August 7th, 2017

Internxt

Welcome To A New Internet

August 7th, 2017

Table of contents

Abstract

There is a growing demand for more aspects of the modern internet to be decentralized. Though internet applications are built on top of protocols like TCP/IP and HTTP, a large portion of the internet stack remains centralized. Much of the desire for more decentralized computing systems comes from concerns regarding mass surveillance over the web and security of the files on the cloud. At Internxt we are helping shape a new, more secure, private and efficient internet. Using the most fundamental infrastructure behind the internet and with the use of Blockchain and decentralized technology, we strive to create a better internet than the one that is currently available. We believe that right now the cloud is relatively easy to access by hackers trying to retrieve someone's files. Additionally, we believe that Governments and corporations have too much say over people. Although there needs to be regulation to avoid unethical and illegal activities, as of now, control over people is excessive. Overall, we believe it’s time for a better internet. In the following sections, we explain more about our work, and give details about what and how it will be done. Welcome To A New Internet. Welcome to Internxt.

Introduction

Internxt is a Peer-To-Peer (P2P) cloud computing network that allows users from all over the globe to cooperate in the creation of a decentralized internet. Users can sell the resources of their machines to those looking to host their data in a more private, secure and efficient way. The cloud platform Internxt will provide will not only offer a better alternative to traditional cloud services in terms of technology, but it will also strive to be competitive in terms of price. Besides that, we also strive to make our technology intuitive and as user friendly as already existing top-tier services. We want to make sure this new internet is accessible by everyone, regardless of their knowledge, thus we want to focus on providing a seamless transition from traditional services to Internxt, without compromising on features.

Why

There are many online industries where decentralized computing systems are uprooting existing businesses. With the rise Bitcoin and its underlying technology, industries like online commerce, file storage amongst many more, are being disrupted by shifting to more decentralized models. Blockchain technology enables many of such applications to be decentralized. P2P systems differ from other distributed systems in enabling the user network to function without the need for an entity monitoring them. In pure P2P architectures, there are no centralized services or control mechanisms dictating the actions of other nodes. Each user decides with how many computing resources he/she will contribute to the network, as well as when and for how long. The architecture is designed to handle large numbers of nodes joining or leaving the network at the same time. In addition, these systems emphasize equality and balance the load across nodes. This flexibility, self-determination and low participation cost encourages a much larger number of participants, which, in turn, greatly increases the number and value of the services provided by the system to all. We plan to make use of such architecture and create a robust network that will be the foundation stone of our technology. Everything ran on Internxt's network will be built using P2P technology. By spreading the files over the network and removing the central points of failure we remove data centralization and get enhanced security, which is a big flaw of cloud computing that is present today.

A major issue with existing cloud platforms is security and data centralization. In fact, cloud computing is a computing paradigm, an abstraction where data and services are accessible all over the network to authorized users and processes. Abstraction of computing away from the physical host entails a loss of control of corporate data and loss of visibility into where the data lives and who has access to it. Another dimension to cloud computing with very serious implications for security is the deployment model: who owns the infrastructure and how is it accessed. A Private cloud refers to a collection of resources used by a single organization. This is typically owned and managed by the organization itself, and hence in practical terms is little different than any other data owned and managed by that enterprise. A Public cloud refers to resources accessible by anyone usually over the public internet, managed and owned by a third party. A third category, Hybrid cloud, refers to a combination of private and public clouds along with the connecting fabric between the two. Our cloud brings a new and revolutionary solution into this. The data is spread on public network run by users, but all the files are encrypted before being sent out. There are no single points for the hackers to attack, it is very hard for someone to grab a hold of files, and even if they do get them, they cannot use them without the private key held by user. By using such architecture, our cloud is better built than any other available solution, and it benefits in many matters. We want to fight this security and privacy issues the current internet is facing with the development of Internxt.

What

Websites and Applications

We want to develop a decentralized solution through which users can create and browse any kind of decentralized application, static or dynamic website. Decentralized applications include any application that is currently built on Java, php etc. Decentralized apps and websites would be built with OpenShift technology, and it would include all the languages supported by default in OpenShift, although we would add more languages as we develop the technology. Decentralized websites would be able to be created via SFTP and SSH, and browsable from any Web browser, through http protocol. Combining industry standard technologies with powerful automation features would improve both the time from idea to having the infrastructure ready for either staging or going directly to production. Our decentralized web and application hosting would be much secure than other solutions, thanks to the distribution of files. Users will be able to deploy and run websites and applications on computing resources running on novel P2P Grid application virtualization. Websites and Apps would be completely distributed, which would make these much more secure than traditional centralized services.

Secure Massive Scale Object Storage (dCDN)

The concept of object storage was introduced in the early 1990's by the research community. Since then it has greatly matured. An object is defined as data (typically a file) along with all its metadata, all bundled up as an object. This object is given an ID that is typically calculated from the content of that object (both file and metadata) itself. An object is always retrieved by an application by presenting the object ID to object storage. Unlike files and file systems, objects are stored in a flat structure. You have a pool of objects, and you simply ask for a given object by presenting its object ID. Objects may be local or geographically separated, but because they are in a flat address space, they are retrieved the same way. An object is not limited to any type or amount of metadata.

Internxt will enable the creation of P2P self-managed, shared and secure storage for storage networks. This moves lower-level functionalities such as space management into the storage device itself, accessing the device through a standard object interface

In an object store environment space is allocated by the object store and not by the higher-level software such as a file system. Users of an object store, e.g., the file system, operate on data by performing operations such as creating an object, reading/writing at a logical location in the object, and deleting the object. In addition, all operations carry a credential, and it is the responsibility of the object store to validate that the user's request carries a valid credential. This per-object credential allows the storage to enforce different access rights for different portions of a volume on an object granularity.

Object storage is ideal for solving the increasing problems of data growth. As more and more data is generated, storage systems must grow at the same pace. Another advantage to object storage its responsiveness to the need for resiliency while mitigating costs. Objects remain protected by storing multiple copies of data over a distributed system; if one or more nodes fail, the data can still be made available, in most cases, without the application or the end user ever being impacted.

Internxt would run an algorithm to ensure that only legal files are being hosted on the network. Unlike traditional providers, Internxt is private and does not make any use of the user information. In the decentralized file hosting solution, Internxt would run scans to ensure that these securely and privately hosted files do not contain any illegal activity.

Smartphone storage

Internxt would offer two different solutions directly related to how smartphones (IOS and Android) store user data. The first of the two technologies is aimed for the end-customer. We will integrate a mobile application which will let the user backup part of its data in our decentralized infrastructure, with the aim of hosting it more securely than having it locally, and allowing the user to save up space in the smartphone. An example could be photos stored in the phone itself or on services like Google Photos, or game data that is internally stored.

The second smartphone solution residing on Internxt's platform would be oriented towards developers. These will be able to use a service like Parse (rapid development oriented services for easily storing mobile application data), but on a decentralized manner. This would allow developers to host part of their mobile application data on a more private and secure cloud.

Internxt Serverless Framework solution

Serverless was first used to describe applications that significantly or fully depend on 3rd party applications / services (in the cloud) to manage server-side logic and state. These are typically rich client applications (single page web apps, or mobile apps) that use the vast ecosystem of cloud accessible databases (Parse, Firebase), authentication services (Auth0, AWS Cognito) etc. These types of services have been previously described as '(Mobile) Backend as a Service', and I'll be using 'BaaS' as a shorthand in the rest of this article. Serverless could also be defined as applications where some amount of server-side logic is still written by the application developer but unlike traditional architectures is run in stateless compute containers that are event-triggered, ephemeral (may only last for one invocation), and fully managed by a 3rd party.

With Internxt, the user will be able to deploy their web, mobile or IoT applications directly to Internxt, and the whole infrastructure will entirely be provided by us. It offers a powerful new way of developing/running applications as micro or nano services.

The Vulnerability Scanner (VS) is our threat detection component. When users deploy new kubernetes clusters, they register the clusters with the Vulnerability Scanner and install scanner agents in the clusters. When the scanner detects a threat, it posts a notification to the users' registered OpenWhisk action which will determine the appropriate actions to take to mitigate the threat. The notification contains the identity of the container pods impacted by the threat. The OpenWhisk policy action invokes the appropriate Kubernetes API extension which will relay the operation to the Security Enforcement Operator (SEO).

Control Tower application

Another key concept about our platform is the “Control Tower” app that will allow users to monitor and control everything related to our platform. It will give users a clear overview of the resources that they are hosting in the Internxt network. We will also create a web interface, followed by smartphone and computer applications. Additionally, it will give users an overview of the number of tokens they have, and they will see the payments that are pending both to you for the resources they are providing, and payments pending for resources requested by them.

This will also allow the overview and the control of the applications that are hosted by users. There will be developer API bundled with this, that will allow easy integration with other supported products and platforms.

User support

Due to the core importance of providing a unique user experience, our decentralized cloud services will provide 24/7 support to our users, apart from a seamless easy-to-use interface. Although we plan to integrate an AI chatbot throughout all our environment, we also plan to introduce 24/7 live chat support with specialized agents for deeper user concerns and 24/7 phone support for any matters.

Additionally, we plan to create a decentralized network of support where users with technical background could provide their knowledge at the disposal of users who using our platform who would need support. We would use Internxt’s token to pay for their services. Reputation system would also be created, which would help improve the overall performance of the platform.

How

Internxt network

Historically, the state of computing has gone through a series of platforms and environmental changes. Distributed computing holds great assurance for using computer systems effectively. As a result, supercomputer providers and datacenters have changed from providing high performance floating point computing resources to concurrently servicing a huge number of requests from billions of users. A distributed computing system uses multiple computers to solve large-scale problems over the internet. It becomes data-intensive and network-centric. The applications of distributed computing have become increasingly wide-spread. In distributed computing, the main stress is on the large-scale resource sharing and always goes for the best performance.

Distributed systems can be considered conventional networks of independent computers. They have multiple system images, as each node runs its own operating system, and the individual machines in a distributed system could be, for example, combinations of Massively Parallel Processors (MPPs), Symmetric Multiprocessors (SMPs), clusters, and individual computers.

Peer-to-peer (P2P) networks and grids are distributed computing models that enable decentralized collaboration by integrating computers into networks in which each can consume and offer services. P2P is a class of self-organizing system or application that takes advantage of available distributed resources. A grid is a geographically distributed computing platform comprising a set of heterogeneous machines that users can access through a single interface. Unlike the classic client-server model, in which roles are well separated, P2P and grid networks can assign each node a client or server role according to the operations they are to perform on the network, even if some nodes act more as server than as client in current implementations. Both are concerned with the same general problem, namely, the organization of resource sharing within virtual communities; both take the same general approach to solving this problem, namely the creation of overlay structures, but the underlaying organization of this structure greatly differs amongst these two technologies.

While computing Grids have been widely used in computational science, Peer-to-Peer computing has achieved wide prominence in the context of multimedia file exchange. It uses the computing power at the edge of a connection rather than within the network. The client/server architecture does not exist in a peer-to-peer system. Peer nodes act as both clients and servers - their roles are determined by the characteristics of the tasks and the status of the system. This architecture minimizes the workload per node, and maximizes the utilization of the overall computing resources among the network. Today, the sheer numbers of desktop systems make the potential advantages of interoperability between desktops and servers into a single Grid system quite compelling. However, these commodity systems have significantly different properties than the conventional server-based Grid systems. They are usually highly autonomous and heterogeneous systems. And their availability varies from time to time.

Cloud services are mainly divided into three services delivery models: SaaS (Software as a Service, e.g. Google Mail), PaaS (Platform as a Service, e.g. Google AppEngine) and IaaS (Infrastructure as a Service, e.g. Amazon EC2). We normally view cloud computing as providing some mixture of three core types of service: Infrastructure as a Service (IaaS), typically servers, storage and network devices provided through virtualization; Platform as a Service (PaaS), used to deploy applications that are provided by the cloud provider, provider's partners or the cloud customer; and Software as a Service (SaaS), where software applications are accessed over the network (usually the internet) as hosted services. But there is always an intention to develop a better technology.

Clouds have evolved as the next-generation platform that facilitates creation of wide-area on-demand renting of computing or storage services for hosting application services that experience highly variable workloads and requires high availability and performance. Interconnecting Cloud computing system components (servers, virtual machines (VMs), application services) through peer-to-peer routing and information dissemination structure are essential to avoid the problems of provisioning efficiency bottleneck and single point of failure that are predominantly associated with traditional centralized or hierarchical approaches. These limitations can be overcome by connecting Cloud system components using a structured peer-to-peer network model

Internxt's mission is to combine P2P and Grid into nonhierarchical decentralized grid system by going beyond standard P2P. We aim to build P2P Grid network, next generation peer-to-peer platform. It will be a truly decentralized structured P2P system which does not require central coordination or knowledge. On top of P2P Grid we aim to provide decentralized platform, a combination of IaaS and PaaS with multiple services that act as building blocks for variety of usages. P2P systems are commonly classified into two categories: unstructured systems exposing emergent phenomena driven from purely local interactions, and structured (DHT-based) systems with probabilistic execution guarantees. Internxt will combine both based on research done in P2P Grid computing.

Node organization

In the P2P community, a fundamental distinction is made among unstructured and structured P2P systems for resource location. In unstructured P2P systems in principle peers are unaware of the resources that neighboring peers in the overlay networks maintain. Typically, they resolve search requests by flooding techniques. Using the network maintenance protocol, a peer discovers new peers in the network by flooding discovery. Using the network maintenance protocol, a peer discovers new peers in the network by flooding discovery messages. From the responses, it (randomly) selects certain peers to which direct network links are established. The property is accounted to the mechanism of how these networks are being constructed: New nodes preferentially attach to already well-connected ones. Thus, it's a completely decentralized but also self-organizing system: From randomized interactions of peers' global structures emerge.

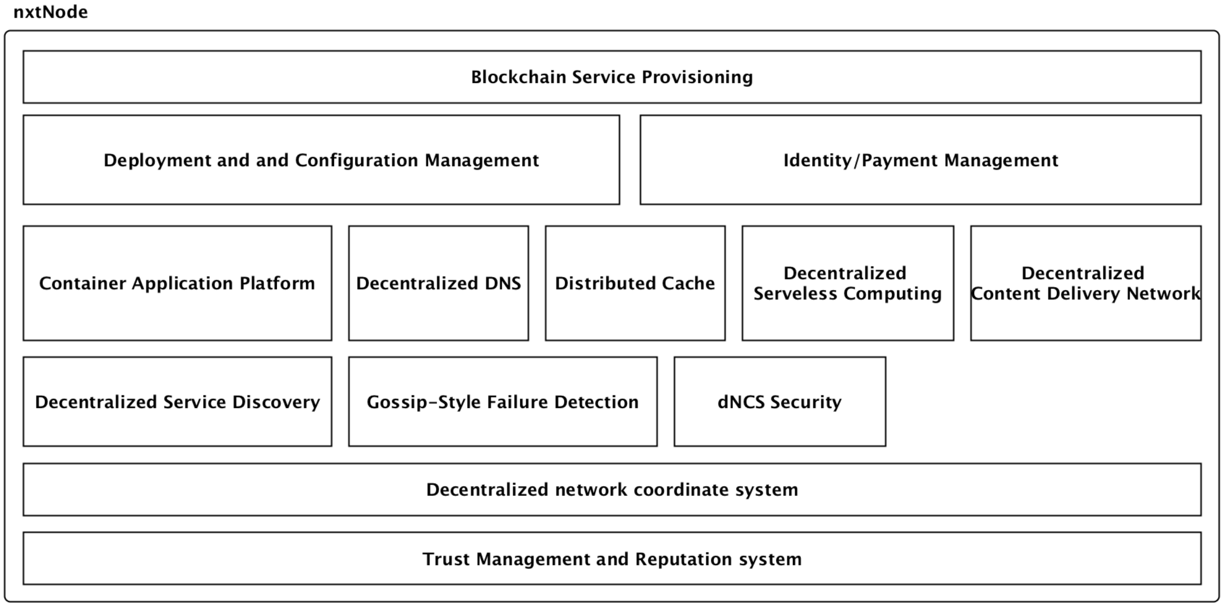
In contrast, standard structured P2P systems follow a different approach with respect to network maintenance. They assign static identifiers to peers and the distributed data structures (e.g., DHTs) are constructed based on these identifiers by distributed algorithms. As a result, the overlay network structure is mainly determined by the choice of identifiers and in turn any self-organization of the system is prevented.

Load balancing as a resource allocation problem is critical to support high scalability, availability, accessibility, and throughput. For systems supporting equality-based lookup of data only, the problem of non-uniform workloads may be circumvented by applying hash functions to the data keys, thus uniformly distributing workload, both for storage and query answering. In combination with using balanced search structures, i.e., balanced distributed search trees, this approach leads to uniform load distribution among the participating peers. However, it is limited if further semantics of the data keys is exploited, for example, in the simplest case when the ordering of data keys is used to support prefix or range queries. This is critical for DB-oriented applications.

P2P Grid is a peer-to-peer lookup system based on a virtual distributed search tree, similarly structured as standard distributed hash tables, and which can solve load balancing issues present in traditional technologies. Each peer holds part of the overall tree. Every participating peer's position is determined by its path, that is, the binary bit string representing the subset of the tree's overall information that the peer is responsible for. Query routing approach is as follows: For each bit in its path, a peer stores a reference to at least one other peer that is responsible for the other side of the binary tree at that level. Thus, if a peer receives a binary query string it cannot satisfy, it must forward the query to a peer that is “closer” to the result. The salient feature of P2P Grid, in contrast to other DHT-based P2P systems, is the separation of concern between peer identifier and peer’s path. In P-Grid peer paths are not determined a priori but are acquired and changed dynamically through negotiation with other peers as part of the network maintenance protocol. Thus, P2P Grid’s prefix-routing infrastructure is constructed by means of a decentralized, self-organizing process in which it adapts to a given distribution of data keys stored by the peers.

We assume here that Grids are sharing environments implemented via the deployment of a persistent, standards-based service infrastructure that supports the creation of, and resource sharing within, distributed communities. Resources can be computers, storage space, sensors, software applications, and data, all connected through the internet and a middleware software layer that provides basic services for security, monitoring, resource management, and so forth. Because accessing these decentralized resources means operating in an environment of unstable connectivity and unpredictable IP addresses, P2P design requirements commonly include independence from DNS and significant or total autonomy from central servers. Their implementations frequently involve the creation of overlay networks [53] with a structure independent of that of the underlying internet

We will build upon these and foundations of distributed computing paradigms in an effort to create a decentralized secure and reliable platform. The following graph show an overview of the technology behind Internxt, which is further discussed in the coming sections.


nxtNode
Blockchain Service Provisioning
Deployment and and Configuration Management
Identity/Payment Management
Container Application Platform
Decentralized DNS
Distributed Cache
Decentralized Serveless Computing
Decentralized Content Delivery Network
Decentralized Service Discovery
Gossip-Style Failure Detection
dNCS Security
Decentralized network coordinate system
Trust Management and Reputation system

Blockchain Service Provisioning

The process of deploying application services on publicly accessible clouds that expose their capabilities as a network of virtualized services (hardware, storage, database) is known as Cloud Provisioning. The Cloud provisioning process consists of two key steps VM Provisioning, this involves instantiation of one or more VMs on physical servers hosted within public or private Cloud computing environments. The selection physical server for hosting VMs in a cloud is based on number of mapping requirements including available memory, storage space, and proximity of the parent cloud; and Application Service Provisioning, the second step is mapping and scheduling of requests to the services that are hosted within a VM on a set of VMs.

We aim to build cloud provisioning platform with following features: On-demand dynamic computing, pay-per-use, Proximity aware server location.

To meet today's requirements, provisioning techniques and services should be able to: dynamically adapt to performance needs, failure, and leave and join of hardware and software, including VMs, servers, storage, software, applications, and networks; discover and monitor state of services in completely decentralized and distributed manner; accomplish coordinated and load-balanced provisioning of VMs and application services; handle spikes in service demand (workload) through dynamic scaling in of services from other public clouds; and handle authentication and authorization of services for users; provisioning users’ access; federated security model.

A fundamental challenge in managing the Cloud provisioning system is to maintain a consistent connectivity between the components (self-organization). This challenge cannot be overtaken by introducing a central network model to connect the components, since the information needed for managing the connectivity and making the decisions is completely decentralized and distributed.

Further, centralized network model does not scale well, lacks fault-tolerance, and requires expensive server hardware infrastructure. System components can leave (VM instance destruction), join (VM creation), and fail (service outage) in a dynamic fashion; hence it is an impossible task to manage such a network centrally. Therefore, an efficient decentralized or peer-to-peer solution is mandatory that can gracefully adapt, and scale to the changing conditions.

In peer-to-peer organization of Cloud provisioning systems both control and decisions making are decentralized by nature and where different system components interact together to adaptively maintain and achieve a desired system wide behavior. A distributed Cloud provisioning configuration is decentralized if none of the components in the system are more important than the others, in case that one of the component fails, then it is neither more nor less harmful to the system than caused by the failure of any other component in the system.

Traditionally, DHTs have been efficient for single-dimensional queries such as “finding all resources that match the given attribute value”. Since Cloud computing IaaS and PaaS level services such as servers, VMs, enterprise computers (private cloud resources), storage devices, and databases are identified by more than one attribute; therefore, a search query for these services is always multi-dimensional. These search dimensions or attributes can include service type, processor speed, architecture, installed operating system, available memory, and network bandwidth.

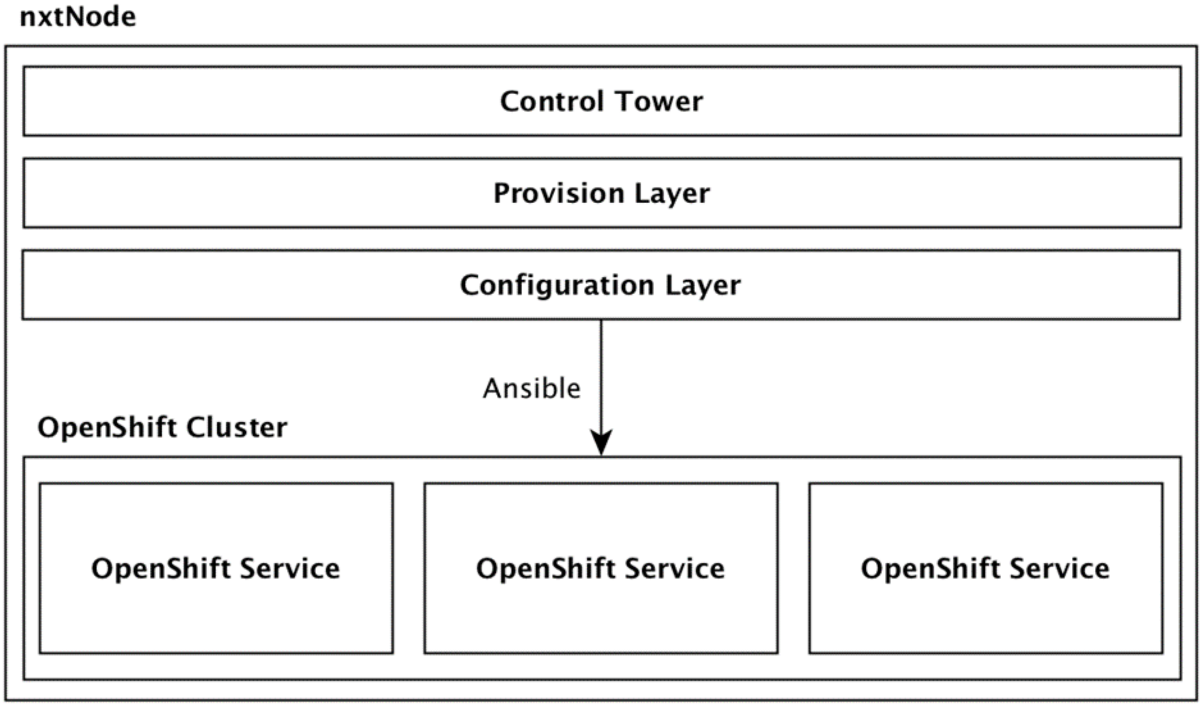
Internxt will be making use of Blockchain technology in different parts throughout its whole environment. Blockchain will mainly be used as a backbone for all payments throughout Internxt’s network. We will make use of Ethereum smart contracts and will create Internxt (IXT) network tokens based on the ERC 20 token standard due to its robust structure. Smart contracts are instances of code that live inside the Ethereum network, and it is impossible for anyone to breach or modify them. They are public, and allow transparent payments and job running. For the standard user, smart contracts remove the need of an additional middle man when processing payments. With this in mind, we will create an interface that communicates with smart contracts, and fully automate all the tasks done in our platform like deployment of applications inside the network, or storing user files. Internxt network’s availability and consistency are of core relevance for the platform. We will therefore run algorithms for watching over the performance of network hosts. This would allow to users to increase their rewards with constant performance over time.

Deployment and Configuration Management

Automation plays a vital role in ensuring we can release software repeatedly and reliably. One key goal is to take repetitive manual processes like build, deployment, regression testing and infrastructure provisioning, and automate them. This is the part of the system that handles initial distribution of files onto the Internxt network. After the user has uploaded the files he wishes to run or store on our network, and if the number of tokens connected to his account is enough for the calculated expenses, files are delivered to the network. All the settings related to site hosting, level of security and additional encryption and distribution of files are presented to the users, and in depth explained.

We will use Ansible to provide best user experience for configuration management, and application deployment. In contrast with popular configuration management software — such as Chef, Puppet, and CFEngine — Ansible uses an agentless architecture. With an agent-based architecture, nodes must have a locally installed daemon that communicates with a controlling machine. With an agentless architecture, nodes are not required to install and run background daemons to connect with a controlling machine. This type of architecture reduces the overhead on the network by preventing the nodes from polling the controlling machine.

To orchestrate nodes, Ansible deploys modules to nodes over SSH. Modules are temporarily stored in the nodes and communicate with the controlling machine through a JSON protocol over the standard output.



Identity and Payment Management

Identity management, also known as identity and access management (IAM) is, in computer security, the security and business discipline that "enables the right individuals to access the right resources at the right times and for the right reasons". It addresses the need to ensure appropriate access to resources across increasingly heterogeneous technology environments and to meet increasingly rigorous compliance requirements.

Application hosted in our network, that allows users to have an overview over the balance of Internxt tokens, how they spend them (in terms of the content that is hosted on the INX network), and how much profit they are making from the resources their provide. This includes a full fledged web and mobile apps that user can use to manage his own content. Communication between user client and the server will be encrypted. The server will then communicate with the blockchain and if the payment is successful, deployment and configuration management parts of the system are contacted. Automation and User-friendliness are key for our network. User is able to pay for what he uses, and earn tokens for resources he provide through our platform, and all of that secured by the Ethereum backbone network. Not only that Ethereum is a great solution today, it is also in constant improvement and as it evolves, so will our platform grow as well, and this will allow us to add new features in the close future.

Container Application Platform

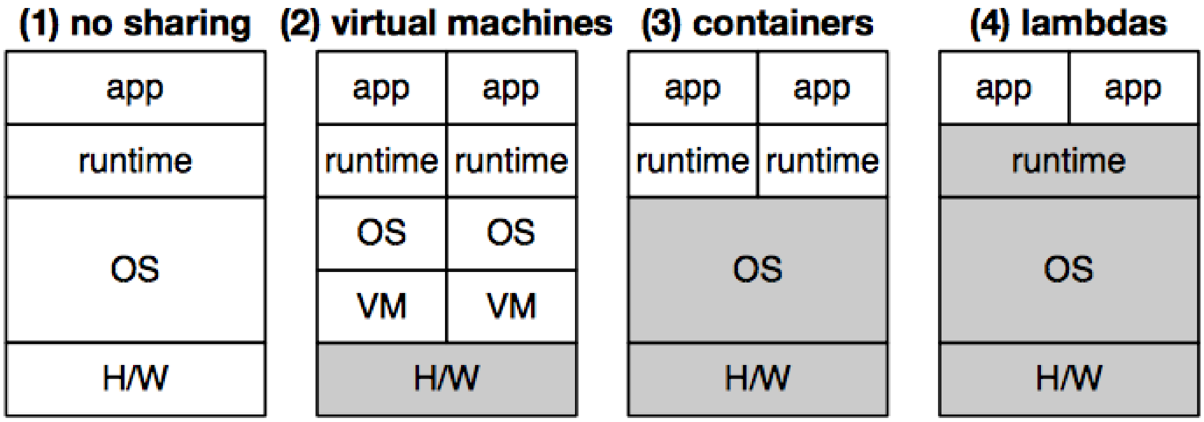
Application containerization is an operating system level (OS-level) virtualization method for deploying and running distributed applications without launching an entire virtual machine (VM) for each app. Instead, multiple isolated systems are run on a single control host and access a single kernel. The application containers hold the components such as files, environment variables and libraries necessary to run the desired software. Because resources are shared in this way, application containers can be created that place less strain on the overall resources available. This is the layer that allows the regular users to plug in the their personal computers, or dedicated machines into Internxt. For this we plan to use OpenShift, that provides enterprise Kubernetes for developers. OpenShift Origin is a distribution of Kubernetes optimized for continuous application development and multi-tenant deployment. OpenShift adds developer and operations-centric tools on top of Kubernetes to enable rapid application development, easy deployment and scaling, and long-term lifecycle maintenance for small and large teams. We plan to use this for easy distribution of Docker images to user machines, that will allow all the users to work with our system with ease. OpenShift allows a huge specter of tech to be run out of the box, including: Java (Wildfly, JBossEAP, Tomcat), PHP, Node.js, Python, Perl, MySQL, PostgreSQL, MongoDB, Jenkins, Cron, and JBoss xPaaS Services (Fuse, BPM Suite, BRMS, Data Virtualization, Aerogear, and more). We plan to extend on their framework, and if needed adapt it to our needs. This will represent the backbone of our system that connects all the other parts: it will be used for deployment and provisioning of all the resources inside the network. It allows all the computers inside the network to be able to run as one, and create the Internxt as we imagine it. We will also provide an developer API that will allow users to connect to the network, and use it instead of others existing cloud solutions. The API will be created so that there is no danger of running malicious code or harming the network in any way. We find OpenShift superior solution compared to similar solutions like Cocaine. OpenShift runs and supports both stateful and stateless applications. This allows you to take full advantage of containers without needing to completely re architect your enterprise apps.

Distributed Cache

In computing, a distributed cache is an extension of the traditional concept of cache used in a single locale. A distributed cache may span multiple servers so that it can grow and in transactional capacity. It is mainly used to store application data residing in database and web session data. Distributed caches, although deployed on a cluster of multiple nodes, offer a single logical view (and state) of the cache. In most cases, an object stored in a distributed cache cluster will reside on a single node in a distributed cache cluster. By means of a hashing algorithm, the cache engine can always determine on which node a key-value resides. Since there is always a single state of the cache cluster, it is never inconsistent. The idea of distributed caching has become feasible now because main memory has become very cheap and network cards have become very fast, with 1 Gbit now standard everywhere and 10 Gbit gaining traction. Also, a distributed cache works well on lower cost machines usually employed for web servers as opposed to database servers which require expensive hardware. We want to provide this, since we think it is a feature that is perfectly fit into the architecture of our system, and brings a lot of benefits in terms of speed and network reliability. A cache provides high throughput, low-latency access to commonly accessed application data, by storing the data in memory. For a cloud app, the most useful type of cache is distributed cache, which means the data is not stored on the individual web server's memory but on other cloud resources, and the cached data is made available to all of an application's web servers (or other cloud VMs that are used by the application). Since the concept of a “server” is different for us, and we have a lot of memory available in the network, it is easy to make use of, and deliver content with high speeds, regardless of scaling of applications on a high number of places. A distributed cache runs as an independent process across multiple nodes and therefore failure of a single node does not result in a complete failure of the cache. Because of a node failure, items that are no longer cached will make their way into surviving nodes on the next cache miss. Also in the case of distributed caches, the worst consequence of a complete cache failure should be degraded performance of the application as opposed to complete system failure.

Decentralized Serverless Computing

Serverless computing is a cloud computing execution model in which the cloud provider dynamically manages the allocation of machine resources. Pricing is based on the actual amount of resources consumed by an application, rather than on pre-purchased units of capacity. It is a form of utility computing. Serverless architectures refers to applications that significantly depend on third-party services (knows as Backend as a Service or "BaaS") or on custom code that's run in ephemeral containers (Function as a Service or "FaaS"), the best-known vendor host of which currently is AWS Lambda. Instead of thinking of applications as collections of servers, developers instead define applications with a set of functions with access to a common data store. The Lambda model has many benefits as compared to more traditional, server-based approaches. Lambda handlers from different customers share common pools of servers managed by the cloud provider, so developers need not worry about server management. The code specific to an application will typically be small, and hence it is inexpensive to send the handler code to any worker in a cluster. Finally, applications can scale up rapidly without needing to start new servers. In this manner, the Lambda model represents the logical conclusion of the evolution of sharing between applications, from hardware to operating systems to the runtime environments themselves.



Again, we consider this to be a perfect fit for the architecture that runs the Internxt network. We will be making use of Open Lambda, as an open source and reliable project, that allows serverless computing and is compatible with other technical solutions we already use (eg OpenShift, Docker). Serverless computing is used more, and it represents the future of applications. With serverless computing and Internxt network it is easy to detect and scale the needs for resources needed for application to run. We allow users to host their services on our platform, with easy integration to any existing application through standard REST interface. Additional benefit is ease of payment for the user - like for everything else in our system, it is done by smart contracts that monitor the usage of resources. By writing good code, this would allow the publishers to get an incredible scaling, at low prices.

Decentralized Content Delivery Network

Due to increase in internet usage nowadays users are accessing information from all part of the world so we need to have an effective content delivery system to deliver content in a fast and effective way. The problem of existing content distribution to users is by delivering the data through single transmission from source to one point (cache server in that branch) and the copies are created at that branch point and distributed to each requested user node. We have many existing content delivery networks which delivers content quickly and effectively by responding to the user request from the nearest geo location cache server. But enabling a decentralized content caching capability removes the need of central cache server and increases the performance, data availability and speed of content delivery process. Here content is cached in mostly every intermediate node within the network and content is fetched by its content name not based on the hostname. Requested content is delivered to the node from the nearest node holding the content cache, in case if that node is unavailable the request is shifted to the next nearest node holding the content. Content delivery networks (CDN) is a cache based technique which stores a data in cache server place at different locations. When a user request for a specific data from the server, the nearest CDN will respond with data to the user’s request. Peer-to-Peer (P2P) system also use a caching technique but here the participant node who are all accessing the data will become a cache server and respond to other users. P2P data is stored in one user’s node is made available to another user. So, the distribution load is shared among users instead of central serve. Even if the user is respond by the CDN having single cache server for one nation or one zone can’t be the solution. In Peer-to-Peer the data requested is fetched from another participant who is requesting the same data from the server.

Decentralized DNS

DNS (Domain name service) is used to resolve human readable addresses - like internxt.io, into a series of numbers, that are machine readable, but hard for users to memorize. DNS allows users to browse the internet with ease, and most of the users don’t even think about it. Communication between user and DNS server is done in multiple steps, and it only takes a few milliseconds to complete. But, while DNS is a very good and well technical solution, it has its flaws. DNS is by default decentralized, and it is very massive set of supercomputers running behind it. But on top level, it is centralized, and this has a lot of flaws users usually don’t think of. DNS allows for hacker attacks to be conducted on massive number of users. It allows also for someone who controls it, to censor the internet in the way he likes. Many countries around the world censor the internet already, and since day by day internet is more widely used, the need for this will only grow bigger, by big organizations and governments. This is where a truly decentralized DNS would set in place, and bring its many benefits onto the table. The main benefit is that decentralized DNS would allow zero down times while at the same time making censoring of content impossible.

Our nodes will feature a mapping layer that will be self-organizing. These mapping nodes could be HTTP ingress proxies that route client requests from a given locale to the appropriate data centers, the model adopted by Google and Yahoo. Or the mapping nodes could be authoritative DNS servers that resolve local queries for the names of Web sites, the model adopted by Akamai and most CDNs. Furthermore, these DNS servers may use IP anycast to leverage BGP-based failover and to minimize client request latency. Whatever the mechanism for interacting with clients, each mapping nodes has only a partial view of the global space of clients. Node will be developed based on research in decentralized server selection.

Decentralized Service Discovery

Service-Oriented Computing (SOC) is emerging as a paradigm for developing distributed applications. A critical issue of utilizing SOC is to have a scalable, reliable, and robust service discovery mechanism, that would work perfect in combination with likes of network like Internxt, with a lot of nodes in form of user computers. This will allow for new services that enter network to be discovered the moment they are available to use. Since it is all decentralized, there is no need for update of some central table that everyone reads from, and makes updates visible to users in an instant. However, traditional service discovery methods using centralized registries can easily suffer from problems such as performance bottleneck and vulnerability to failures in large scalable service networks, thus functioning abnormally.

A weak point of web services is the lack of suitable technologies for implementing service registries. Universal Description, Discovery and Integration (UDDI), as a centralized approach, conflicts with the distributed nature of service-oriented architectures and did not prevail in practice. Internxt novel service discovery scheme is scalable. Each service provider runs one or more discovery proxies in its internet domain and sets DNS records pointing to them. Then, local services can register at these proxies. Service users from the local network as well as from the internet can now locate these proxies using the DNS and query them for available services in this domain. Thus, the local proxies are linked together by the DNS forming one coherent global service directory.

Centralized approaches to service discovery, monitoring, and load balancing have several critical design limitations including: single point of failure; lack scalability; high network communication cost (such as network bottleneck, congestion) at links leading to the service; requirement of high computational power (which may be not feasible with commodity machines that public clouds offer) to serve a large number of resource look-up and updated queries on the server running the central service.

Decentralized service discovery is considered as a promising approach to addressing the problems caused by centralized infrastructures. Internxt is solving this, allowing extremely high scalability, data availability and quality of service aware discovery. As new nodes are added to the network, it allows us to scale the services better, and have shortened waiting times when loading content. This allows the user to always download content from the closest node. In perfect scenario, if the user is to have someone in the same local network with him already serving the content, it can be downloaded using local network, and allow the content to be shown instantly, saving on DNS and file download times. Otherwise, the closest and most performant node is selected, and the content is served from there. Performance of node and delivery is calculated based on node load, network speed, latency and other factors that may decrease the overall experience.

Gossip-Style Failure Detection

When building a system on top of a set of wildly uncooperative and unruly computers you have knowledge problems: knowing when other nodes are dead; knowing when nodes become alive; getting information about other nodes so you can make local decisions, like knowing which node should handle a request based on a scheme for assigning nodes to a certain range of users; learning about new configuration data; agreeing on data values; Failure Detection is valuable for system management, replication, load balancing, and other distributed services. To date, Failure Detection Services scale badly in the number of members that are being monitored.

A gossip protocol is a style of computer-to-computer communication protocol inspired by the form of gossip seen in social networks. Modern distributed systems often use gossip protocols to solve problems that might be difficult to solve in other ways, either because the underlying network has an inconvenient structure, is extremely large, or because gossip solutions are the most efficient ones available.

Since this is a key point in scaling networks with high number of nodes, like ours is, we will implement algorithms that prevent content from being unavailable at any moment. The protocol here needs to take into consideration many negative effects, like power outage, internet failing to deliver messages between nodes, etc. We would have this running on each separate node, and on federated nodes, also run by users. Such nodes would need more processing, storage, and network power, and would logically be rewarded with higher number of tokens for their work. Our solution will be based on scientific research and current proven open source solutions.

Decentralized Network Coordinate System

Network tomography is the study of a network's internal characteristics using information derived from end data. Network tomography advocates that it is possible to map the path data takes through the internet by examining information from "edge nodes," the computers in which the data are originated and from which they are requested.

Network coordinates (NC) system is an efficient mechanism for internet distance prediction with limited measurements. The goal of these systems is mainly to develop public infrastructure that provides distance information between any two arbitrary points on the internet. The ability to predict latency without prior communication allows systems to use proximity for better performance with less measurement overhead. A coordinate system could be used to select which of a number of replicated servers to fetch a data item from; such a system is particularly helpful when the number of potential servers is large or the amount of data is small.

While Vivaldi is a representative NC, our solution will be based on research done in Pharos and PCoord papers. Pharos is a fully decentralized NC system. All nodes in Pharos form two levels of overlays, namely base overlay for long link prediction, and local cluster overlay for short link prediction. Vivaldi algorithm is applied to both base overlay and local cluster. As a result, each Pharos node has two sets of coordinates. PCoord is a fully decentralized network coordinate system with each host updating its coordinate iteratively to refine the prediction accuracy of its estimated position. Each host updates its coordinates to minimize a loss function that measures the difference between the actual and the geometric distances between itself and a small set of other hosts. PCoord does not require any special bootstrap process - each node goes through an iterative calibration process to refine its coordinates.

Evaluation of performance done in both papers clearly shows benefits over Vivaldi and with that reason it will be basis of our implementation.

Node Reputation and Security System

The significance of efficient security mechanisms in P2P and Grid systems is unquestionable, since security is a quality of service factor for such systems. Traditional security mechanisms like encryption and sand-boxing incur additional overhead. By using trust and reputation, the additional overhead is minimized. Trust mechanisms result in safer communication between P2P and Grid nodes, increasing the quality of service and making P2P and Grid technology more appealing.

Due to the openness of distributed networks, security issue becomes one of the most important challenges when deploying these networks into application. Traditional strategies, such as traditional encryption and access control, because of their poor scalability, are no longer suited for resolving security issues of distributed P2P system. Trust management resolves the security issues in semantic and behavioral levels and filters malicious nodes based on their real-time behaviors between transactions. Trust mechanism can transfer between heterogeneous mixed networks seamlessly, and the researches of trust management are of considerable interest in recent years. P2P grid computing combines and integrates grid and P2P technologies to implement peer-to-peer communications with greater distribution and scalability. Trust management is a complicated and difficult task in such an environment, because resources are geographically distributed and belong to distinct organizations. In a P2P environment, some peers may provide services with low quality and may not promise to satisfy user requirements. An unfavorable situation arises when providers offer incorrect information about their resources/services to exaggerate the quality of their services. To inspire resource sharing among nodes and protect against malicious node behaviors, a reputation system for trust management is necessary. Peer-to-Peer (P2P) reputation systems are essential to evaluate the trustworthiness of participating peers and to combat the selfish, dishonest, and malicious peer behaviors.

Our solution will be based on recent advances in trusted grids and P2P computing systems. PowerTrust paper will be basis for our solution. In comparison with two established P2P reputation systems, EigenTrust and PeerTrust, we believe PowerTrust is more robust and scalable. Inspired by the power-law findings in peer feedbacks, the PowerTrust system dynamically selects a few power nodes that are most reputable by using a distributed ranking mechanism. The good reputation of power nodes is accumulated from the running history of the system. Like a democratic system, power nodes are dynamically replaceable, if they become less active or demonstrate unacceptable behavior.

Market position

We are providing a completely revolutionary technology that outperforms the current solutions in the market. As it has already been mentioned, our main competitive competences are privacy, security and efficiency, providing a price at least as good as the one traditional solutions provide. We also want to provide a user experience at least as good as the ones they offer. There already exist a few direct competitors to Internxt though, which provide a decentralized cloud solution that aims to improve the way the internet is organize. Although there are not many competitors, their decentralized technologies are sometimes outstanding. Even though this is the case, these competitors have not yet become mainstream. We believe that the main issue resides in user experience. The reality is that, although decentralized and blockchain technology is better than the traditional technology we use, it is too complex for the average user to make a transition from one technology to the other. We are putting the focus on providing an outstanding user interface, that's as easy to use as the current solutions offered by big corporations. By providing a competitive price and a seamless user experience, we believe we can become mainstream with our decentralized technology, which in some cases does not have to be better than the one offered by our direct competitors.

IXT Token

ICO overview

The goal of the ICO is to allow users to participate in the development of our platform. The minimum goal in EUR for the project is €200.000, and the cap is set at €30.000.000. With maximum founding, we would be able to develop the complete environment we had envisioned. With the minimum raised cap of €200k we would be able to develop a basic version of our decentralized file storage technology (dCDN). If we fail to raise the minimum cap, the BTC and ETH will be returned to the corresponding investors wallets. IXT token will be used as a payment method for the services provided in our network. The tokens will be issued according to ERC 20 Ethereum standard. Since the ICO will be possible in both ETH and BTC, users will be assigned IIT tokens (IIT – Internxt ICO token) based on the donation in ETH. For BTC, the number of tokens is calculated by first converting the BTC into ETH, and then the tokens are assigned to the user that made the purchase. 2 weeks after the ICO ends, ICO tokens will be transferable to IXT (Internxt token), at the rate 1:1. Tokens will then be available for the users in their Ethereum wallets, and available on multiple exchanges for users to trade in. In depth guides for buying tokens during ICO, and later exchange of ICO tokens to regular ones, are available on our website, and for any help needed there will be live support during the ICO, and before it starts, though our slack channel. Minimum and maximum amounts of tokens that are sold in the ICO sale can be seen in the table below. The values on the following table are calculated assuming a value of €300 per ETH. The following numbers will be updated 12 hours before the actual ICO starts with the actual price in EUR per ETH. For each ETH collected, there will be 500 IXT tokens generated. The number of tokens created for Internxt's developers and the company is fixed in percentage, and the amount is calculated based on the total tokens created during the ICO sale. These tokens will mainly be used to test the current technology internally. Tokens generated for the team will be "frozen" for 6 months after the ICO sale, since we consider that will be the period that will be needed for the initial technology to be developed and to start getting tested.

Tokens issued per ETH	500
Minimum ether to collect	666.666667 ETH
Maximum ether to collect	100,000 ETH
Tokens generated for Internxt's team	5%
Tokens for Internxt	10%
Date of start	September 7th 2017

End date	September 28th 2017
Maximum tokens generated	57,500,000
Maximum tokens for the team and company	7,500,000

Post ICO budget allocation

Based on the amount of money collected during the ICO, we will be able to develop a bigger or smaller number of features for our technology. In the event that we do not raise the required amount to build all the features we aim to develop, we will create a functional, less complete technology up to the extent that we are able to. Maximum financing scenario would allow us to follow the delivery process without running into insufficient budget. The biggest part of the budget will go into the actual development of the project, with other expenses such as marketing and taxes. We want to go into the market as fast as possible, and expand the value of the tokens for everyone holding these, by giving an actual value to the product. Once the ICO takes place, we will start working in the development of the technology and expect an stable version our technology to be readily available by the end of 2018.

References

Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, Doug Terry, Epidemic algorithms for replicated database maintenance, 1987

Brijender Kahanwal, Tejinder Pal Singh, The Distributed Computing Paradigms: P2P, Grid, Cluster, Cloud, and Jungle, International Journal of Latest Research in Science and Technology ISSN (Online):2278-5299 Vol.1, Issue 2 :Page No.183-187, July-August(2012)

Domenico Talia and Paolo Trunfio, Toward a Synergy Between P2P and Grids

Handbook of Research on P2P and Grid Systems for Service-Oriented Computing: Models, Methodologies and Applications, Nick Antonopoulos (University of Derby, UK)

Ian Foster, Adriana Iamnitchi, On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing, International Workshop on Peer-to-Peer Systems IPTPS 2003: Peer-to-Peer Systems II pp 118-128

Jingpei Wang and Jie Liu, The Comparison of Distributed P2P Trust Models Based on Quantitative Parameters in the File Downloading Scenarios, 2016

Karl Aberer, Philippe Cudré-Mauroux, Anwitaman Datta, Zoran Despotovic, Manfred Hauswirth, Magdalena Punceva, Roman Schmidt, P-Grid: A Self-organizing Structured P2P System

Li-wei H. Lehman, PCoord: A Decentralized Network Coordinate System for internet Distance Prediction

Michael Factor, Kalman Meth, Dalit Naor, Ohad Rodeh, Julian Satran, Object Storage: The Future Building Block for Storage Systems, IBM Haifa Research Laboratories

Nilton Bila, Paolo Dettori, Ali Kanso, Yuji Watanabe, Alaa Youssef, Leveraging the Serverless Architecture for Securing Linux Containers

Qiang He, Jun Yan, Yun Yang, Ryszard Kowalczyk, Hai Jin, A decentralized service discovery approach on peer-to-peer network, 2013

Rajiv Ranjan, Liang Zhao, Xiaomin Wu, and Anna Liu, Peer-to-Peer Cloud Provisioning: Service Discovery and Load-Balancing, 2010

Red Hat, Ansible in depth

Red Hat, The Benefits of Agentless Architecture, Ansible

Robbert van Renesse, Yaron Minsky, and Mark Hayden, A Gossip-Style Failure Detection Service

Runfang Zhou, Member, IEEE, and Kai Hwang, PowerTrust: A Robust and Scalable Reputation System for Trusted Peer-to-Peer Computing

Russ Cox, Frank Dabek, Frans Kaashoek, Jinyang Li, Robert Morris, Practical, Distributed Network Coordinates

Saeed Javanmardia, Mohammad Shojafar, Shahdad Shariatmadari and Sima S. Ahrabi, FR TRUST: A Fuzzy Reputation Based Model for Trust Management in Semantic P2P Grids

Sergio Marti, Trust and reputation in Peer-to-Peer networks, Stanford university Phd dissertation, 2005.

Sitaram Iyer, Antony Rowstron, Peter Druschel, Squirrel: A decentralized peer-to-peer web cache, 2002

V.Deepika , G. Mohana Priya , R.Gayathri , A.Chitra, An Efficient Content Delivery Scheme Using a Decentralized Content Caching, 2017

World Wide Technology Inc, Issues in Cloud Security, 2016.

Yang Chen, Yongqiang Xiong, Xiaohui Shi, Beixing Deng, Xing Li, Pharos: A Decentralized and Hierarchical Network Coordinate System for internet Distance Prediction