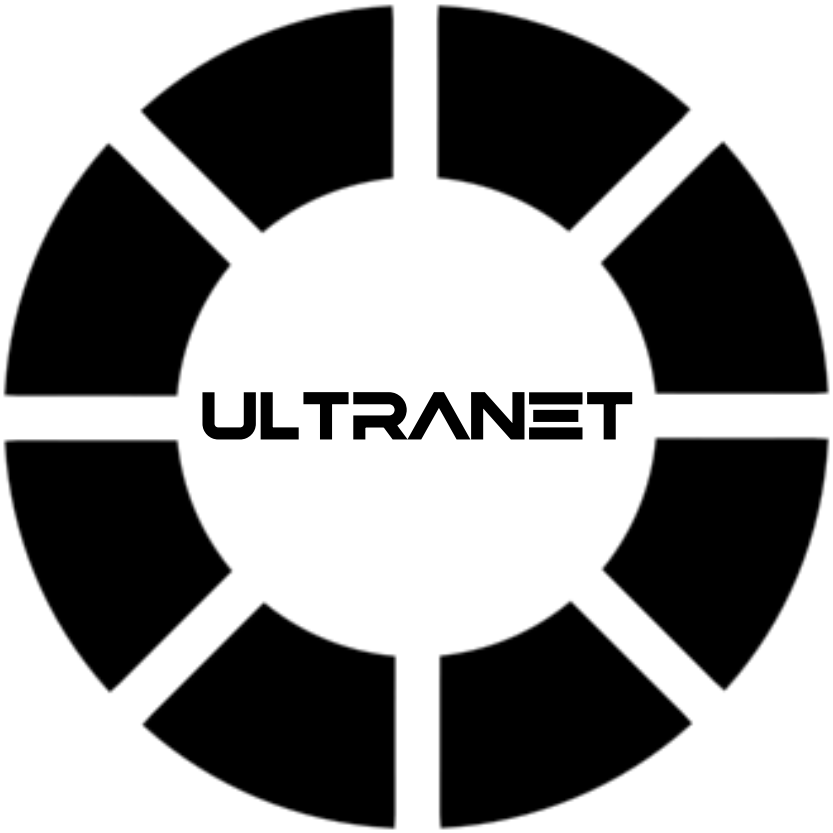


Technical White Paper

v. 0.2

ultranet.org
twitter.com/ultranetorg
youtube.com/channel/UCauxo1yuwGkr_D4N1DxaYUg

Abstract: Ultranet introduces an open decentralized infrastructure for a new generation of Internet applications that combines the advantages of both Web and local platforms complemented by the decentralization and trust provided by peer-to-peer and blockchain technologies. The technology of Ultranet takes the high performance, rich UI capabilities, and access to local resources typical of local platforms, while inheriting its platform-independent network nature, safety, and ease of use from the Web platform. Peer-to-peer and blockchain technologies are in turn used for application distribution and for protecting the whole ecosystem from malicious software and other threats.

The technology consists of five major components:

- Free Distribution Network – FDN
- Distribution Management System – DMS
- Anti-Malware Protection Network – APN
- Unified Operating Superstructure – UOS
- Unified User Interaction Environment – UUIE

Peer-to-peer and blockchain components revolutionize the way in which applications and websites are delivered to users. The existing InterPlanetary File System (IPFS) global network is used by Ultranet for a decentralized software distribution. Third-party smart-contract platforms, such as Ethereum, are used to manage distribution and may be considered as a hybrid of a decentralized application store and a domain name system. Another blockchain network is used for anti-malware protection purposes acting as a decentralized certification service and signaling network.

The UOS component allows developers to create cross-platform applications that are run as easily as the opening of a Web page, but with the full power of a local system and without any of the issues typical of today's cross-platform development. Many and various Web applications can utilize the power of a desktop-grade UI to provide a rich and fast user-interface experience that would be unattainable with classic HTML-based technologies. Ultranet applications run under the UUIE are not limited by classic desktop UIs, but are also capable of rendering any 3D visual interfaces, which is important for support existing and future VR gears.

This platform does not adhere to a particular OS, and once it is accepted worldwide, it will render a wide variety of existing operating systems unnecessary. Many years ago, platforms such as Windows and macOS brought us multi-tasking and graphical UI, but nothing new has happened since then. An overlooked shortcoming of existing OSs also involves privacy, in that all major proprietary OSs spy on their users. Thankfully, those days are now over. Initially launched as a cross-platform technology, Ultranet may finally require only one free, secure, open OS, such as Linux, thus ending the reign of big corporations in this sector of the economy.

Disclaimer: *This Technical White Paper is for information purposes only. Ultranet Organization does not guarantee the accuracy of, or the conclusions reached in, this white paper, and this white paper is provided "as is". Ultranet Organization does not make, and expressly disclaims, all representations and warranties, whether express, implied, statutory or otherwise, whatsoever, including, but not limited to: (i) warranties of merchantability, fitness for a particular purpose, suitability, usage, title, or non-infringement; (ii) that the contents of this white paper are free from error; and (iii) that such contents will not infringe third-party rights. Ultranet Organization and its affiliates shall have no liability for damages of any kind arising out of the use of, reference to, or reliance on this white paper or any of the content contained herein, even if advised of the possibility of such damages. In no event will Ultranet Organization or its affiliates be liable to any person or entity for any damages, losses, liabilities, costs, or expenses of any kind, whether direct or indirect, consequential, compensatory, incidental, actual, exemplary, punitive, or special for the use of, reference to, or reliance on this white paper or any of the content contained herein, including, without limitation, any loss of business, revenues, profits, data, use, goodwill, or other intangible losses.*

Contents

Glossary

HTML – Hypertext Markup Language: the standard markup language for creating Web pages and Web applications.

HTTP – Hypertext Transfer Protocol: an application protocol for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web, where hypertext documents include hyperlinks to other resources that the user can easily access.

GUI – Graphical user interface: a form of user interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notation.

OS – Operating system: system software that manages computer hardware and software resources and provides common services for computer programs.

Dapp – Decentralized application (dapp, Dapp, dApp, or DApp): a computer application that runs on a distributed computing system. DApps have been mostly popularized by distributed ledger technologies (DLTs), specifically Ethereum Blockchain, where dApps are often referred to as smart contracts.

IPFS – InterPlanetary File System: a protocol and network designed to create a content-addressable, peer-to-peer method of storing and sharing hypermedia in a distributed file system.

VR – Virtual Reality: an experience taking place within simulated and immersive environments that can be similar to, or completely different from, the real world.

Blockchain – a growing list of records, called blocks, which are linked using cryptography.

PoS – Proof of Stake: a type of consensus algorithm by which a cryptocurrency blockchain network aims to achieve distributed consensus. In PoS-based cryptocurrencies, the creator of the next block is chosen via various combinations of random selection and wealth or age (i.e., the stake).

ICO – Initial coin offering, or initial currency offering: a type of funding using cryptocurrencies. The process is mostly accomplished by crowdfunding.

DDOS – Distributed Denial of Service: a cyber-attack in which the perpetrator seeks to make a machine or network resource unavailable to its intended users by temporarily or indefinitely disrupting the services of a host connected to the Internet. In a DDOS attack, the incoming traffic flooding the victim originates from many different sources. This effectively makes it impossible to stop the attack simply by blocking a single source.

The Problems of the Web

Every site on the Web uses HTML to generate its pages. HTML is a static text markup of page elements. It was developed as a language that allows the creation of rich-text documents. Rich-text means a text document that can also contain various media elements such as images, animations, links, tables, and other items. JavaScript language is used to bring some dynamics to the static nature of HTML pages. Finally, the language of Cascading Style Sheets (CSS) makes Web design and markup considerably easier.

These are three major technologies that all websites use, and these technologies are perfect for sites that represent themselves mainly as linked rich-text documents. An additional benefit of text-based Web technology is the capability of being intrinsically indexable by search engines. As the Web has developed and grown, though, something more flexible and faster than HTML has become a critical requirement for many websites. If we take a close look at these websites, it becomes clear that they are in fact more applications than sites, and that their requirements differ from those of websites *per se*. They need a complex and responsive GUI, they need to work at the highest possible level of performance, and they do not need to be indexable by search engines.

Here are some examples of these Web applications:

- Social networks and other sites that provide their content for authenticated users only
- Corporate, public, and other portals
- Video, photo, and other media hosting
- Site admin. and personal sections
- Email clients
- Games
- Development tools, task managers, and source controls
- Exchanges, online banking, accounting, and POS
- Services control panels such as Web hosting panels
- Online office tools
- Maps
- Streaming services
- Various online tools such as converters and editors
- Many, many others.

Several technologies provide solutions, the best-known of which is Flash. While Flash allows the creation of things that are impossible under pure HTML and JavaScript, this is a proprietary product, and still something alien to the Web. The Internet still needs some technology that will deliver the power and flexibility needed for Web applications, but security requirements hinder progress in this area. Existing Web technologies do not allow the running of native code and by design do not provide access to the resources of the local OS. These restrictions help to prevent almost all kinds of malicious activity by website code, and they make Web surfing much safer than installing software locally.

Another long-standing problem with the Internet is its server-centric architecture. Every website and every Web request is processed by Web servers. Because of this, they often suffer from heavy loads, but more importantly, this makes them ideal targets for censorship, DDOS, and hacker attacks. Decentralized Application Platforms is a promising new technology that has the potential to eliminate the need for Web servers by transforming the current Internet architecture into a homogeneous peer-to-peer network. However, it can only replace a data layer and still has to rely on conventional infrastructure for application and UI delivery.

The Problems with Local Platforms

By the term *local platforms,* we mean locally installed desktop or mobile operating systems such as Windows, Linux, MacOS, iOS, Android, and the like. *Local applications*, in turn, are software programs installed and run on local platforms. Software vendors want their products to run on as many platforms as possible. However, cross-platform software development is difficult, as platform differences and the look and feel of different GUIs frequently create a dilemma:

EITHER

a quick and cheap development using some cross-platform framework with bad user experience,

OR

a long-term and expensive mostly separate development for each platform with the best user experience.

On the other hand, though, are there really any significant differences between the local operating systems available on the market? After all, all of the systems come with a very similar set of functions: multitasking, 2D-window GUI, security, networking, and some others; so it is clear that cross-platform development difficulties caused by inter-platform differences are not inevitable. If we could introduce a standard for cross-platform APIs and a unified OS-independent GUI standard, then we would be free from problems caused by inter-OS specifics and differences.

Another problem with current platforms is that their GUIs do not fit properly on existing and future 3D-display devices, and it would be nearly impossible to effectively adapt legacy 2D-window user interfaces to the 3D environment provided by 3D gears. Furthermore, the classic GUI has some problems with utilizing the screen space of wide and ultra-wide display monitors: Maximizing makes windows too big, and finding a good solution for this will very likely require changing an application’s code.

Existing solutions

There are several existing technologies that address the major issues of local and Web platforms. Table 1 below gives a comparison of some of their pros and cons.

Table 1. Pros and Cons of Existing Solutions

	HTML5	Flash (Adobe)	AIR (Adobe)	ActiveX (Microsoft)	Silverlight (Microsoft)
Industry standard	Yes	No	No	No	No
Open source	Yes	No	No	No	No
Cross-platform support	Yes	Partial	Partial	No	No
Local system API	No	No	Partial	Yes	Partial
UI integration (mostly)	Inside a browser	Inside a browser	Local OS UI	Inside a browser	Inside a browser
Performance	Low	Moderate	Moderate	High	Moderate
Maximum acceptable level of UI complexity	Low	Moderate	Moderate	Moderate	Moderate
Malware resistance	High	Low	High	Low	High
VR support (3D UI Environment)	No	No	No	No	No
Deprecated	No	Yes for browsers	No	Yes	Yes

Requirements

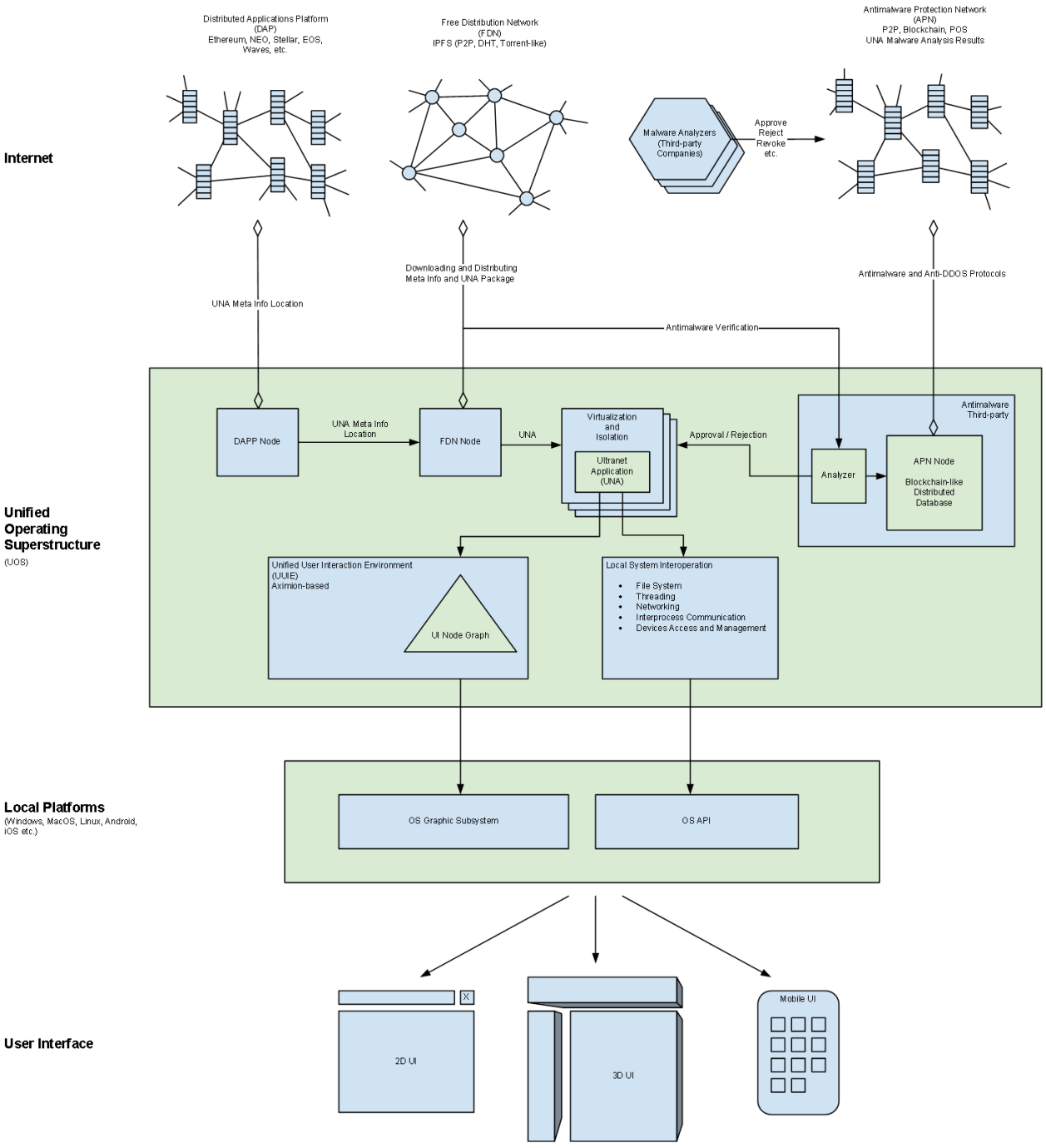
Given the problems described above, let’s summarize all the requirements of a new technology:

- Open platform to avoid any patent infringement cases; profit-oriented evolution strategy; and promotion of proprietary software
- Completely free for users; free or as cheap as possible for publishers
- Maximum decentralization
- An infrastructure for free software distribution with high throughput and the highest possible resistance to censorship
- A platform that allows the building of OS-independent applications that are accessible in a similar way to Web pages using well-known URL navigation, but appear and function like conventional local applications. Their lifecycle should not include installation and uninstallation stages.
- The possibility of replacing application-like HTML websites with a new technology that brings the full power of local applications.
- Minimal impact of malicious code on infrastructure, local system, and user data
- Platform independence. Make the choice of local OS unimportant for users, so users will not need to choose between Windows, MacOS, and Linux anymore.
- A unified UI that is able to adapt a single platform-independent user environment to desktop, mobile, and VR environments, and to leverage the full power of local hardware. It should support ordinary and ultra-wide monitors, double-screen laptops, mobile devices, and modern and future VR gears.

We hereby propose the technology that meets all these requirements: **Ultranet**, the next step in the Internet evolution.

Figure 1. The Infrastructure

(click to enlarge)



The following are the major components of Ultranet technology:

FDN: Free Distribution Network – provides free and decentralized distribution of Ultranet applications and implemented on top of an IPFS global swarm.

DMS: Distribution Management System – a smart-contract platform that is used to host UNA meta-info management Dapps. The primary purpose of this program is to get meta-info from the FDN using a single globally unique address of a UNA.

APN: Antimalware Protection Network – consists of several components that minimize the impact of malicious code on the whole Ultranet ecosystem. It is a distributed blockchain database of approval, rejection, and other records, and also a real-time signaling network for the rapid broadcasting of recently identified threats.

UOS: Unified Operation Superstructure – a cross-platform local system layer that provides a safe execution environment and exposes a standard API for Ultranet applications.

UUIE: Unified User Interaction Environment – evolved from the Aximion project. Aximion is a concept of unified UI that is able to adapt a single platform-independent user environment to desktop, mobile, and VR environments, and to leverage the full power of local hardware. All implementations must strictly comply with a special technical standard to guarantee the same look and feel across all platforms.

UNA: Ultranet Applications – a new class of platform-independent Internet applications that are distributed via FDN, run under UOS, and must comply with the UNA Open Standard whose specifications cover API, building, packaging, and deployment of Ultranet applications.

Below are some additional benefits that come with Ultranet:

- It's completely free to use for all users. Only publishers have to pay a small fee, and this only for increasing the level of trust in their products
- Utilizing various Dapp platforms as the application server-side (back-end) and Ultranet as the application client-side (front-end) gives rise to an ecosystem that is invulnerable to DDOS attacks. Moreover, Ultranet proposes a special protocol that is capable of actively shutting down any botnet that participates in such types of attack.
- There is the potential to create applications that work on various currently incompatible Linux distributions, which would give a new lease of life to the open-source community.
- It's not possible to ban a particular application or its distribution as it does not depend on a specific URL or IP address. Even publishers (without special measures) have no power to prevent users from using older versions if users are not satisfied with the latest ones.
- The infrastructure does not require high performance Dapp and blockchain platforms, as all existing solutions are already powerful enough for all Ultranet functions.
- Each running application can use its own custom data transfer protocol to communicate with the server side, making network traffic-sniffing much harder.
- Utilizing antimalware companies' *ad hoc* infrastructure makes it possible to perform a much deeper and more comprehensive analysis of application code.

Free Distribution Network

FDN allows participation in the distribution of applications and is implemented using Distributed Hash Table (DHT) technology on top of IPFS. Anyone can publish applications, and anyone can download and seed the applications. No censorship is possible. Not only can stand-alone applications be distributed, but also shared component dependencies. Downloading of these components and the main application is done automatically.

To gain additional flexibility, UNA meta-info is used as a mediator to provide the following useful features:

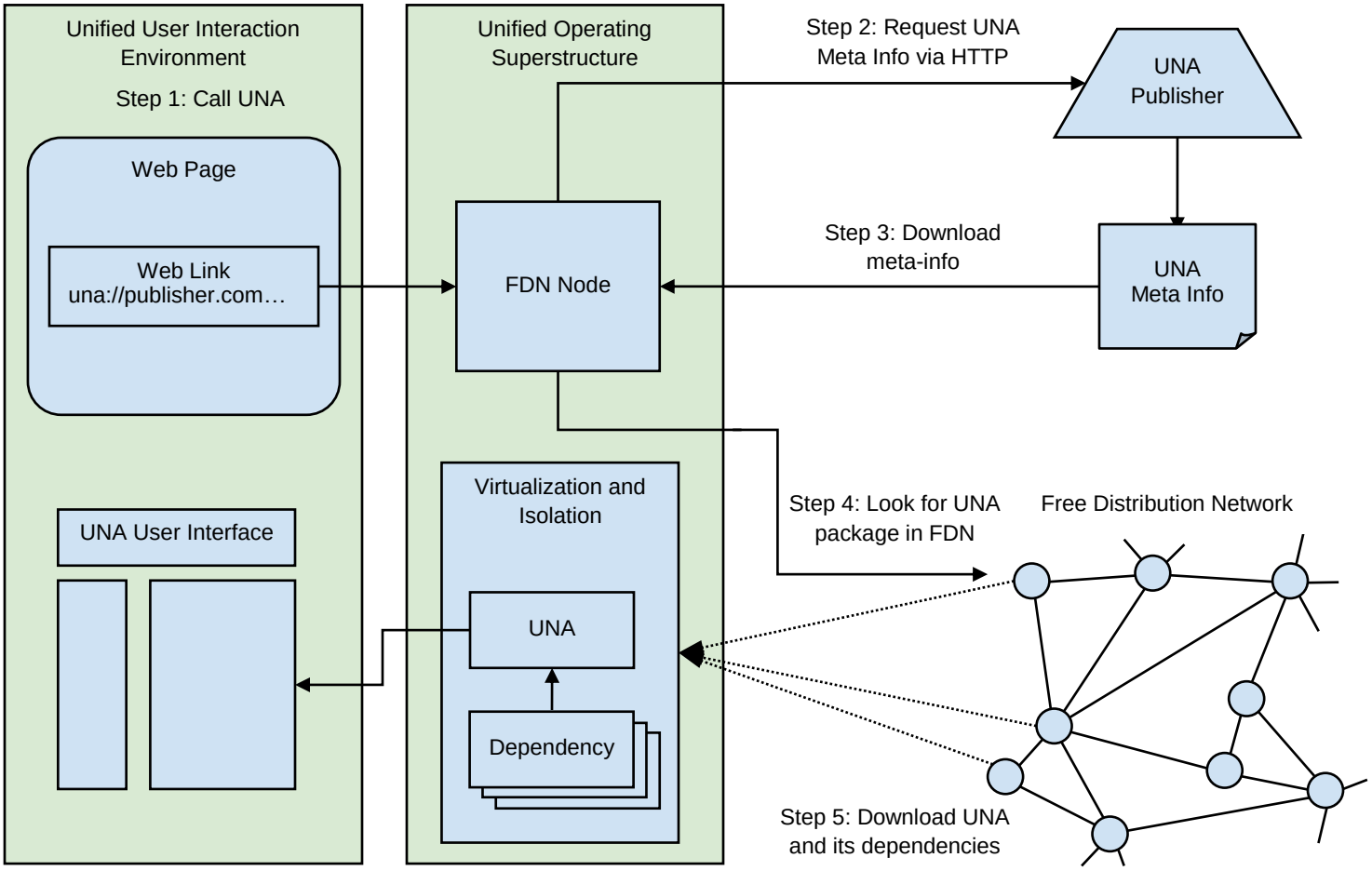
- A basic description of the UNA to be retrieved (title, company, version)
- A list of application files and resources (as hashes)
- Dependencies (as hashes) required for running the UNA
- A visual stub. This may be an image, or even a very simple 3D model, which is displayed in the UI environment during a download of UNA components.
- The publisher's digital signature.

There are two ways to publish UNA meta-info. The first is by using a publisher Web server. The second is by using distributed applications running under an existing Dapp network (Ethereum by default). This is a fully decentralized approach with many benefits.

In the first case, when the user requests to run a UNA, the system performs the following steps:

1. The user clicks on the link.
2. If the link protocol identifier is "una", then the system calls the UNA's URL handler.
3. The UNA URL handler makes an HTTP GET request using a specified URL path to a UNA publishing server.
4. In response to this request, the server returns the UNA's meta-info in a standard format.
5. The system reads the meta-info and shows a visual stub in the UI environment with notification of the download's progress.
   Simultaneously, the system reads the application and its dependencies' hashes and sends requests to the Free Distribution Network, looking for peers to download the UNA and shared components from.
   Simultaneously, the system checks all hashes for anti-malware approvals in the APN database (explained in the next section).
6. Once the application download is complete, its integrity and high approval level are either confirmed or manually overridden by the user. It then runs in default configuration in a virtualized and isolated execution environment.

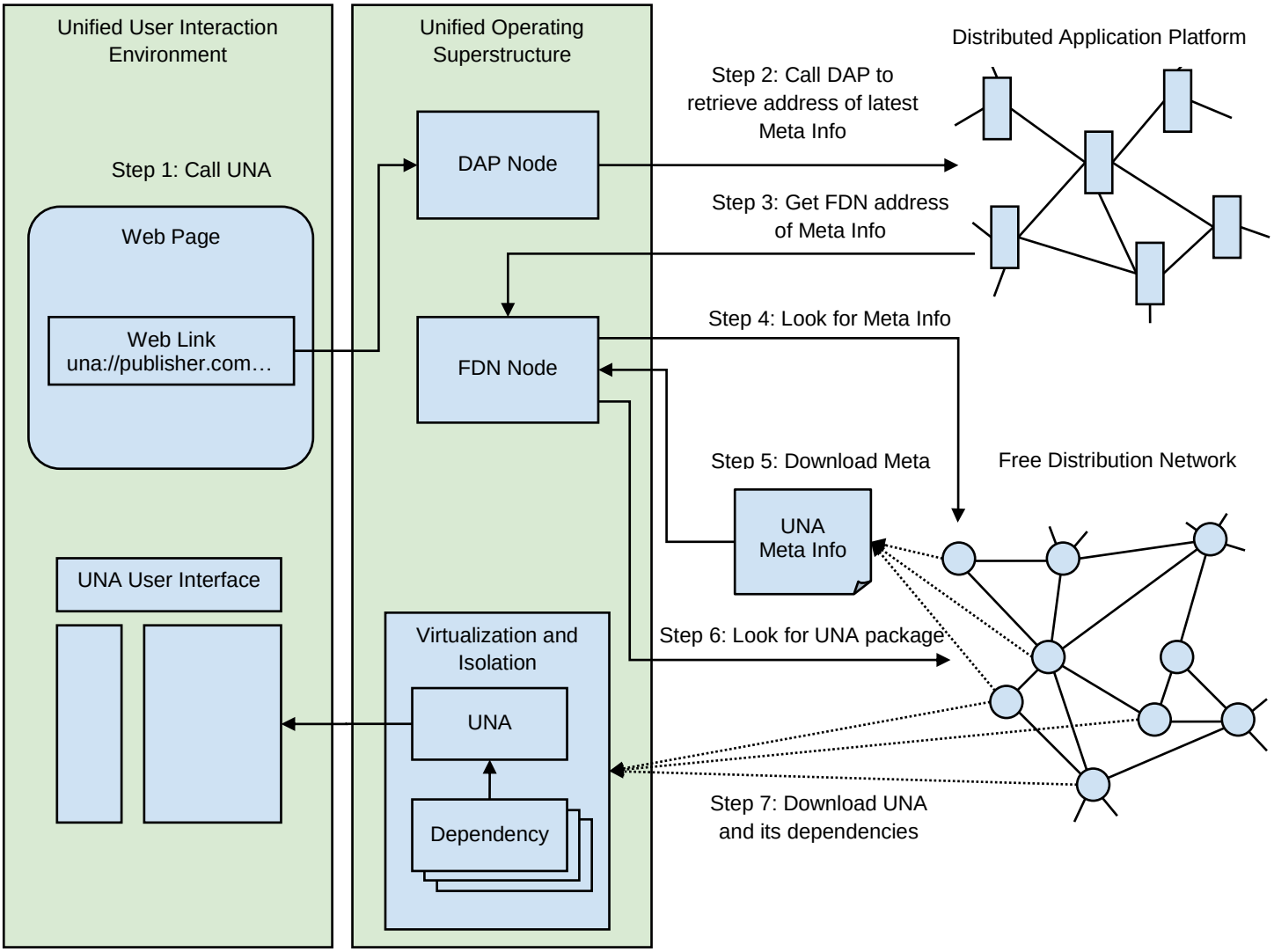
Figure 2. Server-based Publishing / Decentralized Distribution



In the case of a decentralized approach, when the user requests to run a UNA, the system performs the following steps:

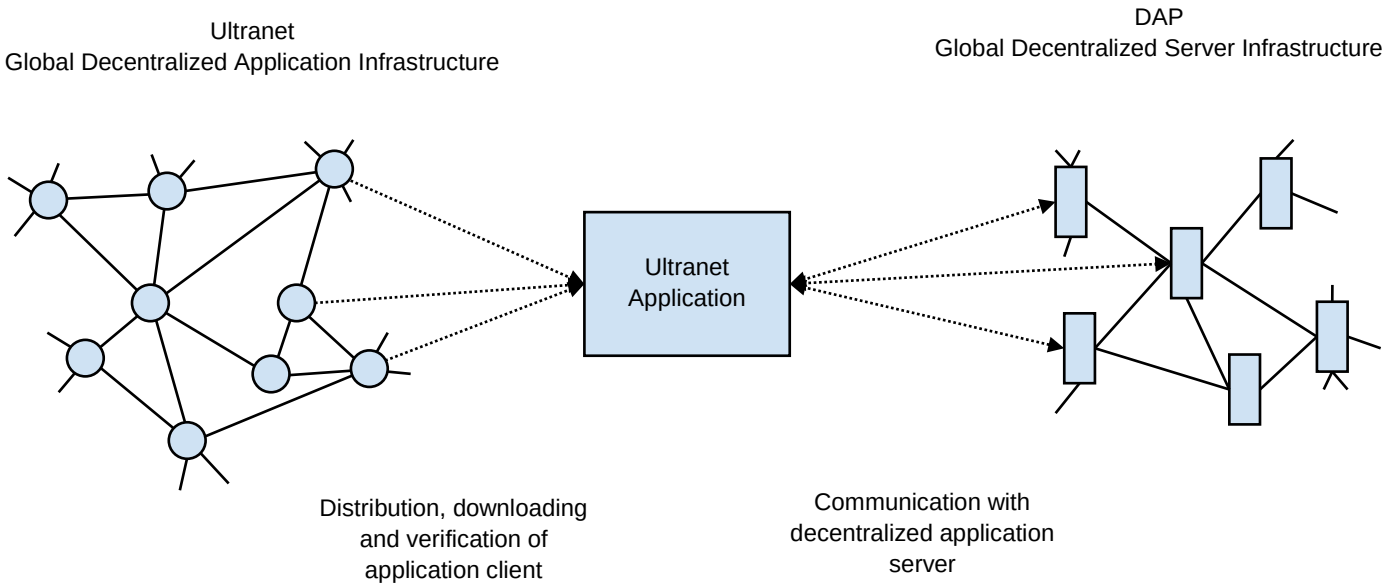
1. The user clicks on the link.
2. If the link protocol identifier is “una” and the URL is a DMS address, then UOS uses a DMS node to obtain the meta-info address.
3. The DMS node call uses a special method of decentralized application to retrieve the FDN address of the latest (or specific) meta-info.
4. With the meta-info address retrieved, the system can find the meta-info file in the FDN.
5. Following downloading of the meta-info, the remaining steps are the same as for the first approach.

Figure 3. Decentralized both Publishing and Distribution



Together with decentralized application platforms, which would act as the back-end for the UNA, this approach revolutionizes the way of how the Internet works: There would be no vulnerable centralized nodes, like Web servers, anymore.

Figure 4. Total Decentralization of the Web



This results in the impossibility of DDOS attacks and censorship, as none of the components rely on IP addresses or DNS records, and so there are no physical targets to attack or block. The scaling is also not an issue for the front end, because the FDN component of Ultranet is built on top of DHT (IPFS) technology that works in a similar way to the Torrent networks, which in turn are purposely designed to handle load at any scale. As for server infrastructure, existing blockchain-based DMS platforms have already shown their potential to bear heavy loads, and there is a strong expectation of major progress in this area in the very near future.

The technology provides an embedded mechanism to update applications so that no special development efforts are required to support auto-updating.

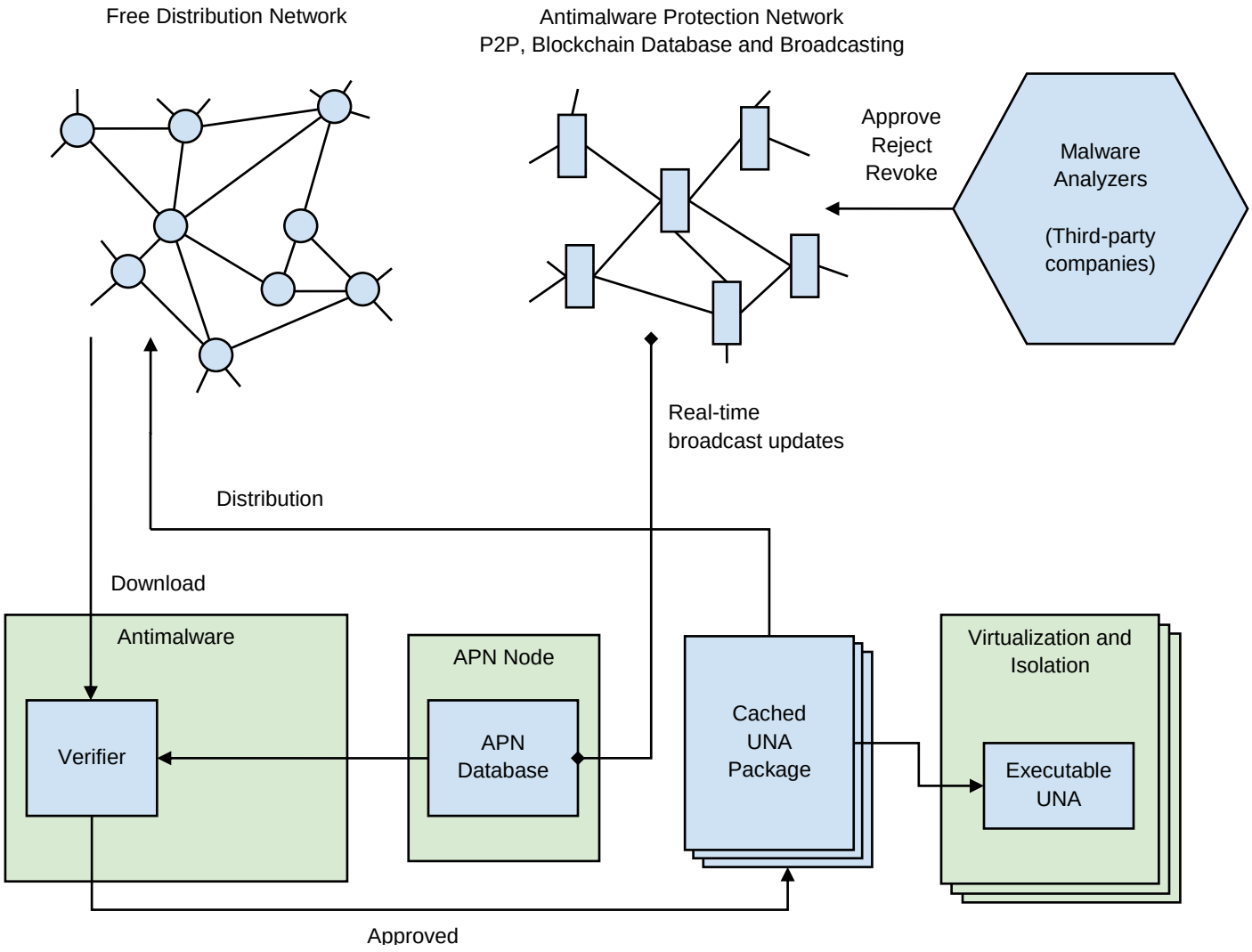
Together with cloud data storage methods (private, IPFS, Sia, Swarm, Storj, or other), the technology enables semi-thin clients. This means that both the application and its data are stored remotely in different distributed networks or clouds. Once a user loads his profile, all the required applications and data are downloaded to a local device for a follow-up.

Antimalware Protection Network

The APN carries out a fast check of just-requested and cached UNAs for malicious code. It is a distributed database of malicious analysis records and a real-time signaling network for the rapid broadcasting of recently identified treats. The distributed database is built as a blockchain registry with a hybrid Proof-of-Stake consensus mechanism.

In this database, each UNA is identified via its hash. A presence of approvals in the APN increases the level of trust for that UNA. The more records are issued for a particular application, the higher the trust level the application has. If no records are present, an unknown trust level occurs. This, in turn, means there is no pre-verification performed and that use of the application is risky.

Figure 5. Publishing, Analysis, Verification



In order to gain approval, the UNA publisher requests a Malware Analyzer (MA) to check an application for malicious code, and if nothing suspicious is found then the MA creates a record with an application identifier (hash), signs it with a company signature, and initiates broadcasting. The signature ensures that the received broadcasted record is not spam and that it is issued by the known MA and stored in the distributed database. The list of trusted anti-malware vendors may vary depending on particular requirements. Ultranet Organization always provides the default list of certified analyzers.

If the previously approved application is later found to be malicious then the MA can issue an approval revocation record. This type of record revokes previously issued approvals for this application, and immediate broadcasting is initiated to let other nodes know about that change.

The antimalware component is standard or third-party software that acts as a regular antivirus and APN node. It searches for MA approvals each time a new application is downloaded or updated, and participates in record broadcasting.

In the future, due to expected progress in AI, this network may transform into a distributed AI built as a decentralized application using Ethereum, NEO, EOS, or some other appropriate technology. This AI would collect and process analytics data from all network nodes and produce application ratings that in turn can be used to allow or prevent execution of the application.

UNS and UNC Tokens

The general philosophy and key principles are as follows:

- The technology requires initial investments.
- The investors should benefit from their investments.
- The investors should have an interest in the development and support of the technology infrastructure.
- The cost of the approval process should be as low as possible to make the technology as affordable as possible for all UNA publishers.
- The future success of the technology should reward the investors without affecting affordability for the publishers.
- The publishing affordability should not depend on market speculations.
- Ultranet Organization should have a certain level of income to support further research and development.

To comply with all of the above criteria, we have introduced two different tokens: UNS and UNC.

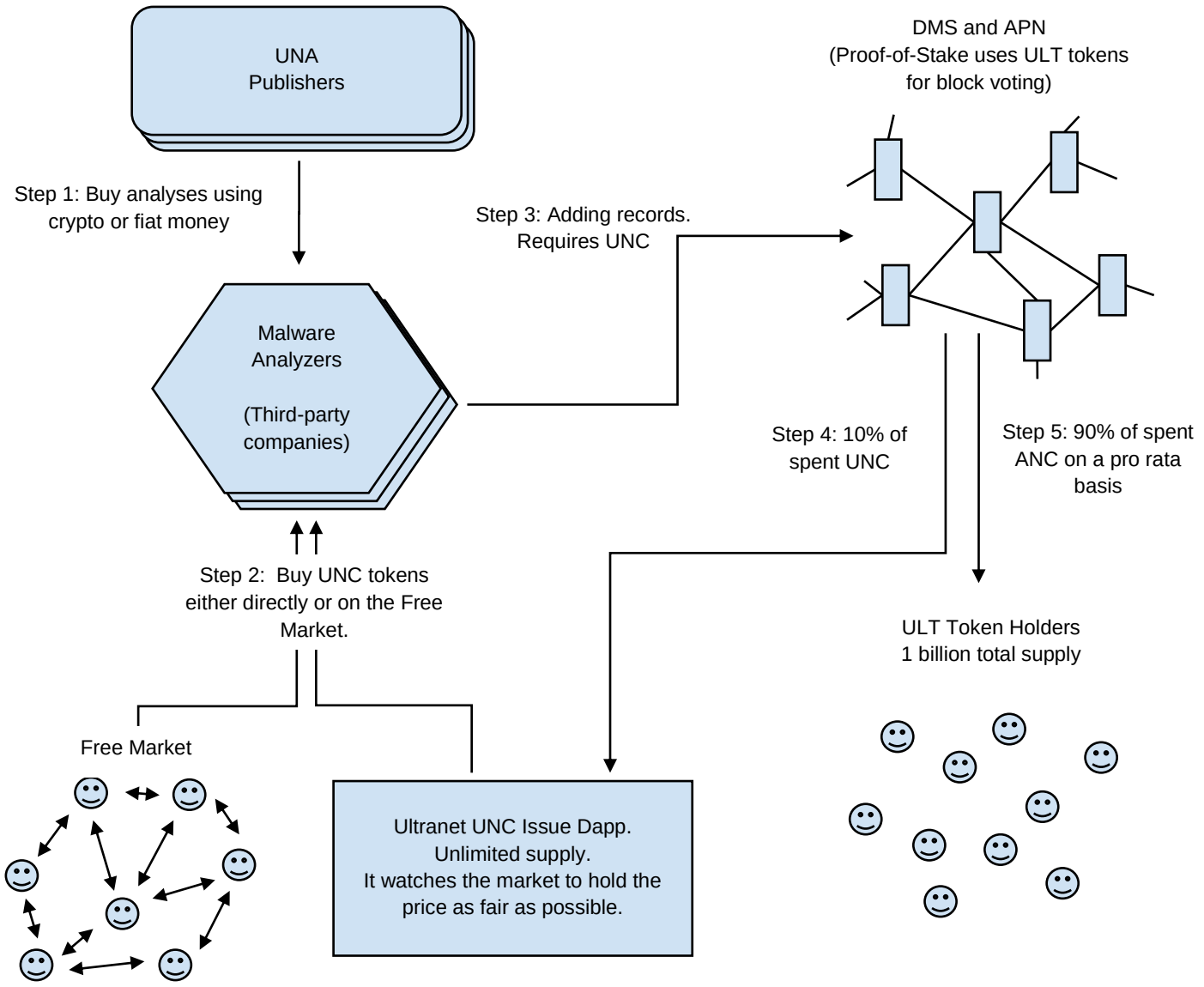
UNC tokens allow publishers and MA's to add records to the DMS and APN blockchain databases. These tokens can be purchased either on the free market or directly from Ultranet Organization. Ultranet UNC Issuer is implemented as a Dapp to ensure as high as possible availability. To make the approval process affordable for any publisher, an UNC token has an unlimited supply, and the Ultranet Organization watches the market to hold its price as fair as possible.

ULS is an infrastructure-supporting and governance token. It is to be distributed via a token sale and will have a maximum supply of 1 billion tokens. When the ICO is finished, this token will be purchasable only on the free market.

Let's look at the tokens' lifecycle on an example of UNA approval process:

The process of UNA approval begins when the publisher requests an MA to verify its application and add approval to the APN database if no threats are found. The publisher pays some amount of crypto or fiat money to analyzers for that service. Analyzers use funds to purchase UNC tokens either on the free market or directly from Ultranet Organization. After purchasing tokens, analyzers can use it to add records in the ratio of one token to one record. Following the creation of a block of records in the APN blockchain, 10% of the collected UNC tokens are sent to Ultranet Organization's account to support further development and to enable some deflation of token supply so as to not allow the price to fall too much. Another 90% is distributed among accounts with locked UNS tokens on a pro-rata basis so that all Ultranet supporters have a continuous income proportional to the amount of UNS they have locked for participation in the proof-of-stake consensus.

Figure 6. UNS and UNC Tokens Lifecycle



Analysis wholesalers help to obtain approvals from many analyzers with only a single request so that the publishers don't need to deal with each analyzer company individually.

Ultranet Organization is unable to discriminate against any publisher as it only sells anonymous tokens, and any entity can purchase them whenever needed. An alternative transaction approach of a single token could lead to a PoS problem of permanent nobility. However, with our separated tokens approach, participation in a block confirmation does not change the proportions of stakeholder deposits: i.e., big deposits do not get bigger with time. The distribution and circulation of UNC are implemented via a Dapp developed and governed by Ultranet Organization. The APN nodes' owners have the power to reject any protocol changes by not switching to a new version of APN Dapp if it disrupts the principles of decentralization.

Unified Operating Superstructure

The UOS provides safe execution and exposes the local system API for Ultranet Applications. A virtualized and isolated execution environment is another measure to minimize the impact of malicious software or incorrect behaviour of a running application. This security measure is similar to how applications work on iOS and Android. In this environment, each application could be in one of the following three isolation states:

- Isolated: Execution is allowed within a virtual environment
- System: Execution is allowed, and virtualization is disabled so the application can access any resources it needs – even the data of other applications
- Blocked: A low trust level is received from the APN; or antimalware found its code suspicious and had this confirmed by the user, so prevented the application from executing.

The isolated application has access only to the virtual file system. It has a single root, and other various sources can be mounted to it. At the OS level, the file structure will be organized in the following way:

Local
Global
SystemLocal
    Application [N]
SystemGlobal
    Application [N]

The Global and SystemGlobal directories are used to store personal user data such as documents, and this data may be synchronized with external cloud or distributed storage networks.

The SystemLocal and SystemGlobal directories may contain special files that are used to make application-specific changes to a local or global configuration using the file merger method.

The Unified Operating Superstructure provides access to a host OS via an open-standard cross-platform API framework. Both native (C/C++) and LLVM APIs are provided. Particular LLVM implementation is an extremely important task and subject to further research. CIL(MSIL) and WASM are major candidates.

UNA developers can choose the type of code they want to distribute. The best way is to publish cross-platform code in the form of LLVM binaries. However, if a developer wants to provide the highest performance, then s/he can create an application using a cross-platform C/C++ approach and publish binaries for all major platforms. In the latter case, when the application is requested to run, appropriate binaries will be downloaded and used according to the type of local platform.

There is no longer is an installation process. To speed up the loading of the UNA, the process is broken down into three stages:

Step 1: When the application is initially requested, the core and minimum required set of components are downloaded, and the default configuration is used to launch the UNA.

Step 2: After the application is started, standard components are downloaded in the background.

Step 3: Finally, when the user needs some extensions, the UNA can download and use them separately.

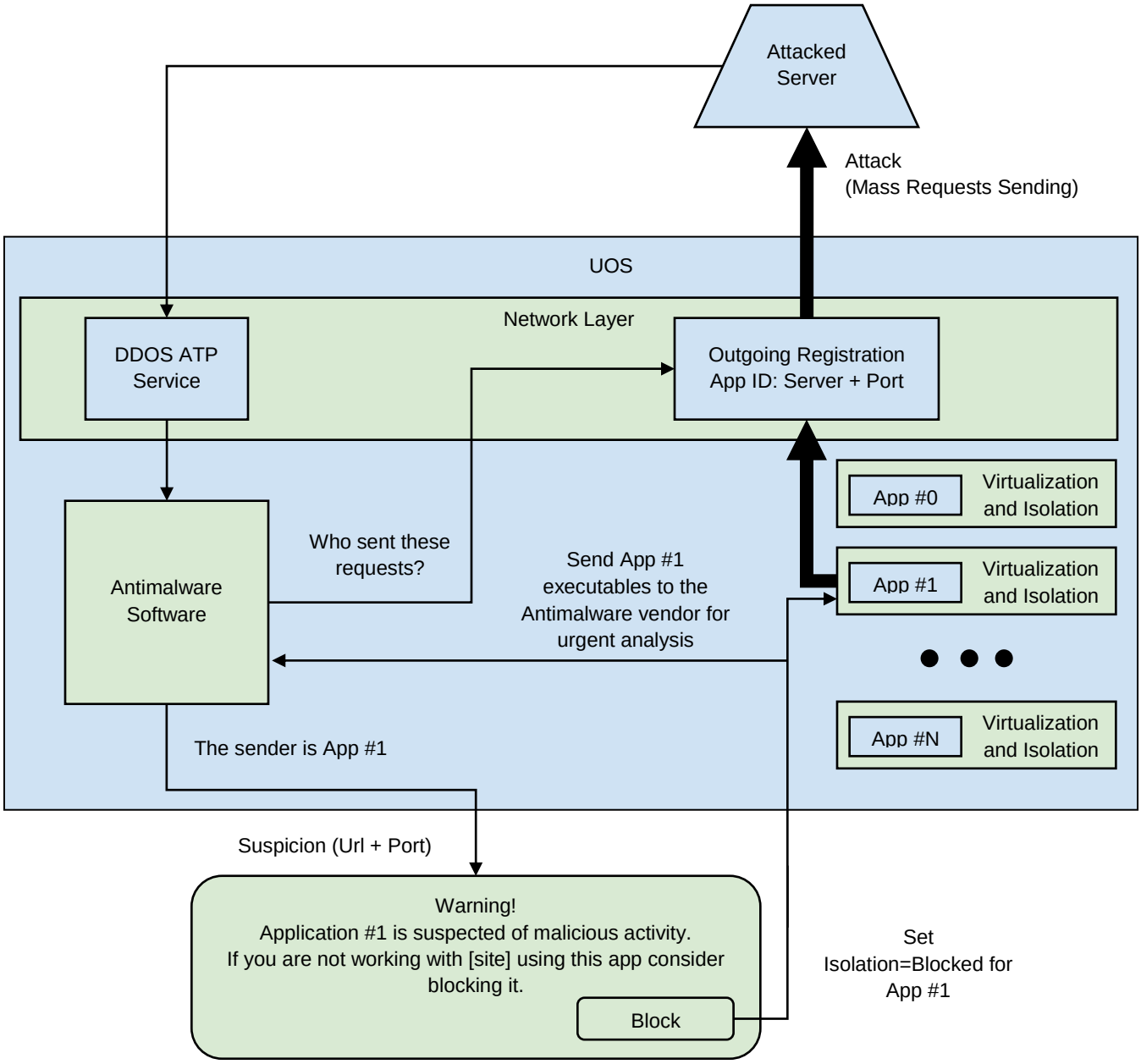
So all this means that the UNA implements a component design in a slightly different way to traditional local applications.

Distributed Denial of Service Active Termination Protocol (DDOS ATP)

The UOS provides the functionality to actively react when a local system is infected and participates in a DDOS attack. This process requires a server side to support this mechanism.

In the case of a DDOS, the attacked server analyzes incoming traffic and sends a special signal (SUSPICION) back to each node that is sending too many requests. On the local side, the UOS stores application-to-destination mapping information and can use it to identify the suspicious application, warn the user about its malicious activity, and then block it if needed.

Figure 7. DDOS Active Termination Protocol



Once the UOS identifies an infected application, it then asks antimalware to send it to its vendor for urgent analysis. Various anti-malware companies then identify malicious code signatures, quickly update their virus databases, and add rejection records to the APN database, or revoke approvals if they were issued previously.

With this technology, an attacked server has the ammunition to not only to stop a DDOS attack but to shut down a whole botnet permanently.

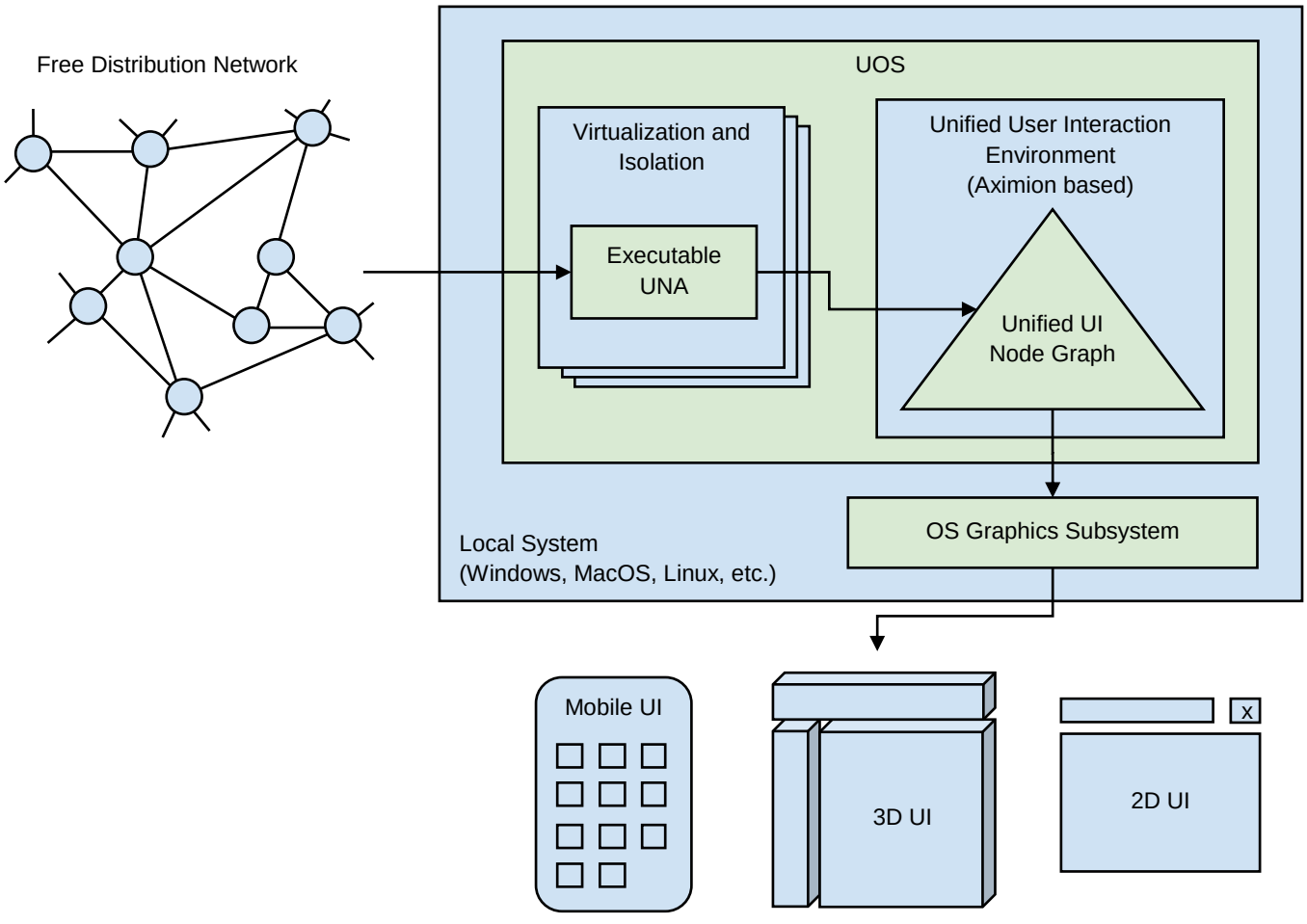
Unified User Interaction Environment

The major ideas behind the UUIE are as follows:

- The user does not run programs. Instead, s/he opens applications from the Internet and builds a personal environment.
- The environment is independent of any particular physical device.
- The technology can render any user interfaces, not only regular windows or Web pages but also any possible 3D UIs, in a single visual environment
- Ordinary display devices are supported, and the capabilities of modern and future VR gears are utilized.
- Any object has a unique global address in the form of a URL to support referencing.

The concept is based on the Aximion project. The mission of Aximion is to take the next step in UI evolution by providing a new visual experience that inherits the important features of the classic windows GUI and the Web-like nature of Aximion objects and adds the capabilities of modern and future display devices and VR gears. The special open standard will be introduced to guarantee the same look and feel across all OS platforms. The development of this standard is still at a very early stage but it's going to be like some hybrid of HTML and OpenGL.

Figure 8. Unified User Interaction Environment



Unlike any of the usual OSs, in the Aximion, all root UI nodes have a global persistent unique address in the form of a URL. This gives UI objects capabilities intrinsic to Web resources, such as links, history, etc. and makes their life cycle different from that used in ordinary operating systems. With such a capability, it's now possible to globally refer to any object using URL semantics. Whenever a user creates something, it always gets a predefined or random unique address assigned to it, and all data related to that instance is kept until at least one reference to this object exists. This allows the user to link to any objects exposed by the UNA in the same way as s/he does for files, Web pages, and other resources.

At the shell level, several UI concepts have been introduced in the Aximion to help users personalize and organize their environments.

The Fields

The Aximion rethinks the idea of the Desktop. In the Aximion, desktops have evolved into the *fields*. The field is a place where different aspects of a single project/context/interest can be grouped into a single UI entity.

Figure 9. Present-Day Application-Context Paradigm

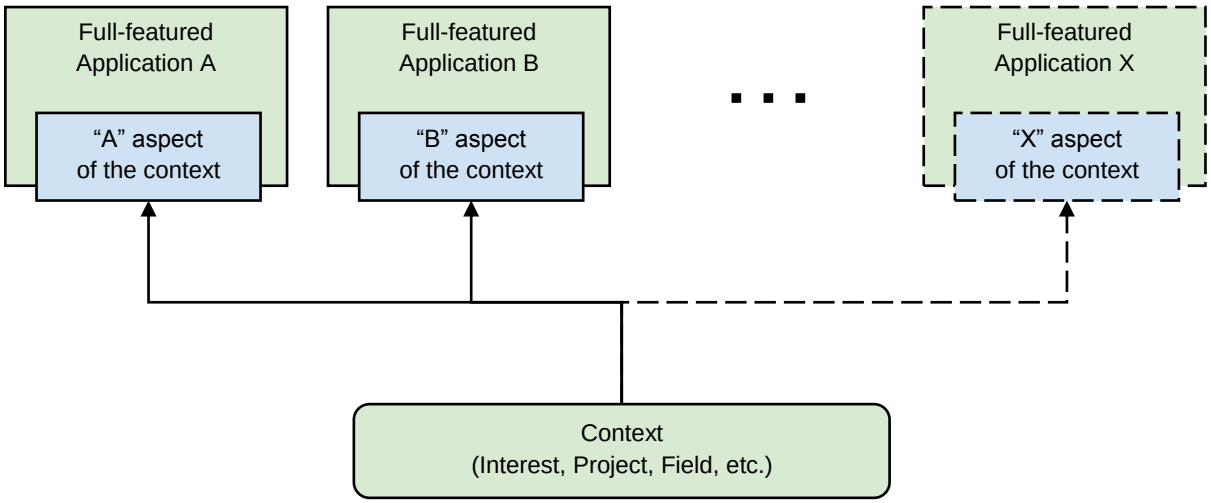
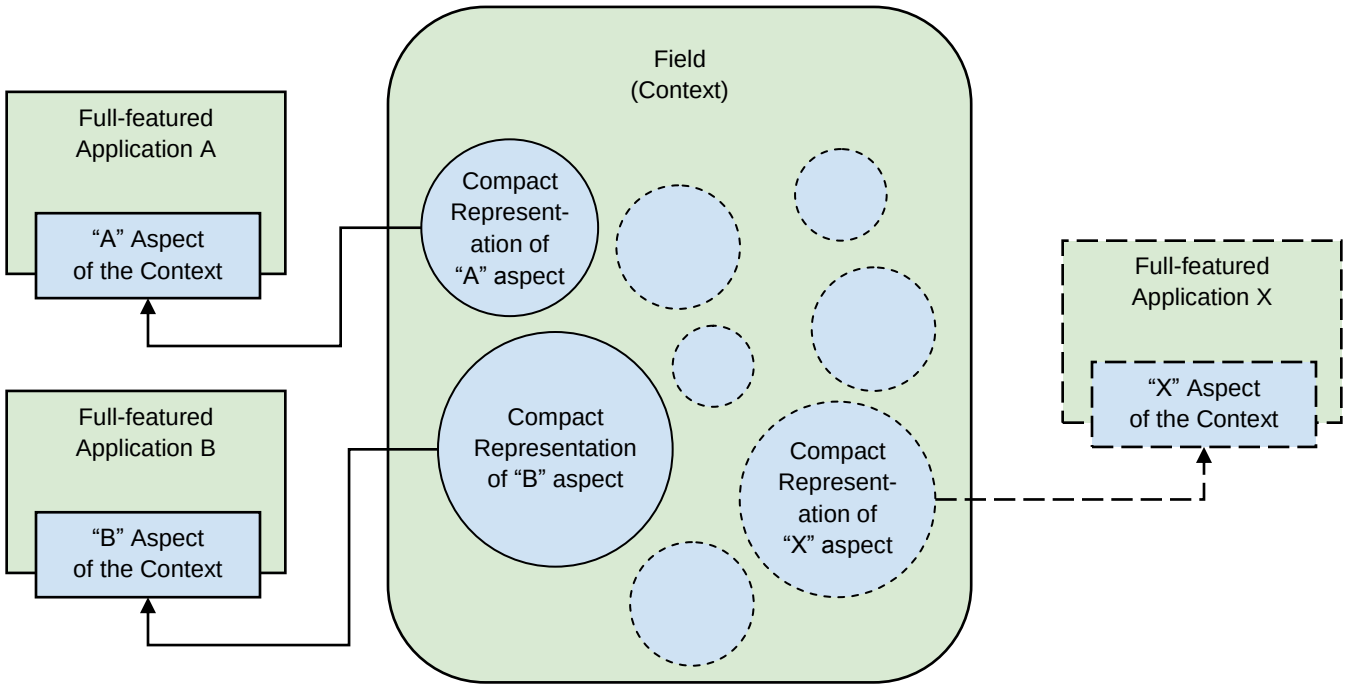
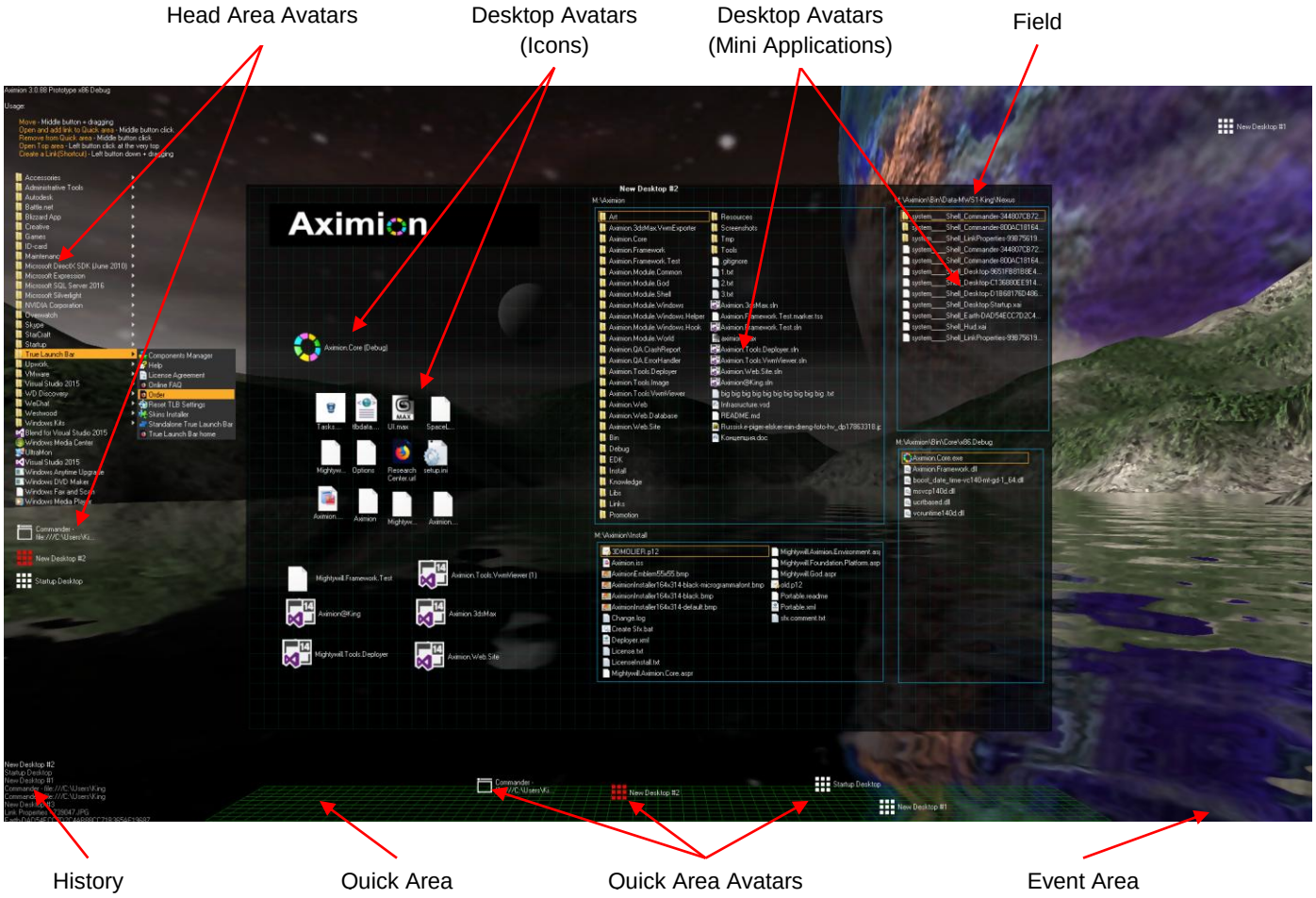


Figure 10. UUIE Application-Context Paradigm



For example, if you have a particular project, you may want to gather all related files, links, and remote and other resources in one place and organize them according to your preferences. Aximion fields look similar to regular desktops but can contain not only normal icons but also so-called *avatars*. Each avatar represents some aspect of your particular context, project, or interest. They can change their look according to their importance for a user or to other needs. The look of avatars may vary from simple icons to mini-applications with rich functionality more like mobile applications.

Figure 11. Aximion – The Implementation of UUIE



Any application UI can be attached to any other. For example, if a top-level UI element of application A is attached to a top-level UI element of application B, and then a user opens B, A shows up automatically. The user can then create various application unions that reflect some corresponding contexts. For example, if the user is a software developer then s/he can create an individual field for each of their projects and also create several other fields that would contain shared tools and resources required for all of those projects. Once any of these project fields are open, all attached desktops with shared tools will appear together with it.

The Quick Field is used to place links to objects required for the current session, in much the same way that tabs are used in browsers. A middle button click helps the user to place an avatar in the form of a link to that area automatically. They do not disappear when the user closes the object, so when needed, the object is available for reopening.

All avatars placed in the Head Field are always available for the user and always stay on top of other objects. Using VR gear, this area acts as a heads-up display: i.e., it always stays in the field of view whenever a user's head moves.

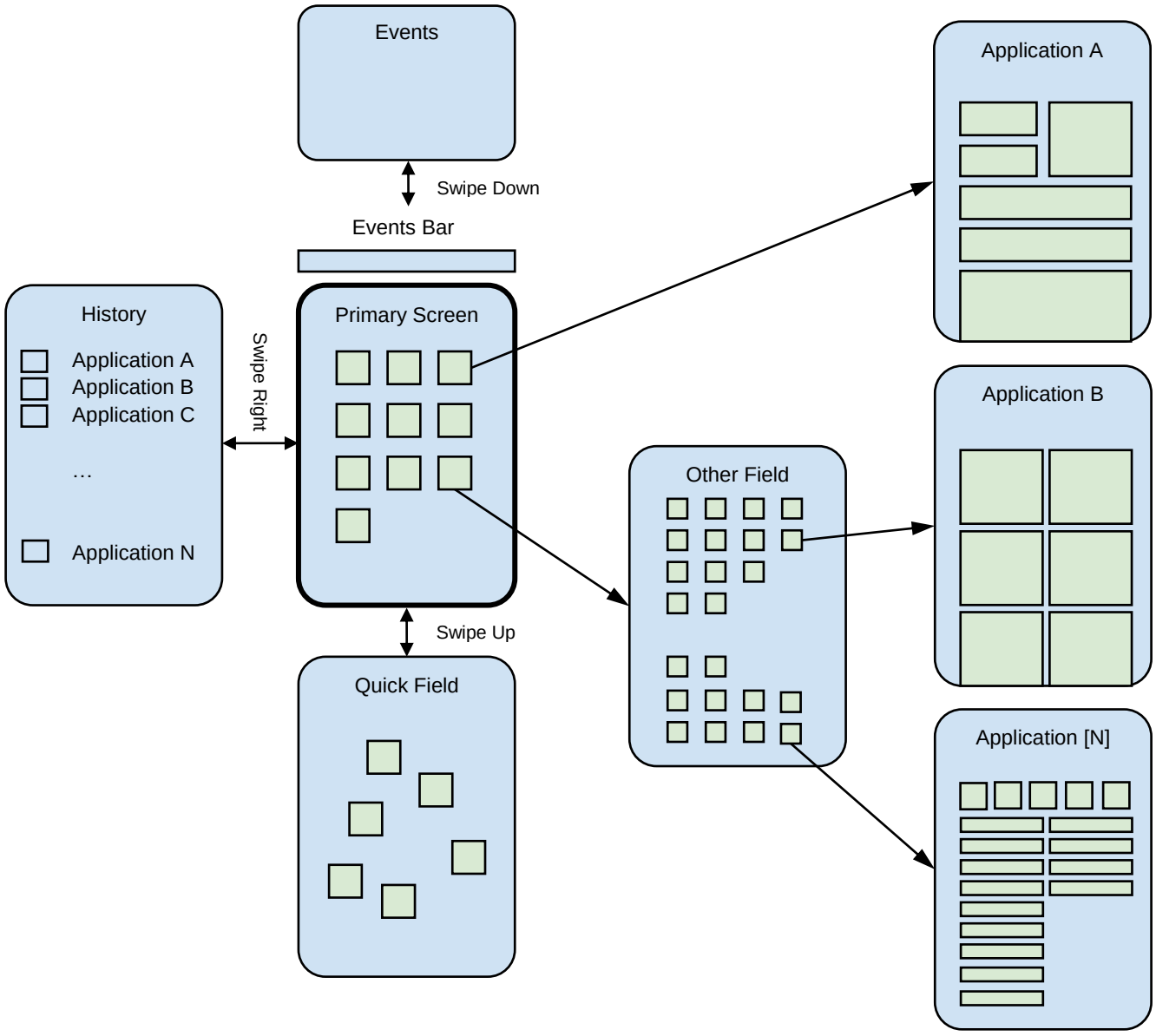
The History area is a history of user navigation. This function is similar to a browser history, and together with Quick Field effectively replaces the standard desktop taskbar behavior.

The Events area is for showing events and progress reports, similar to the desktop tray but with advanced functions. It can show simple notifications about events that require user attention. In addition, any user-initiated long-term operations should place representation items in this area so a user has a continuous report about current time-consuming processes.

Mobile Mode

The technology is designed to support not only desktop and VR but also mobile environments. In this case, only compact representations (avatars) are used to display the application UI. The following diagram shows how the UI concept described above maps to mobile environments.

Figure 12. Mobile UI Mapping



Conclusion

Ultranet has been developed to solve long-standing problems associated with the Web and local platforms, and at the same time it introduces many innovations to revolutionize and improve the development, deployment, delivery, and user interaction experience of various applications, websites, and services. By combining and utilizing the most advanced information technologies available, Ultranet is shaping the next era of the Internet.