Data integrity	3	3
Server side processing	1	3
PRACTICAL CONSIDERATIONS		
Browser Compatibility	3	1
Compatibility with network elements	3	2
TOTAL	18	19

Figure 15: Table of quantitative results

5.1 Design and Implementation

The WebSocket protocol has significant changes and improvements over AJAX. The API in particular is designed in a more intuitive manner with clearer method names and a reduction in unnecessary procedures. It is easier to develop simple applicaitons with little knowledge of underlying protocols, and faster to develop complex applications as the design is more logical.

However due to the fact that WebSockets are a new protocol: documentation is lacking. This has the implication of it being difficult for new developers to learn and use thr ptotocol if they do not want to read the technical specifications.

5.2 Speed and Reliability

The most significant improvement that WebSockets bring to web applications is in the area of speed. AJAX requests are traditional HTTP requests that need to be processed by the server, and created every time an update needed to be fetched. WebSockets on the other hand are able to handle two way communication through an always open TCP tunnel. Furthermore, WebSockets allow for server sent pushes that save the client from having to do costly polling on the server.

With respect to server side processing, AJAX requests need to reprocess each request as a separate one, and thus reauthenticate a user every time whey want to perform an action. The use of a single channel eliminated the problem as the user only has to be authenticated once and subsqeuent requests can be trusted.

These improvements will substantially improve both developer, user, and administrator experiences because of the reduction in unnecessary steps and simplification of problems.

5.3 Practical considerations

WebSockets are lacking when it comes to practical considerations. Many end user browsers still do not have full support for this protocol and will need be able to run such applications. AJAX on the other hand has nearly ubiquitous support in web browsers and is widely used in web applications today.

The fact that the protocol has only recently been drafted adds problems when it comes to network compatibility as network equipment such as proxy serves may not understand the new protocols. AJAX is based directly on HTTP and therefore cannot be distinguished from HTTP by proxy servers, making integration seamless.

6 Conclusion

Based on this investigation as a whole it is possible to have a clear position on the viability of WebSockets as a replacement for AJAX in large scale web applications. Through this in depth analysis of both protocols , we can see the strengths and weaknesses of each and how this relates to the research question.

AJAX is a very well supported, understood, and documented programming technique. It is widely implemented but lacks support for two way communication, which is the main reason that developers are looking to alternatives such as WebSockets for future web applications. WebSockets has the advantage of being able to send server side "pushes" of information to the client , and clients have the advantage of making less connections.

WebSockets also features a clean API that is easy to understand and work with, unlike that of AJAX which requires in depth knowledge of underlying protocols to implement a working solution. This has the implication of making large web application development faster as the code is more comprehensible.

The main limiting factor, however, is that WebSockets are not very well supported by client side , and network devices. Browser support is not consistent and documentation is not always available. These features, however are subject to change with time. As more and more people consider WebSockets, support will grow and developer resources will be written.

It is possible to conclude that today, WebSockets should be explored in beta versions of web applications to understand the power it can have of enriching the user experience. That said it should only be rolled out when support is more widespread in the near future.

6.0.1 Evaluation

This evaluation only looks at a portion of the protocols that were deemed important to answering the research question. That is to say there may be other factors outside the scope of this essay that may affect a real life implementation. Furthermore this evaluation was done from a very academic and theoretical viewpoint. In real life computers and networks operate at speeds unimaginable to the human mind, and the theoretical differences may not be noticeable to the user. However in very large web applications, every bit transferred, watt consumed and HTTP request made matters as it is amplified over the thousands or millions of people who use it.

7 Bibliography

"Autobahn WebSockets." AutoBahn WebSocket Client and Server framework. Tavendo GmbH, n.d. Web. 6 Aug. 2012. <autobahn.ws/>.

Fette, I, and A Melnikov. "RFC 6455 WebSockets." Internet Engineering Task Force (IETF). N.p., n.d. Web. 3 June 2012. <tools.ietf.org/html/rfc6455>.

Grigorik, Ilya. "Server-Sent Event Notifications with HTML5 - igvita.com." Ilya Grigorik - igvita.com. N.p., n.d. Web. 1 June 2012. <http://www.igvita.com/2011/08/26/server-sent-event-notifications-with-html5/>.

Garrett, Jesse James. "Ajax: A New Approach to Web Applications." Adaptive Path. N.p., n.d. Web. 16 June 2012. ¡http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications¿.

Glanz, James. "Google Details, and Defends, Its Use of Electricity." The New York Times 8 Sept. 2011: 5. Print.

Hansa, Umar. "Start Using HTML5 WebSockets Today — Nettuts+." Web development tutorials, from beginner to advanced — Nettuts+. N.p., n.d. Web. 16 May. 2012. <http://net.tutsplus.com/tutorials/javascript-ajax/start-using-html5-websockets-today/>.

Harold, Elliotte Rusty, and W. Scott Means. XML in a nutshell. 3rd ed. Sebastopol, CA: O'Reilly, 2004. Print.

Hickson, Ian. "The WebSocket API." W3C Public CVS Repository. N.p., n.d. Web. 16 Aug. 2012. <http://dev.w3.org/html5/websockets/>.

Holdener, Anthony T.. Ajax: the definitive guide. Beijing: O'Reilly, 2008. Print.

"JavaScript - What browsers support HTML5 WebSocket API? - Stack Overflow."Stack Overflow. N.p., n.d. Web. 18 June 2012. <http://stackoverflow.com/questions/1253683/what-browsers-support-html5-websocket-api>.

Jquery Foundation. "Jquery API Reference : Ajax." Jquery JavaScript Library. N.p., n.d. Web. 20 July 2012. <api.jquery.com/jQuery.ajax/>.

MacCaw, Alex. JavaScript web applications. Sebastopol, CA: O'Reilly, 2011. Print.

Mozilla Foundation. "XMLHttpRequest - MDC ." Mozilla Developer Center. N.p., n.d. Web. 30 May 2012. <https://developer.mozilla.org/en-US/docs/DOM/XMLHttpRequest>.

Mozilla Foundation. "WebSockets." Mozilla Developer Center. N.p., n.d. Web. 12 June 2012. <https://developer.mozilla.org/en-US/docs/WebSockets>.

Powers, Shelley. Adding Ajax. Farnham: O'Reilly, 2007. Print.

Van Kesteren, Anne. "XMLHttpRequest Level 2 Editor's protocol working draft." World Wide Web Consortium (W3C). N.p., n.d. Web. 16 Oct. 2012. <http://www.w3.org/TR/XMLHttpRequest/>.