Darwinet

Open P2P Service Cloud Specification

Version 1.0 Revision 10

This manual was produced using *ComponentOne Doc-To-Help.*™

Contents

Introduction to Darwinet P2P Service Cloud

Creative Commons Share Alike license

Licensed for use and contribution

Welcome!

You are reading the introduction to the Darwinet P2P service cloud project and components.

Please note that this is an open source project in the making. We are trying to update documentation as fast as we can but always expect a gap between "where we are" and "what we are telling" in documentation.

To get a grip of what Darwinet may be used for, see "Darwinet application examples" on page 5.

To know more about the driving vision of Darwinet, read "Darwinet Overall vision" on page 19.

If you are interested in technical details of Darwinet read "Platform independent Darwinet Mechanisms" on page 37.

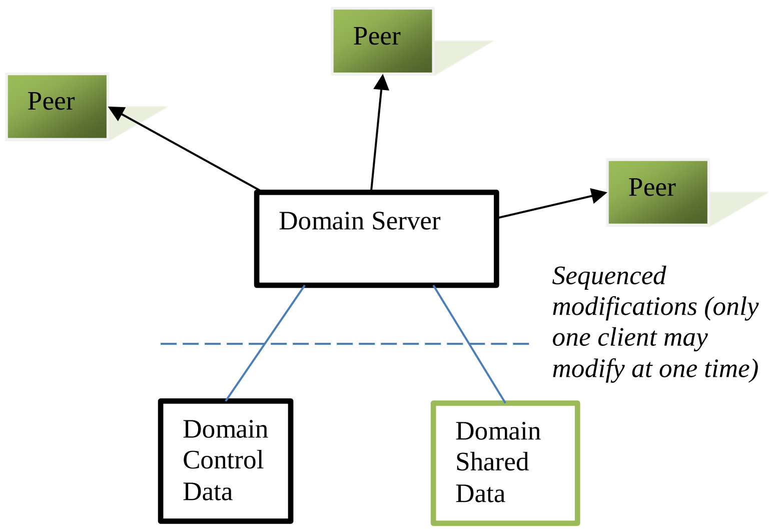
To guide the development of Darwinet software there are "Darwinet Requirement specification" on page 27

Welcome to the world of Darwinet – your ticket to a free digital life!

Introduction to Darwinet P2P network versus other networks

Server hosted network domain

In a server centred network all node interaction are made through and synchronized by a central server.

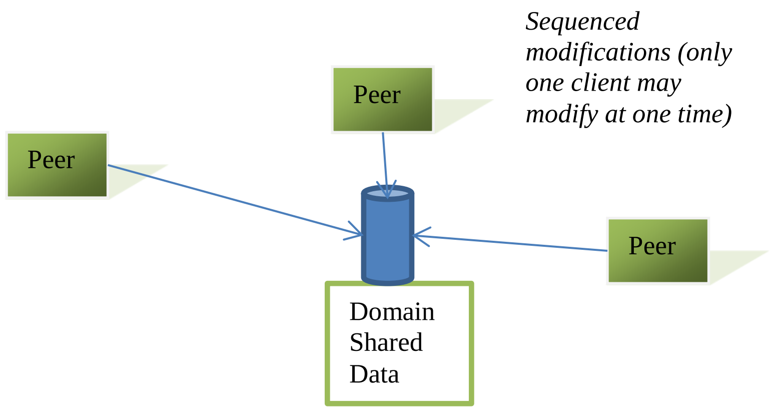

Peer
Peer
Peer
Domain Server
Sequenced modifications (only one client may modify at one time)
Domain Control Data
Domain Shared Data

Characteristics:

- Online access only.
- Peers have to wait for each other to modify shared data. The Server ensures modification sequencing.

Network file folder P2P sharing

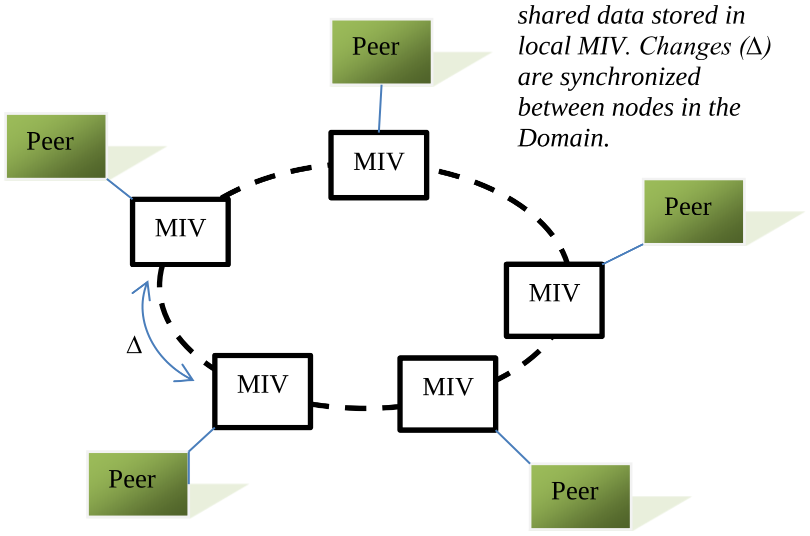
Desktops based on Windows, Mac OS, UNIX etc. all provide file folders that may be shared between computers in a network. This is a form of P2P interaction where each node interacts directly with the node that exposes a shared resource.


Sequenced modifications (only one client may modify at one time)
Peer
Peer
Peer
Domain Shared Data

Characteristics:

- Online access only.
- Peers have to wait for each other to modify shared data. The file system implements file lock and release to ensure only one Peer may modify a file at one time.

Darwinet P2P Service cloud network

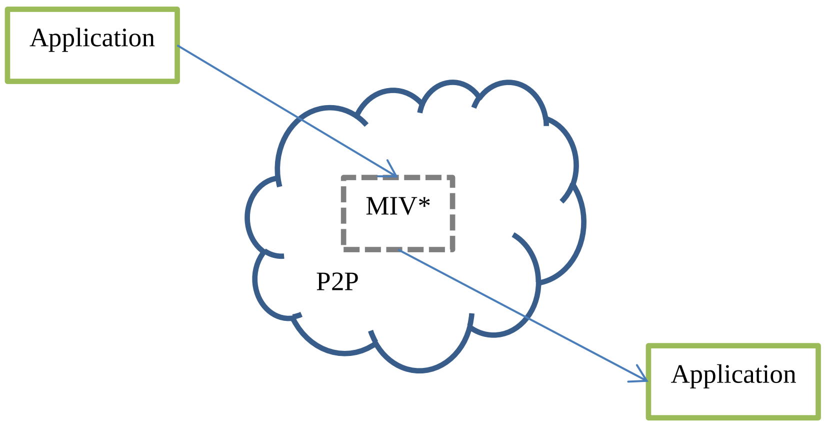


Domain control and shared data stored in local MIV. Changes (Δ) are synchronized between nodes in the Domain.

Darwinet application examples

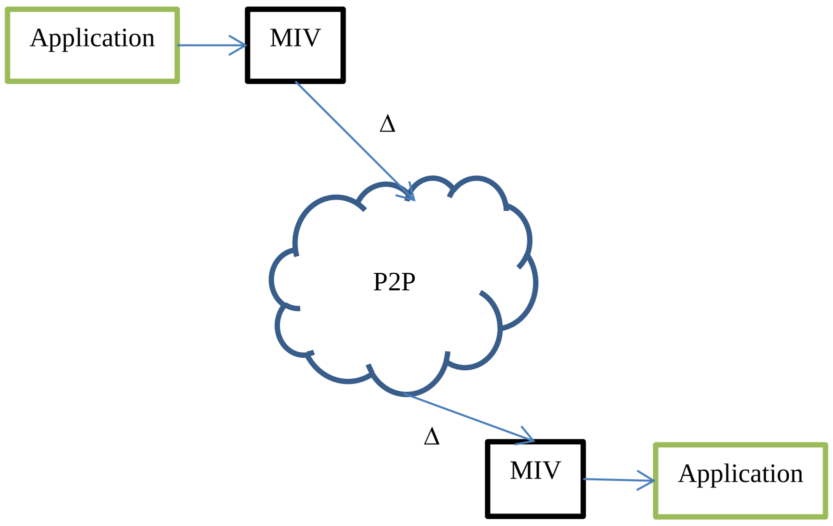
General model for Darwinet Application P2P interaction

A Darwinet Application interacts with a virtual shared data model in the Darwinet P2P Domain.



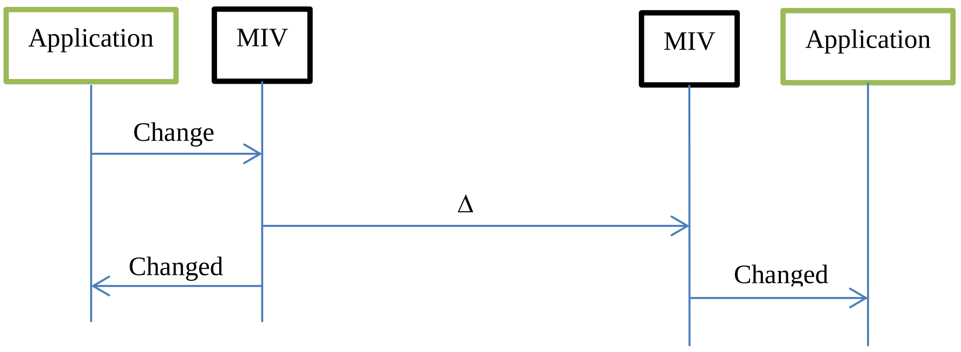
The Shared Data Model (The Model, Instance and Value) ties the application together by synchronizing changes throughout the Domain.

In reality each Darwinet node has its own MIV instance and Darwinet synchronizes each node MIV state as changes occur within the Domain.



The figure above show how each Darwinet node has its own MIV instance.

When one Application makes a change to the MIV the actual change is distributed as changes (called deltas marked with symbol Δ) to the nodes in the Domain.

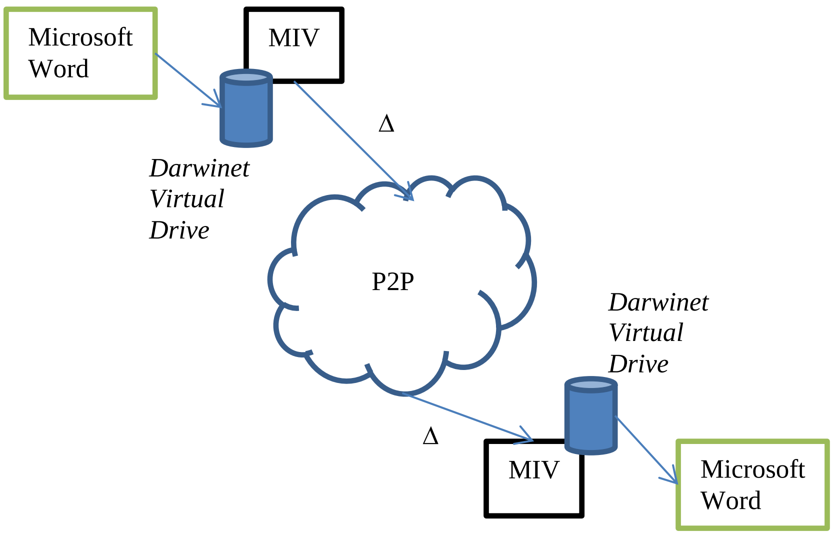


A changed MIV reports the Change back to the Application.

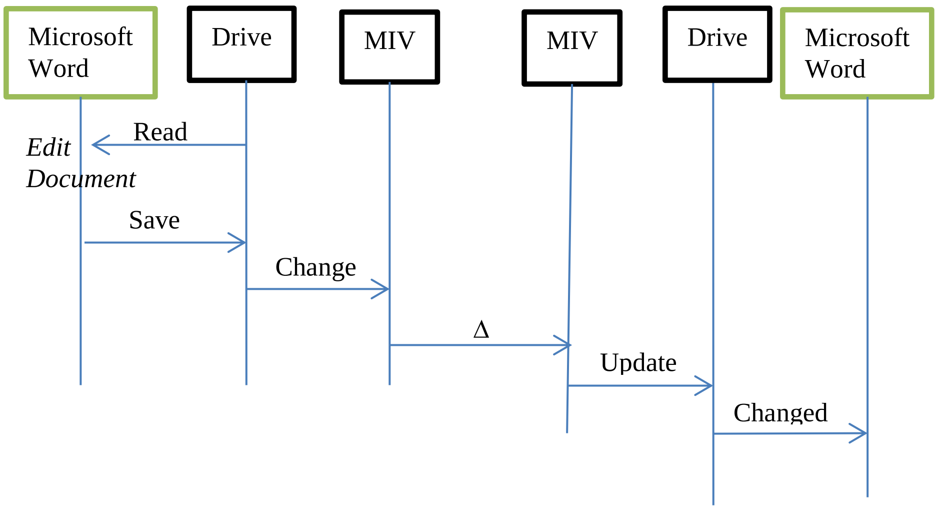
Example of sharing a Word file in Darwinet

A Darwinet domain may expose files to applications at a node. A Darwinet node may expose these files through a virtual drive. This enables existing applications to interact with the Darwinet Domain files.

The user may then use Microsoft Word to edit a file in a Darwinet Domain through the file system interface provided by the virtual drive.



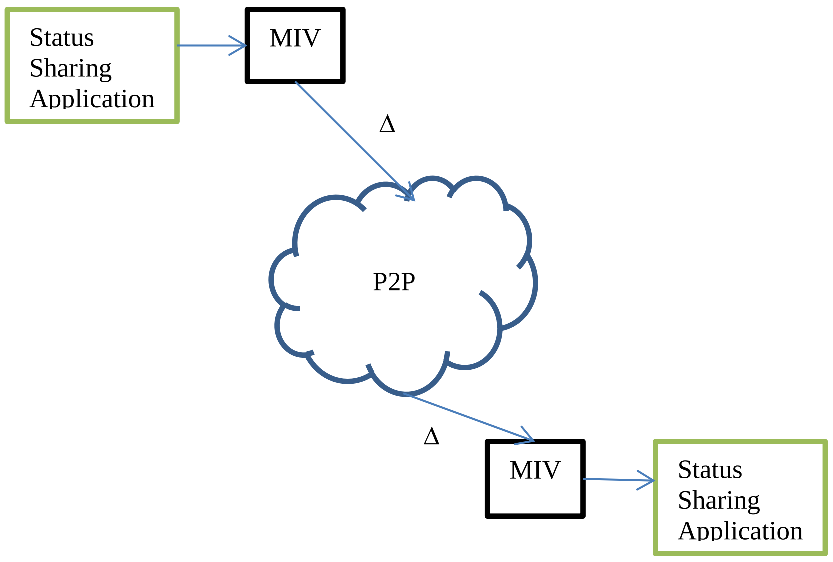
When the file is stored Darwinet performs a "file diff" operation and sends the file change to the other nodes of the Domain.



Note: See the Darwinet Change Conflict handling about how Darwinet handles changes that affects the same file or data.

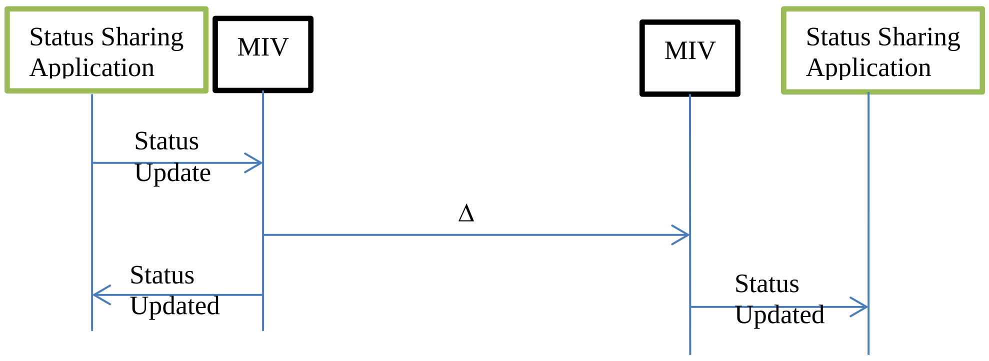
Example of a status sharing (social interaction) application in Darwinet

The shared MIV mechanism of Darwinet makes it easy to create a social interactive application based on Darwinet.



The figure above show how a Status Sharing application interacts through the shared MIV.

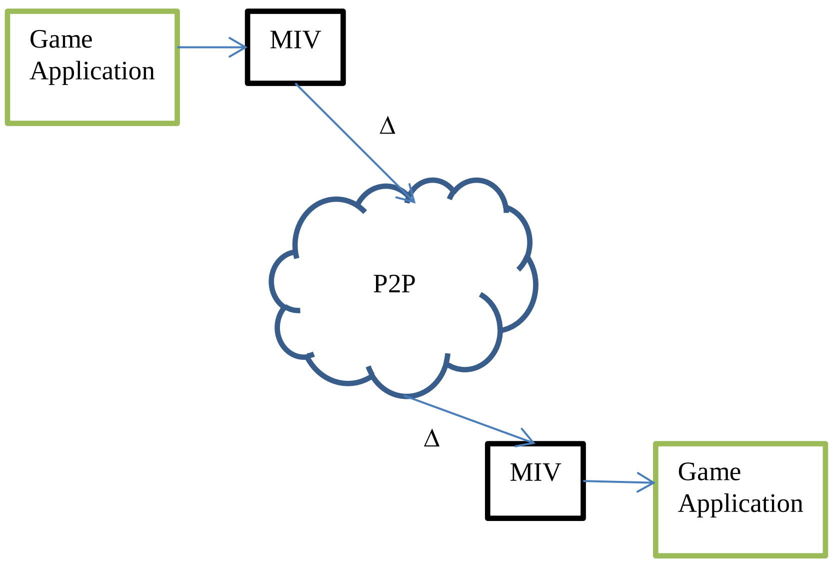
When the user posts a status update the application stores it in the MIV. Darwinet then distributes the update to all nodes of the Domain.



The posted Status Update will trigger the Updated event of listening Status Sharing Applications in the Domain.

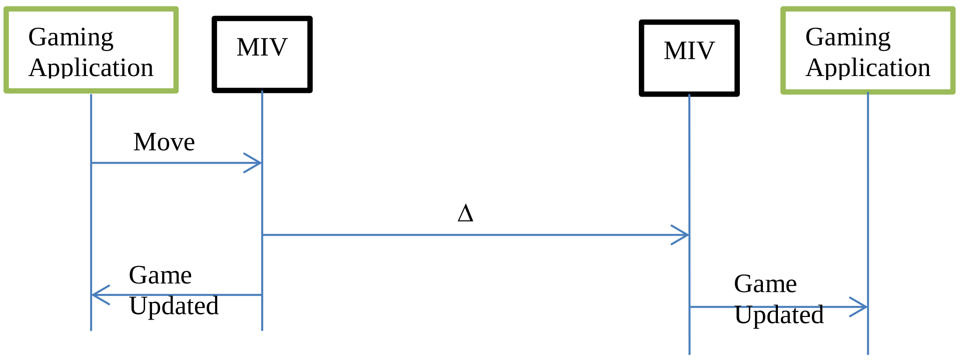
Example of Online Gaming application in Darwinet

The shared MIV mechanism of Darwinet makes it easy to create an online gaming application based on Darwinet.



The figure above show how an online gaming application interacts through the shared MIV.

When the user makes a move in the game the application stores it in the MIV. Darwinet then distributes the move as an update to all nodes of the Domain.

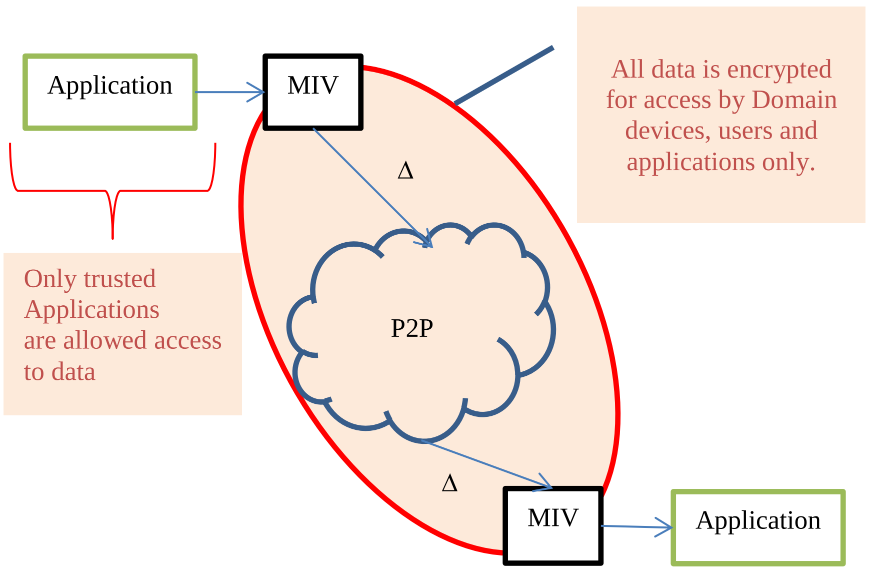


The distributed move then triggers the Updated event of the connected online gaming Applications in the Domain.

Darwinet as a platform for BOYD (Bring Your Own Devices)

Darwinet is private clouds of trusted devices

A company may use Darwinet to build a cloud of trusted devices. All data within the cloud are encrypted for access only by trusted users on trusted devices using trusted applications.


Application
MIV
Δ
P2P
Δ
MIV
Application
All data is encrypted for access by Domain devices, users and applications only.
Only trusted Applications are allowed access to data

A company may add an employee's own device to a company Darwinet Domain making it a trusted and manageable device of the IT-department with regards to the data of the Darwinet Domain.

Darwinet is platform independent

Darwinet clients are developed for all major platforms (devices). This means that a company may invite iOS devices, Android Devices, UNIX devices, windows devices and so on to participate in a company Domain.

Remote destruction of company data

When a Darwinet node is excluded from a Domain it will automatically loose access to the data of that domain.

- The Device is now longer authorized to access other nodes not any data locally stored in the Domain.
- The node encryption keys needed to access any local data are by default deleted.
- The local data itself is by default deleted.

A Darwinet node membership may be configured with a set of Domain data access restrictions. Some of them are:

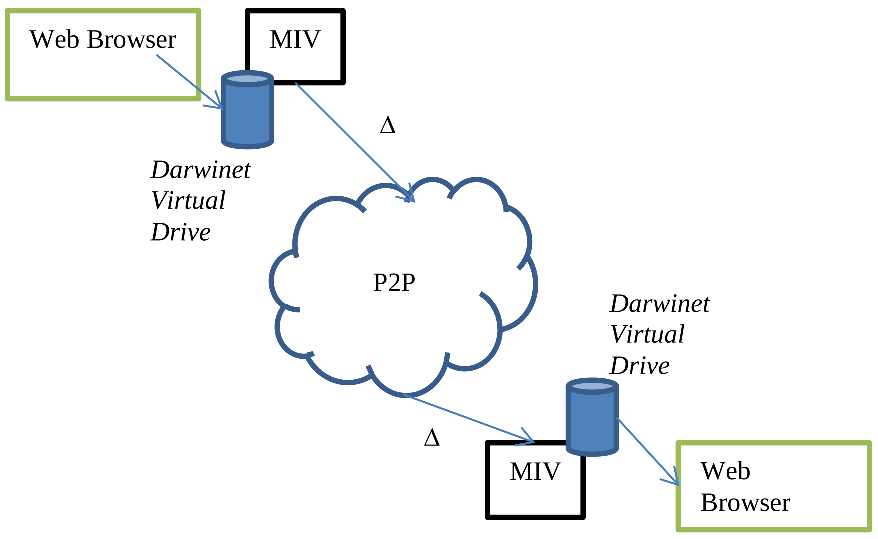
- Regular re-authorization required. If this option is set the node is required to regularly renew domain data access authorization. If the node fails it will by default delete the encryption keys needed to access the domain data.
- Automatic membership revocation. The node may be configured to require its membership in the Domain to be renewed regularly. If it fails to access the Domain and retrieve membership renewal it will by default delete all data of the domain from local storage.

All in all the security mechanisms of a Darwinet Domain means:

- A node may be **excluded online** and the excluded node will acknowledge that the node data is destroyed.
- A node may be **excluded offline** by self-destruct mechanisms.

Darwinet Hosted Local Web Place

By publishing a web place files in a folder structure of a Darwinet Domain it is possible to access and interact with them using a web browser.

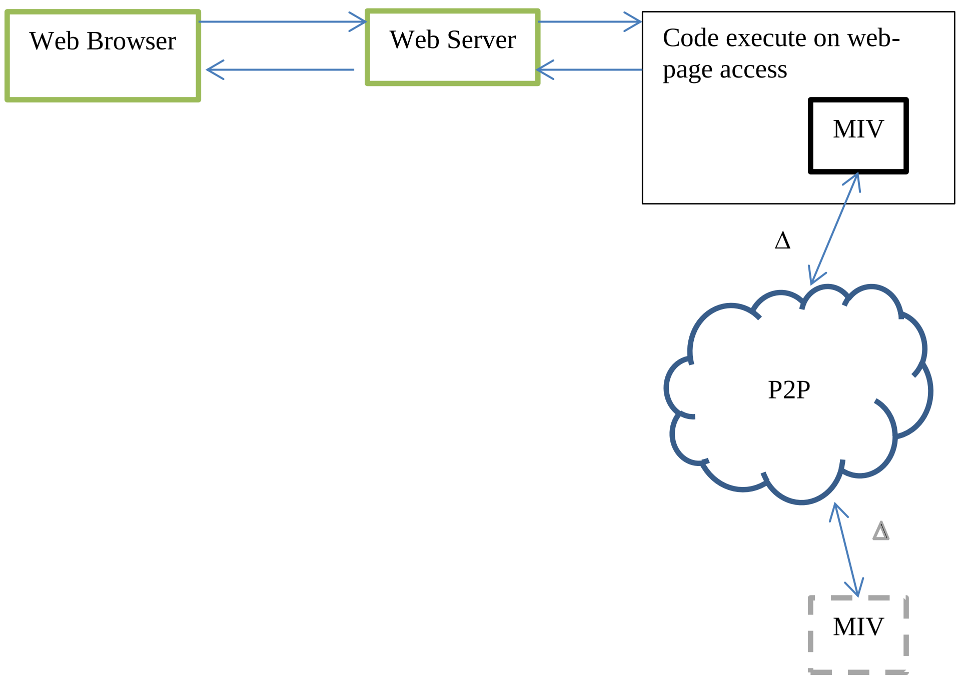


Existing frameworks to create local Wikis will work out of the box and become shared wikis within the domain if published this way.

Note: Functionality of the web place will then be restricted to the functionality of the web browser as there is no Web server involved.

Web Hosted Darwinet Nodes

A Darwinet Domain may be hosted on Web servers by using Web server Darwinet clients. In this way it is possible to operate a Darwinet Domain entirely through web browsers and use no local Darwinet Client at all.



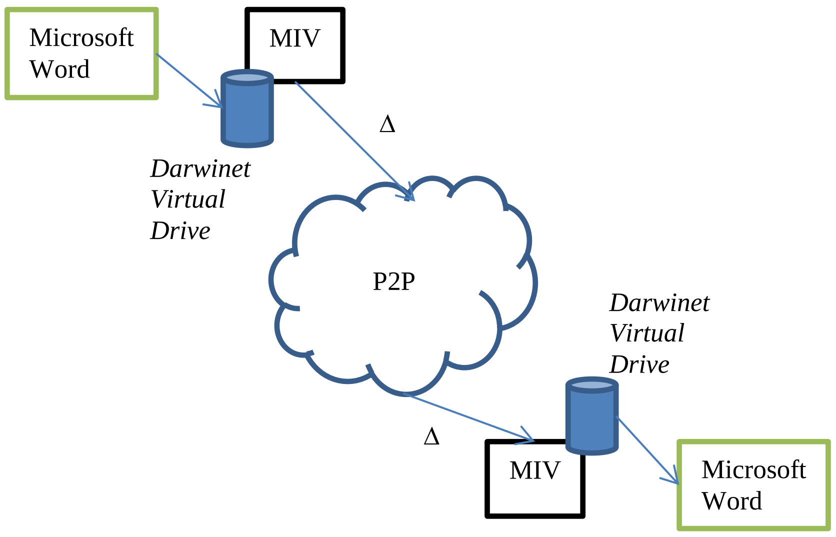
In this case all Darwinet mechanisms have to be implemented into the code executed on Web access.

- A user interface to a view into the Darwinet Domain including login, view selection and so on unless hard coded by the web access.
- The messaging protocol with other nodes may be restricted by the available services of the Web browser. But SOAP, e-mail and other common Web protocols may be used and must then be supported by the other nodes of the Domain.
- If the web hosted nodes are seldom accessed the amount of work needed to update the node may slow down the user experience. The user interface should show current state and then update the state while the node is “catching up” (clearing and settle its state with the Domain).

Darwinet virtual drive

Virtual drive as native application interface

A Darwinet Domain may expose a virtual driver on platforms where that is possible in the hosting OS (Like Windows, UNIX, and Mac).



A Darwinet Domain node on a Microsoft platform may allow a locally installed Word application to edit a word file in the domain exposed through a virtual driver of the domain view.

Handling change conflicts of files in a Darwinet virtual drive

What is a conflict?

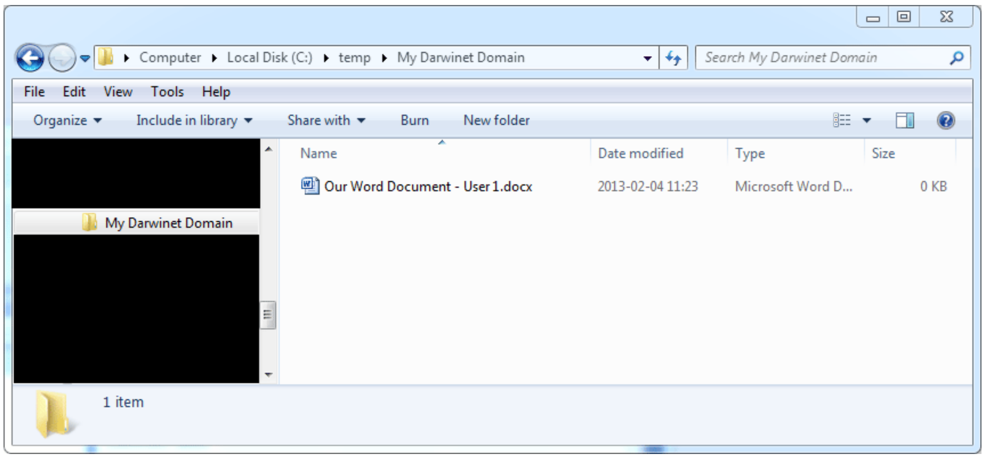
Files hosted by a Darwinet Domain view may be changed asynchronously on several nodes at the same time. The change may be made by different users or by the same user on different devices.

As long as the changes made to the file do not conflict Darwinet will have no problem of updating the shared file properly with each individual change.

But simultaneous change to the same file or data in a file is a conflict that must be resolved by a user. Darwinet will not be able to know which one of the conflicting changes that is the one to keep. Or if a merge of the conflicting changes is the one to go for.

Branch file on a change conflict

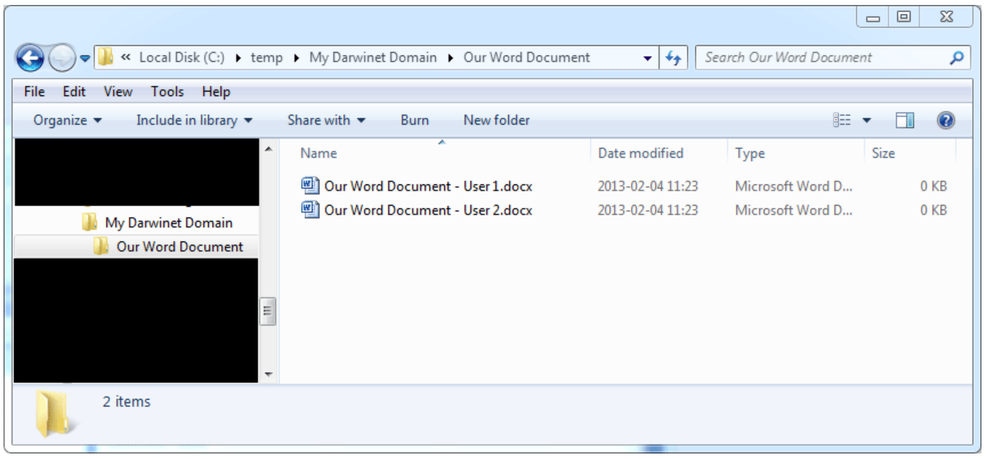
By default the Darwinet virtual driver will create several versions of the file when a change conflict occurs. The file will be exposed as a directory containing the different versions of the file. The user may then use an appropriate tool to merge the files together again.


Computer ▸ Local Disk (C:) ▸ temp ▸ My Darwinet Domain
Search My Darwinet Domain
File Edit View Tools Help
Organize ▾ Include in library ▾ Share with ▾ Burn New folder
[redacted]
My Darwinet Domain
[redacted]
1 item

Name	Date modified	Type	Size
Our Word Document - User 1.docx	2013-02-04 11:23	Microsoft Word D...	0 KB

How it may look when the Word document "Our Word Document" exposed in folder "My Darwinet Domain" in a Darwinet domain virtual driver in Windows 7.

At some point both user 1 and user 2 makes conflicting changes to the file.


« Local Disk (C:) ▸ temp ▸ My Darwinet Domain ▸ Our Word Document
Search Our Word Document
File Edit View Tools Help
Organize ▾ Include in library ▾ Share with ▾ Burn New folder
[redacted]
My Darwinet Domain
Our Word Document
[redacted]
2 items

Name	Date modified	Type	Size
Our Word Document - User 1.docx	2013-02-04 11:23	Microsoft Word D...	0 KB
Our Word Document - User 2.docx	2013-02-04 11:23	Microsoft Word D...	0 KB

Darwinet detects the conflict and exposes the two versions of the file in a folder with the original file name.

The user may now edit the two files to create only one file again.

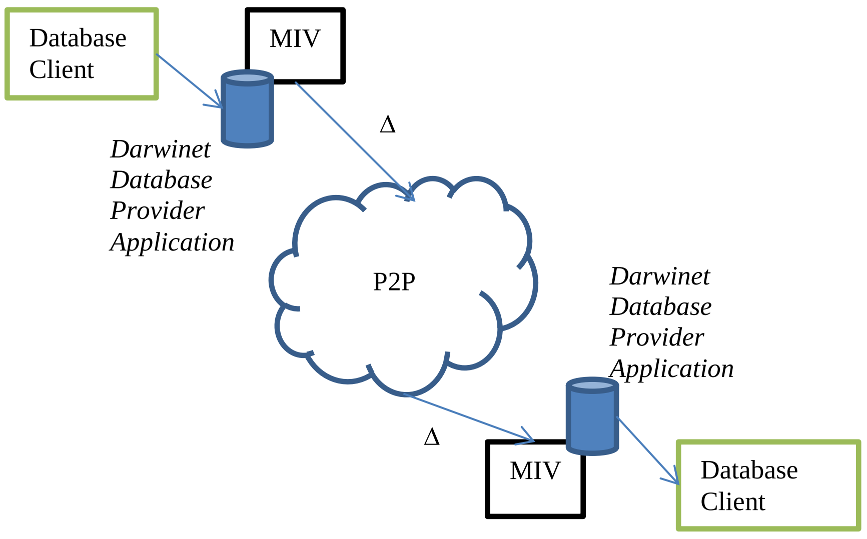
Darwinet will detect when the "folder" contains only one file and remove the folder again, taking the edited file to be the new shared (merged) version.

Database access into Darwinet

Database as native application interface

Database access through adapter application

A Darwinet Database Provider Application may provide a Database access to a shared MIV in effect creating a shared database without using a central server.


Database Client
MIV
Δ
Darwinet Database Provider Application
P2P
Darwinet Database Provider Application
Δ
MIV
Database Client

Handling change conflicts created by Database Provider Application

A Database client may involuntary make simultaneous changes to the same data as provided in the Database interface.

The Database Provider Application must implement a strategy how to resolve such conflicts. If the conflicts are automatically resolved the changes made must be reported back to the database client using the event mechanism of the Database interface used.

TBD.

SQL Database Provider Application

The Database Provider Application may provide an SQL Database interface to the shared MIV.

TBD.

Darwinet Licensing

Licensing goal and application

We, the initial developers of Darwinet states that:

- Darwinet networks shall be made available to all without restrictions.
- Contributors shall license their work to promote maximum spreading of Darwinet.

The following shall apply to maximise spreading and usage of Darwinet networks.

The Darwinet Specification shall be licensed to allow anybody to use and contribute.

Darwinet functionality shall be allowed by both proprietary deliverables. This shall promote maximum freedom of providing Darwinet.

Darwinet shall be available as open source.

Open and Proprietary applications shall be allowed to compete as Darwinet Applications.

- Proprietary Applications must be able to link to open source Darwinet without restrictions. This shall promote maximum spreading of Darwinet development and usage.

Darwinet open source modules shall be licensed using GPL, LGP or MPL.

- Darwinet modules that do not link with Darwinet applications shall be licensed under GPL license. This promotes the open source core to remain open source and have maximum contribution.
- Darwinet modules that link with Darwinet Applications shall be licensed under LGPL to allow proprietary applications to remain that while interacting with an Open Source Darwinet. This shall reduce the incentive for proprietary Darwinet implementations that could reduce contribution to the open source initiative.

There shall be allowed to develop proprietary Darwinet implementations. But they shall not be allowed to derive from Open Source.

- Darwinet Open Source licensed under GPL or LGPL prevent derived work to become proprietary.
- Darwinet Open Source licensed under MPL should also prevent derived work to become proprietary. But MPL allows proprietary Darwinet Core modules to be linked with ones provided opens source. MPL licensed Open Source may thus allow a proprietary implementation to bit-by-bit replace Open Source modules with proprietary ones until an implementation based on totally proprietary source is created. This may or may not be an Issue for spreading Darwinet.

Darwinet Overall vision

Enable anyone to have one home they own and control for their whole Digital life

In this digital world we find ourselves living our digital life at an ever growing number of locations. And the digital information that we produce is stored on more and more servers operated by companies and other organizations.

What we lack is:

- A home for our digital life.
- A home that is ours.
- A home we trust and control.
- A home we may keep regardless of companies and organizations that comes and goes.
- A home that may evolve as the digital world changes.
- A home where we may keep all our digital information.

Darwinet should be that place. It should be the central place to store and organize personal digital assets and manage social networks and digital funds.

The user owns and controls private data

All data within a Darwinet domain is stored only on nodes controlled by the owner/creator of the domain. The domain data lives in a cloud of devices authorized by the Domain. A company is able to build an intranet where data only lives on computers and devices appointed by the company.

Public services may be brought to private Data

In Darwinet the data is kept on the computers and devices within the network while external services may be used to work on it. This enables:

- Internet provided services may be used on sensitive data that needs to be kept within company or country boundaries
- Cross service co-operation on shared data
- Data is not duplicated or distributed to server locations for services used.

Users may centralize and own their digital life

Darwinet enables existing public internet services to be centralized around personal and private data. For example:

- A Darwinet client may access a user's twitter and Facebook account and mirror all shares in the user's private Darwinet domain. Centralization does not include private data being provided to yet another internet actor.
- A user may connect the Darwinet domain to their personal payment services and use a single payment procedure when shopping online.

Darwinet users should be able to trade in the network

A Darwinet user should be able to transfer funds to other Darwinet users at low cost and without any intermediate agents. It may be to transfer money to a friend, purchase something on the web or retrieving money from a bank account.

Shared Data must be able to evolve without conflicts

Users must be able to work and change shared Data while it is being changed. Any conflicting changes must be handled and solved in a non-intrusive way. Ordinary users should be able to understand and solve conflicting changes without any problems or worries.

Application updates "just happens" without involving the user

The user should not have to wait for or accept an update of an application. Instead an Application update should behave as an evolution. It is then up to the user to choose whether to use the evolved application or the one before.

Files grouping to a location should be possible without moving or copying files

It should be possible to store a file once and then just use a reference to that file at several locations. This means that if file folders are used, the folders should not contain an actual instance of a file, just a reference to the file.

This enables files to be organized in different ways within one view, or across multiple views, without having to clone that file.

A Darwinet user should not have to administrate the difference of a "master" file and a set of "linked" files. Consider to let Darwinet add information to the file name so that it is unique cross the whole Domain! If done in a user friendly way this would enable a file to always be accessed by reference and the user will always recognize the same file even in different folders and views.

Not going for this approach calls for a solution to how the user is able to know if a file at one location is a unique file or if it is the same file as one in another location.

- One way is to give the file a storage location in the form of a path recognizable by the user. The user may then query the file storage location and thus identify the file instance.

The downside of this approach is that the user must expect all viewed files to be references to a master file and must query for the master file to detect "sameness".

- Another way is to display reference information for files. Have one file as the "master" file and all references to it are indicated as such in some way. Consider to mimic how the Linux "ls" command displays a symbolic link.

  The downside of this approach is that the user may expect all file viewed files to be "master" files unless shown as a link which may not seem natural to the user.

The framework as open source

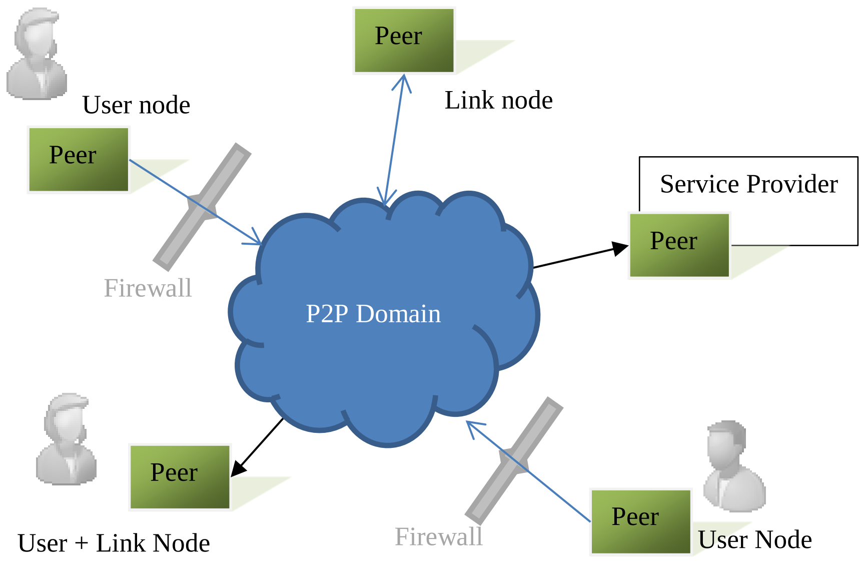
The base framework to set-up and run a Darwinet domain should be free and developed as open source. The vision is that this will create the drive to encourage the network to become a large standard network and thus become a marketplace for application and services available on all devises connected to the internet.

Open market for Darwinet service providers

Anyone should be allowed to provide a Darwinet service provider through which applications and services to Darwinet domains may be provided by third party providers. The vision is to create a marketplace that drives the Darwinet community to grow.

Small set of base Darwinet services used to build all complex services

The vision is that the base services provided by the open source base framework of Darwinet must support all services of the Darwinet domains. This ensures interoperability and robustness of the networks built. An example of this vision is to consider making the Darwinet base network engine a Darwinet application like any other application and have it provided by a Darwinet service provider.



This small set may be the following:

- All Darwinet nodes are peer to peer nodes with a common set of interfaces and services.
- The common set of interfaces and services of a Darwinet node must support the node to act as both a link-node, a user node and a service provider node.
    - A user node is a node where a user may access domain data using available applications.
    - A link node is a node providing two other nodes with a communication link where they are unable to link directly to each other (e.g. nodes behind firewalls).
    - A service provider node is a node that has access to a Darwinet service provider and may provide applications to the domain.

  Finding this set of interfaces and services ensures that the network is true peer to peer and enables building of robust network domains without the need of networking knowledge or configuration issues.
- Darwinet nodes must provide data security for all data and communication within a domain. This enables users to trust the domain with sensitive data (e.g. business data and personal data).
- Darwinet must provide “version control” and change history for all data of the domain. This ensures that any evolutionary issues of data and application development have a solution within the framework behaviour.
    - Enable a “view” of the domain at any specified past time with correct state of data model, data instances, data contents and available applications at that time.
    - Mechanism to correctly synchronise work done on-line and offline at different times.
    - File view that “forks” documents into several documents in cases where manual merge of changes has to be performed. The view being instantly understandable by the users combined with easy-to-understand indications and directives on what to do to come back to the merged document. (Consider transforming the document to a folder containing the different version of documents needed to be merged. The folder collapses to a document again when merging has been performed).

Darwinet usage scenarios

Install Darwinet services on a new node

Darwinet services shall be supported for all platforms in common use and easily ported to any additional platforms that users may require.

- Desktop OS's (Windows, MAC UNIX etc.)
- Mobile Device OS's (iOS, Android, Windows Phone etc.)
- Web Application node (HTML5)

Create a Darwinet Node on Desktops

Install Windows Desktop Darwinet node services

On Windows the applications and services of the Darwinet may be distributed as a standard installation executable.

The Executable will install

- A Windows Service that provides a COM interface to Darwinet components on the desktop.
- As a Virtual Drive creator that enables the user to map a file system drive to the files of a Darwinet View.
- A Darwinet node administrator application with a user interface to perform administrative operations.
- Administrative Darwinet applications are installed as short cuts in the Start Menu and possibly as short-cuts on the Desktop.

Install UNIX desktop Darwinet node services

TBD.

Add a mobile device to a Darwinet Domain

TBD.

Create a Darwinet Node on Web servers

TBD.

Create a new Darwinet Domain

TBD.

Extend a Darwinet Domain with a new node

Invite a new Desktop node

An Administrator (owner) of a Domain may use the Darwinet manager application of a Desktop to invite a new desktop node to the Domain.

TBD.

Invite a mobile device node

TBD.

Invite a Web server node

TBD.

Adding new user to a Darwinet Domain

The Administrator (owner) of a Domain may use a desktop Darwinet Manager to administrate the Domain users.

TBD.

Working with files in a Domain view through Desktop virtual drive

TBD.

Working with files through Darwinet Application

A Darwinet Application may access files of a Darwinet Domain and show a user interface to manage files.

On some mobile devices there is no file system available to the user (e.g. iOS). On these platforms the device application may still work with files in the Darwinet Domain but only as defined by the application.

TBD.

Edit file change conflicts in virtual Drive

TBD.

Attend to file change conflicts though Darwinet Application

TBD.

Attend to Domain data change conflicts

Darwinet applications share data with users within the domain. This data may be edited both online and offline and at different times. Therefore multiple users may involuntary make different changes to the same data.

The default behaviour of a Darwinet application is to branch the view on data change conflicts.

- The Application shall provide enough information to the user so that conflicts may be solved.
- The user shall be able to choose to keep a branch. Other user shall then be able to see the available branches and select which one to follow.

*Note: Support for branching is key to enable an evolutionary way of working together. The branch may represent two possible solutions or progress and enabling both to live on opens up for competition (***survival of the fittest***).*

TBD.

Install Darwinet Application from Darwinet service provider

TBD.

Select version of Darwinet Application to use

Accepting Darwinet Service upgrades

The executable that implements a Darwinet node shall be upgraded through the Domain that provided it.

TBD.

Purchase a Third Party Darwinet Application

The Darwinet network provides functionality through Applications that operates in the Darwinet network.

The application may be developed by a third party and may be published for purchase on a Darwinet service provider.

Sell or Buy Darwinet application

It must be possible to enable provider of the application to charge for the usage of the application.

Consider to use some of the same rules as Apple has defined for iPhone Applications. That is, free applications remain free. Bug fixes and version upgrades are free.

The Darwinet Network must provide a monetary transaction system that enables providers of applications to sell application over the network.

TBD.

Sell or Buy for Darwinet data

It must be possible to use Darwinet as a marketplace of data.

The mechanism behind that should be same as the market place of Darwinet applications.

This enables Darwinet to evolve also based on financial incitements allowing participants to share data and applications based on an economical value.

TBD.

Darwinet Requirement specification

Requirement specification Overview

In this document the requirements of Darwinet nodes are specified.

These requirements describe the core functionality of the Darwinet framework and the functionality that Darwinet applications must implement to operate within that framework.

TBD.

Darwinet functional requirements

Functional requirements overview

Darwinet defines a secure and controlled peer-to-peer network domain.

The Domain defines users, physical nodes and applications that are allowed to participate in data exchange within the domain.

Data and messaging within the domain is accessible only to members or the domain.

Routing of messages between domain nodes members are allowed by nodes that are not members of the domain but routing nodes must be authorized by the domain.

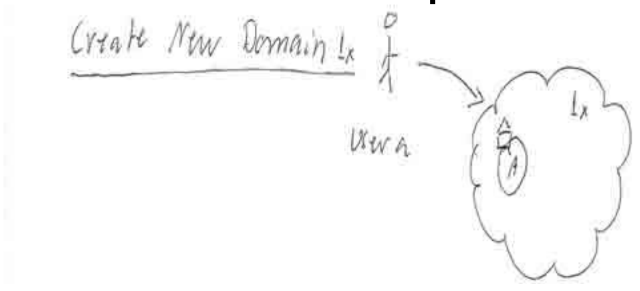
New domain nodes are created by cloning existing nodes in the domain (results in a node seed that may be used to create a new node)

A node seed allows a Darwinet engine to install the node clone on a new location. The new node must be Authenticated and Authorized before it is allowed to be a member of the Domain.

New users are added by a user with a role in the domain that is allowed to do this.

User authentication is made with PIN-code entered through picture mapping. Each PIN-digit is associated with a picture and the user selects the pictures in the correct order. Pictures are distributed in a random order at each authentication input.

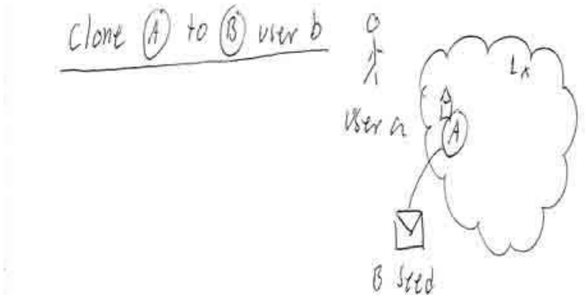
Create new Domain operation



A user “a” creates a new Darwinet Domain “1x” using the Darwinet Engine.

The result is a local file and data folder for the domain “1x” and the user and the created node are members of the domain.

Clone to create new node

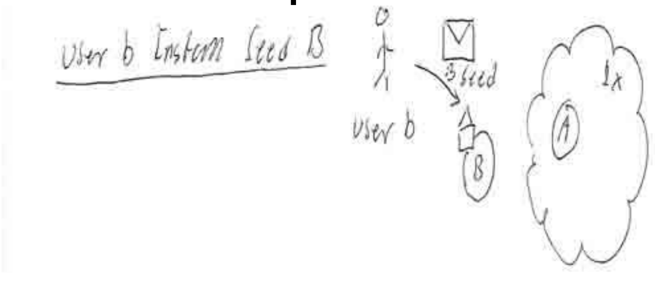


A user “b” is invited to the Domain. User “a” uses node “A” in the domain to clone node “A” into a node Seed for node “B”.

User b gets the seed as a binary and brings it to the location where the node is to be installed (for example a lap-top).

Note: This scenario need to be extended to describe the "invite" procedure in more detail. For example Darwinet needs a process for new users to request to join a domain, and to negotiate whether they want to join, including any financial negotiations/transactions that might need to take place.

Install Seed operation



User “b” uses the Darwinet Engine to install the Seed for node “B”. The result is a node “B” that will be able to become an authorized member of the domain “1x”.

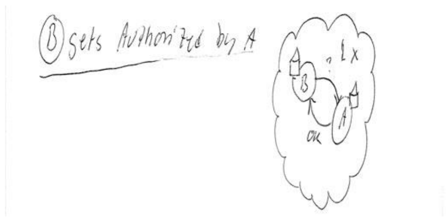
Note: The “Seed” mechanism enables a node that does not have Darwinet services installed to be invited. The Seed carries information so that the new node may “call back” to the inviting Domain and then provide information about itself. A node accessible over a Darwinet Domain may be seeded directly over Darwinet.

Seeding over Darwinet

A node that is already a member of the Darwinet service provider domain (i.e. already has Darwinet Services installed) may be seeded directly using the Darwinet P2P network.

- When a node is cloned the user directly selects the targeted node using the service of the Darwinet Service Provider.
- The seed is then sent to the new node through the Darwinet Service Provider Domain.
- The receiving node will then put the seed pending for a user to select to install. (Imagine the GUI to show a *update request* in a way resembling the indicators used on iPOS devices)

New Node Authorization mechanism



User "b" opens up the Node "B". Node "B" now connects to node "A" of domain "1x" to be authorized as a new node of the domain.

Authorization may require both the node "A" and user "a" to accept the new node "B". When Authorized node "A" returns an Authorization to B which is now a member of the domain "1x".

Node Authorization

A node must be authorized by the Domain before it is allowed to access any data within the Domain.

A node that has been removed from the Domain will fail to get Authorization. If this happens the Darwinet Engine will delete all local data of that domain from the node local storage.

A node is allowed to work off-line during a "grace-time". The grace-time is determined by the Domain, the node, the user and the Application. When the Grace-time is up the Darwinet Engine will not permit the node to access any local data of the domain.

Note: The Authorization mechanism needs to be described in more detail. New Node Authorization and "every day" authorization could possibly be the same.

- *Node Authorization shall include a certificate from the invitation protected by a secret provided by the invitatory and the invited user. The inviting node shall authenticate the new node this way. Also, the invitation must optionally restrict the Seed to be used only on specified platforms, devices etc. The Accept procedure shall include distribution of keys to the invited node when it is authenticated and authorized.*
- *Darwinet should enable components to query to reuse a user authorization so that the user does not have to re-enter credentials over and over. Consider to base the mechanism on OpenID or similar available mechanisms.*
- *Application Authentication and Authorization shall be made if required by the Domain. Consider to base Application Authentication on a mechanism similar used by Windows or iOS (e.g. certificates validated*

by a trusted part). Define the Darwinet Domain that provided the application to act as trusted part.

- *Data shall be protected from access using encryption. First level encryption shall use a Domain stored key. Second level encryption shall be based on a secret provided by the actor that is authorized access. A user password may be sufficient for User access encryption. The highest form of encryption shall require some form of HSM (hardware security module). Consider to follow Google USB key initiative.*

Final note: Any encryption made in software is vulnerable to shared process memory intrusion and the level of security depends on the separation mechanism provided by the hardware executing the OS.

User Authentication and Authorization

TBD.

Application Authentication and Authorization

TBD. Basically the same mechanism as for Node Authorization.

Data synchronization

All data of a Darwinet domain is maintained as a file of changes.

Changes are recorded on the following levels.

- The Data model. Defines data structures as named types.
- Data instances. Declares data objects as named instances of a type.
- Data values. Defines current value of a Data instance.

All changes are recorded and then sent to the other nodes of the domain to update the local views. Changes are “per-user” to enable tracking of who-made-what. This also enables merging of changes and detecting change conflicts or change gaps.

Darwinet networking requirements

The network must grow organically

A Darwinet network must start off with one node and then expand from existing nodes in the network.

New nodes must be authorized by existing nodes

When a new node is invited it must be authorized to join the network by at least one of the existing nodes.

New nodes must spawn from existing nodes

A new node must be spawn from an existing node. The procedure must include:

1. An existing node creates a seed for the new node
2. The seed is used to create the new node

3. The new node uses the seed to require membership from one of the existing member nodes of the network.
4. The network authorizes the spawn node.

The network must be able to grow using Darwinet nodes only

A Darwinet network must be able to expand with new nodes using existing nodes only. There must be no requirement of public network access or special service access. This enables Darwinet networks to be built using Darwinet node built in services only.

Local Darwinet API requirements

Definitions

The Darwinet local API is the interface between a local Darwinet application and the local Darwinet framework node.

The local Darwinet framework node is the local executable that provides the Darwinet functionality and mechanisms to any Darwinet Application.

There must exist a platform independent model description of the Darwinet API

The local Darwinet API must be defined in at least one platform independent model. The model is platform independent if it does not use any terms that are platform specific, language specific or in any other way dependent on a specific technology or framework.

For example, the model is platform independent if it may be used on a windows implementation, an iPhone implementation and a UNIX implementation without change.

It is also platform independent if it may be applied to an implementation using C++, C#, Java or other programming language.

The suggested name on this Darwinet API Platform Independent Model may be **Darwinet API PIM**.

Any Darwinet API implementation must have a one-to-one mapping definition to the platform independent model

When a Darwinet API is implemented, then that implementation must have a definition that maps the implementation uniquely and unambiguously to the platform independent model of the API.

This ensures that all API implementations are compatible and interchangeable. A Darwinet Application using one implementation of the Darwinet API may be adapted to another implementation of the API without losing any functionality or having to change the application level of operations.

For example, a .NET framework implementation is acceptable if the specification of the Darwinet .NET API is specified with a mapping to the platform independent model of the API.

Other examples of Darwinet API Implementations may be Java Reflection, Web-services, and XML over TCP/IP etc. or even file based using Gnuttela or BitTorrent.

Note: Both the API provider and the API client must of course still be compatible. The platform independent mapping to platform specific only states that the implemented API provides all the services required.

API client and provider must be able to ensure compatibility

A Darwinet API client must be able to ensure that it is compatible with the API provider. *E.g. if the API provides XML over TCP/IP but the client expects BERTLV over TCP/IP then they are not compatible.*

TBD.

At least one API implementation must be based on XML over TCP/IP

The Darwinet API must have at least one specification based on XML over TCP/IP.

An implementation of this specification may be based on SOAP or similar existing frame works.

Note: Any messaging representation over some link protocol could have been selected as the one required to be supported. But XML is chosen as it is a wide spread Composite structured data format. And a composite structured message will carry any message of choice. Also, TCP/IP (or possible UDP) is one of the most spread protocols with a high probability to be supported on any Darwinet node.

Darwinet network requirements

Definitions

A Darwinet network consists of Darwinet Nodes.

A Darwinet node is a single executable component that provides a Darwinet local API to Darwinet Applications and is able to communicate to other Darwinet nodes using the Darwinet Network protocol.

Each Darwinet node must be equal in networking capabilities

Each Darwinet Node must provide the same minimum set of network services to enable them to interoperate as equals. Nodes may have different bandwidth allowances from their ISPs, be in different countries, some of which may have national firewalls, etc. The network services they provide must still be defined to enable them to create a usable Domain.

Each node must be able to communicate, peer to peer or through an intermediate node, with any other Darwinet node in the network

This means that there must not be any special server—only-nodes or client-only-nodes or any other operation specialized nodes.

This ensures that the Darwinet network is a true p2p-network.

Each Darwinet node must provide operations that enables any node to be connected to any other node peer-to-peer

In a TCP/IP based network there may be nodes that are behind fire walls or other obstacles to direct node-to-node connection.

Two such nodes behind obstacles must be able to connect to each-other through a third Darwinet node outside the obstacles.

Such a node may be said to act as a Proxy of the other node.

Each Darwinet node must provide such a Proxy service.

Each installed Darwinet node must have a unique Id

The Domain must be able to identify node individuals in the network. This enables node identification, node authorization and node surveillance.

The Id may for example consist of one location descriptor and one instance index. This allows several nodes to execute simultaneously on the same machine. This also allows Darwinet nodes to execute in cloud services

Nodes must share information only on request and only peer-to-peer

Information exchange between nodes bust be on request only. And information must be peer-to-peer with another Darwinet node only using the specified Darwinet API's available.

- When a node has new information to share it connects to other nodes and requires them to accept the new information.
- When a node detects missing information it connects to other nodes and requires the missing information.

Nodes are allowed to but not required to connect to all available nodes of a domain

Nodes are allowed to but not required connect to all available nodes of a domain. This means that no mechanism of any Darwinet application must rely on running nodes being connected all the time.

The Darwinet network and network information model requirements

Definitions

With Information model we here mean how information is modelled in the Darwinet network.

Data must be isolated into a Domain

To enable Data to be protected and contained it must be stored and accessible within a Domain only.

For example: To make a company secure that any data that is supplied to the network does not reach any node that is not trusted by the company it is essential that data is contained within the nodes of a Domain.

This does not prevent Applications to reach data of several Domains if the user is a member of several Domains.

Data must be owned by an Application

All date must be owned by an application. And Darwinet is one such application. Also the Darwinet Domain controller may be such an application.

An Application must be able to use data in several Domains

A user that is a member of several Domains must be able to use data from these Domains.

An example of this is a Time reporting application used by a user that reports time to several projects or several customers. As data is contained within a Domain the only way for such an application to provide data to these domains is that the application may operate on data in several domains.

Properties and data must be structured in a tree of classes of data

The following tree of classes is proposed to organize properties of a Darwinet network.

Properties of a Domain are sub-classes of the Domain class.

Domain.Applications contains properties that define information of Applications to be used within the Domain.

Domain.Users contains properties that define users of the Domain.

Domain.Nodes contains properties that define the Darwinet Nodes of the Domain.

Domain.ActorRoles contains properties that define the roles of actors within this domain.

The Darwinet information security requirements

Overview

It must be able to use Darwinet nodes to share information that is protected. This enables companies, project groups, user groups etc. to set up and use a Darwinet domain of users and nodes with the same protection as other proprietary solutions.

It must be possible to require nodes to be authorized before allowing them access to the Domain

It must be possible to require a new node to be authorized by the Domain before it is allowed to access data and other nodes in the domain.

This enables control over the locations to which the domain may be allowed to grow.

It must be Possible to exclude nodes from a Domain

It must be possible to exclude nodes from a Darwinet Domain. This enables nodes installed at locations which should not any longer be part of the domain to be excluded.

This may happen for example when a lap-top is sold or handed over to another user. Any Darwinet nodes installed on that lap-top may then be excluded.

It must be Possible to require domain data to be destroyed when node is excluded.

It must be able to impose that data in an excluded node of a domain is destroyed by the node when the node is excluded.

This enables the Domain to prevent data to remain in nodes that are no longer members of the domain.

It must be Possible to require a node to destroy Domain data if node is not authorized

It must be able to impose that data in unauthorized nodes of a domain is destroyed by the node.

This enables the Domain to prevent data to remain in nodes that are no longer members of the domain.

It must be Possible to require nodes to die if not regularly re-authorized

It must be possible for a Darwinet domain to require all nodes to check in and re-authorize. And if they are no longer accepted or fails to gain access to the domain they must die and destroy all data of the node for that domain.

Note 090614. It is not yet decided if a node may be member of several domains. If they are – then the node may not die until it is not member of any domains. But data of domains in which it is not a member must still be destroyed.

This enables data to be destroyed in nodes that are no longer members of the domain but that has not actively been destroyed.

The Darwinet performance requirements

The Darwinet data exchange framework must allow optional data compression option

To enable data exchange performance between Darwinet nodes it must be possible to add optional data compression modules into the data trail. Such data compression must be allowed to be optional to enable adding and change to existing networks.

Data exchange using XML may use ISO/IEC 24824-1 (Fast Infoset) data compression.

The Darwinet marketplace requirements

It must be possible to sell Applications within the Darwinet

Consider to use the same business model as Apple store.

TBD.

It must be possible to sell information within the Darwinet

It must be possible to sell information within the Darwinet.

An example of this may be a user may that subscribes to use of a weather forecast service through a Darwinet application.

It must be possible for third party providers to develop and sell Darwinet applications to Darwinet domains

Consider applying a “Service provider” design pattern to enable “Darwinet Application Provider” implementations over the public web.

TBD.

The Darwinet applications distribution requirements

Overview

TBD.

Platform independent Darwinet Mechanisms

Darwinet mechanisms overview

Platform independent mechanism definitions

A Darwinet network is made up of functionality grouped into sets called mechanisms. A mechanism has a purpose and is made up of a set of functions that cooperates in a defined way to create the mechanism.

Most mechanisms shall have a platform independent definition. Some mechanisms may not be available on some platforms.

An Application interacts with a View

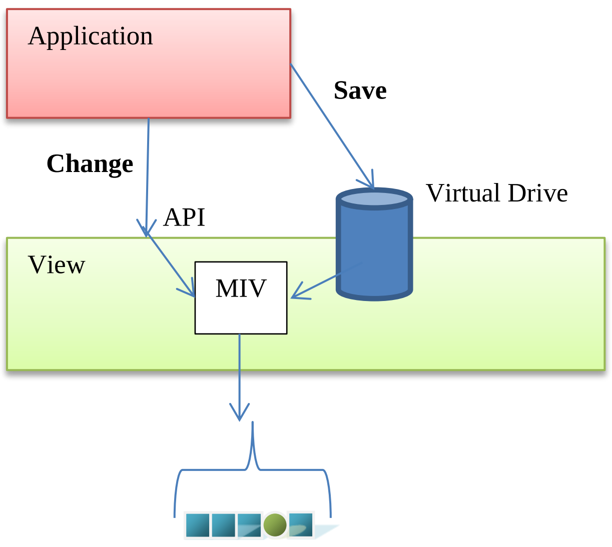
An Application may access a Darwinet Domain through a view. The Darwinet view defines what evolution branch and the point in evolution history of the interaction.

Normally the point in history of the interaction is "current time".

The Darwinet view may provide two interfaces to the viewed data in the Domain:

- The Darwinet view API (in platform available technology)
- A Darwinet view virtual drive (if platform provides services to provide a virtual file system)

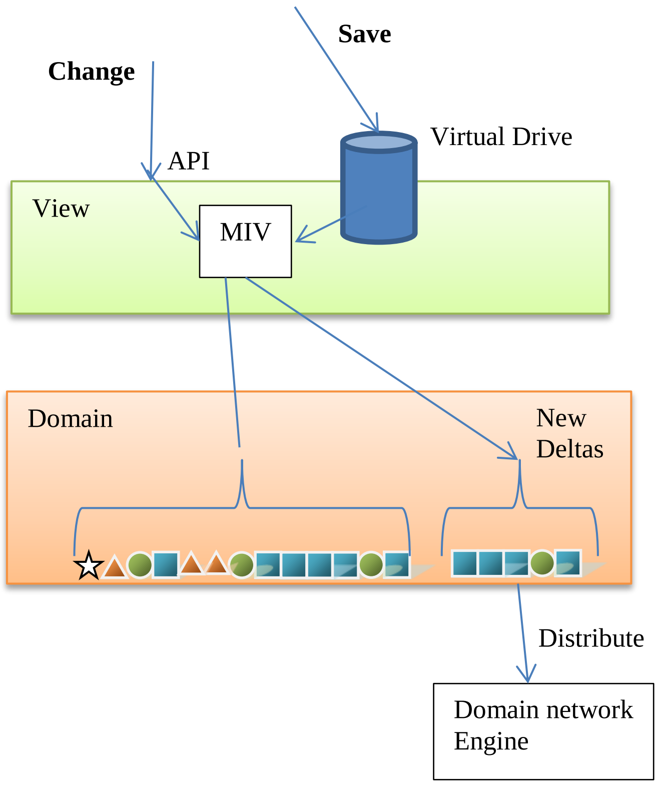
The View detects MIV changes



When the Application makes changes directly to the MIV, or indirectly by saving a file (represented in the MIV by the Darwinet Virtual Drive), the View creates a series of changed (deltas) to the MIV.

The Domain distributes MIV changes to the Domain member nodes

The Domain receives the changes in the form of new Deltas and initiates synchronization with the other nodes of the Domain.



Note: Other nodes may be offline. Those nodes will receive New Deltas when they come online and report their current state.

Note: Each node may use a different messaging protocol. The Darwinet network engine will try and distribute the new deltas using the most efficient protocol available at the moment. (*Messaging with portable devices may require protocol optimization depending on the available service at the nodes current location*).

The MIV (Model, Instance, Value) Delta mechanism

The smallest possible atom of data within a Darwinet Domain is a Delta or Change. And the Change may be to the Data Model (M), Model type instances (I) and instance values (V).

The mechanism for processing, storing, retrieving and provide services for this is the Model, Instance, Value (MIV) Delta Mechanism.

The MIV evolution history mechanism

Domain contents evolution introduction

One basic concept of a Darwinet Domain is to provide mechanism to enable data and applications to evolve through time without having to upgrade or install anything as a separate action.

Evolution happens by introducing changes to current domain state at a node. The changes are then reported to all other nodes of the domain and they change accordingly.

Normal changes are changes to files and repository data values.

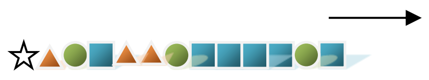
But Darwinet Application developers may release an upgrade by distributing an appropriate change to the current version of the Application.

MIV evolution introduction

Member nodes of a Darwinet Domain exchanges deltas (changes) to the Data Model (M), Instances of Model types (I) and Values (V). In this way the MIV state evolves and a Darwinet vide may keep track of these changes and synchronize so that each view in the Domain is correctly synchronized.

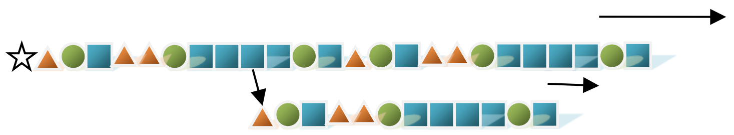
Evolution trails of MIV changes (∆MIV's)

All changes are distributed as Model, Instance and Value deltas and affects files or data. As Applications are represented as executable files in the File system they too are changed using Model, Instance, and Value changes to the file system and to the File contents.



Evolution branch

It is possible to branch an evolution line into two separate lines of evolution. A branch shares data with the line from which it has branched until the data it uses also branches. In effect, a branch is an alternative route to travel with a view that differs in at least one link of the change chain.



MIV Changes (Delta MIV)

Each Delta shall contain

- Creator (source of creation)
- Index (Unique cross the whole Domain)
- Actual Delta

The Creator shall define

- Domain of creator node
- Node (Creator of Delta)
- User (Optional if Delta is initiated by user action)

The Delta shall be one of

- Model Delta
- Instance Delta
- Value Delta

The Model Delta shall contain

- Target of change (Id of the model artefact that has changed)
- The actual change one of
    - Add a type to target
    - Remove a type from target

The Instance Delta shall contain

- Target of change (Id of the variable instance that has changed)
- The actual change one of
    - Create (Add a new instance to target). Id of new instance provided.
    - Destroy (Remove an existing instance from target). Id of instance to remove provided.

The Value Delta shall contain

- Target of change (Id of the value that has changed)
- The actual change one of
    - Fundamental Data Type Value Change (Change to a variable of a Darwinet fundamental type)
    - Compound Type Value Change (Change to an instance of a Darwinet Compound type)

Note: Using terms from the C/C++ language and others

- *The Model Delta corresponds to a type declaration.*

- *The Instance Delta corresponds to a value definition.*
- *The Value Delta corresponds to manipulating a variable value.*

Delta Synchronization

To synchronize Deltas created by the peers in the network each Delta needs to carry enough information to make this possible.

Each node keeps track of a local index so that each delta in the evolution trail gets a unique index. The Index includes the Node ID so that the index is truly domain unique. In this way the Delta index identifies both the production Node and the order in which the Delta occurred in the evolution history.

```
Node 1: Index {1.1 … 1.n}
Node 2: Index {2.1 … 2.n}
…
Node N: Index {N.1 … N.n}
```

Each Delta defines its predecessor so that each node in the domain may synchronize the order of the Deltas in the evolution trail.

$\Delta_{1.1}$, $\Delta_{2.1}$, $\Delta_{1.2}$, $\Delta_{1.3}$, $\Delta_{2.2}$, $\Delta_{2.3}$ …

Each delta also defines the miv it targets. The miv is identified with a miv identifier and the miv state to which the Delta shall be applied.

When synchronization of Deltas shall be made the following conflicts have to be resolved among the receiving peers in the Domain..

- Two deltas defines the same predecessor = ordering conflict.
- Two deltas targets the same miv (id and state) = change conflict.
- A received Delta defines a future predecessor = missing Deltas

Ordering conflicts shall be expected as commonplace in Darwinet networks (each node extends the evolution trail locally and several nodes may extend from common predecessors before intra-node communication enables synchronization). One problem the synchronization algorithm must face is to effectively recognize ordering conflicts that does not result in change conflicts.

An ordering conflict may be resolved by first branch the evolution trails at the shared predecessor Delta.

Node 1 Trail: $\Delta_{1.1}$, $\Delta_{2.1}$, $\Delta_{1.2}$, $\Delta_{1.3}$, $\Delta_{1.4}$, $\Delta_{1.5}$
Node 2 Trail: $\Delta_{1.1}$, $\Delta_{2.1}$, $\Delta_{1.2}$, $\Delta_{2.2}$, $\Delta_{2.3}$, $\Delta_{2.4}$

The detecting node places the conflicting trails into separate branched trails by prefixing their index them with a branch index. The Branch Index is the index of the shared predecessor Delta followed by the Node Id of the first Delta in each branch (the producer of the Delta that caused the conflict that resulted in the branch). This makes the Delta index unique also for branched trails and the index identifies the branch origin.

Node 1 Trail: $\Delta_{1.1}$, $\Delta_{2.1}$, $\Delta_{1.2}$, $\Delta_{1.3}$, $\Delta_{1.4}$, $\Delta_{1.5}$
Ordering Branch: $\Delta_{1.2.1.1.5}$, $\Delta_{1.2.1.1.4}$, $\Delta_{1.2.1.1.5}$

Node 2 Trail: $\Delta_{1.1}$, $\Delta_{2.1}$, $\Delta_{1.2}$, $\Delta_{2.2}$, $\Delta_{2.3}$, $\Delta_{2.4}$
Ordering Branch: $\Delta_{1.2.2.2.2}$, $\Delta_{1.2.2.2.3}$, $\Delta_{1.2.2.2.4}$

The goal of the conflict detecting node is to try and merge these order conflict branches directly. It does this by treating each branch trail as a transaction and check if the end MIV state when applying each transaction to the shared predecessor MIV state is the same. If it is the ordering conflict does not have an end conflict and the node may create a Merge Delta in the Domain evolution trail. The Merge Delta is a transaction carrying two evolution trails (the branched trails) that when applied to a MIV state results in the same end state.

$MIV_n + B_1 \Leftrightarrow MIV_n + B_2$; B_1 and B_2 being the branch evolution trails (transactions).

Change Conflicts due to Delta operations to the same miv target (id and state) must also be handled by branching the evolution trail at the Deltas that targets the same miv (id and state). In this case the Branch Index is the index of the shared Delta target (the trail is shared until that after that Delta) followed by the node Id of the first delta in the branched trail.

Node 1 Trail: $\Delta_{1.1}$, $\Delta_{2.1}$, $\Delta_{1.2}$, $\Delta_{1.3}$, $\Delta_{1.4}$, $\Delta_{1.5}$
Branch: $\Delta_{1.2.1.1.5}$, $\Delta_{1.2.1.1.4}$, $\Delta_{1.2.1.1.5}$

Node 2 Trail: $\Delta_{1.1}$, $\Delta_{2.1}$, $\Delta_{1.2}$, $\Delta_{2.2}$, $\Delta_{2.3}$, $\Delta_{2.4}$
Branch: $\Delta_{1.2.2.2.2}$, $\Delta_{1.2.2.2.3}$, $\Delta_{1.2.2.2.4}$

Note that a change conflict always results in an ordering conflict as it may only occur on separate nodes. On the same node the Domain manager will ensure that there exists only one instance of a miv (id and state) and synchronize change operations so that each mivs state is updated each time it is changed and a delta targets it.

User interface Issue: How should we provide a view of the branch to the user and/or the application? A branched file should be replaced with a folder containing the branched file. But how about branched application data? How do we enable the user to see that a branch has been made? How do we enable the user to merge the data when the branched data is operated by an application?

Branching mechanism

Each node must be associated with a branch index bi(node) and every node of the Domain must calculate the same branch index of any node.

When a Node receives a Delta that conflicts with another Delta it must create a new Delta with the same content but with a branching index and distribute it.

A Branch Delta always replaces the original unbranched one. (*Note how this is true regardless of what node created the replaced delta. The receiving node will receive a branching index from other nodes as they detect conflicts and the original delta is obsolete regardless of if there are just one or many conflicts*).

The Time Travel mechanism

The Time Travel mechanism enables a Darwinet domain to be instantiated and operated at any point in time through its history of evolution.

Domain View Time Travel

Time Travel is provided by enabling a View to be fixed at any point in the time of domain evolution.

A Darwinet Application may provide a user interface control to allow the user to roll back the application and the viewed data to a previous state.

It is the Darwinet View that provides Time Travel Services to an Application.

The Application signals to the View to view a specific point in time. The View then moves from current point in time to the required one. In the travel the View generates Changes and sends them to the Application to update the GUI as the changes are made.

Time Travel over Application change barriers

An Application may request a Domain view to move back to a point where the Application itself changes.

Basically it is the Domain view that is responsible to shut down current version of the Application and then start the new version of the application.

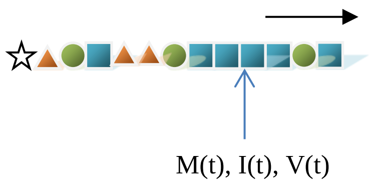
But that may not be possible on all platforms.

- In Microsoft Windows the Domain view may be provided by a virtual drive in the File Explorer. When the view needs to replace an executing application it shuts down current application, assembles the new Exe-file and then executes the new one.
- In OS X…
- In Apple iOS…
- In Android…
- In UNIX…

Well. This needs a little more thinking!

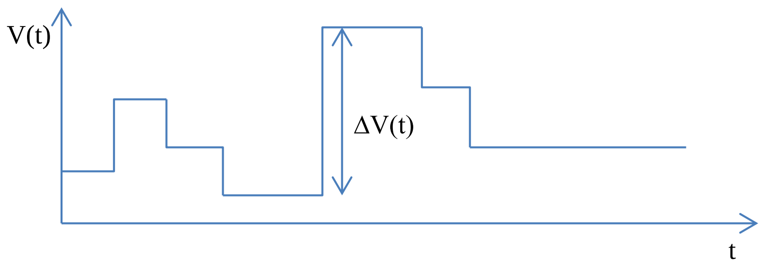
Value as function of time Mechanism

The "Value as function of Time" mechanism provides means to an Application to retrieve the contents of a value at a specific point in time. This service is provided by the Domain view and it is based on the Evolution trail of a specified value instance.



Darwinet provides a mechanism through which a Model (M), Instance (I) or Value (V) state may be queried at a specified point in time (t), i.e. M(t), I(t) or V(t)

An Application may use this mechanism to show a user of an Application how the data handled by the Application has changed over time.



Example of when this is of interest is in a Project Plan tools. A Project Planning tool may provide a history view where the time estimation, project resource allocation, project scope and so on is plotted over time. This may provide valuable feed-back to what has happened over the project life span.

The Peer-to-Peer mechanism

Darwinet P2P introduction

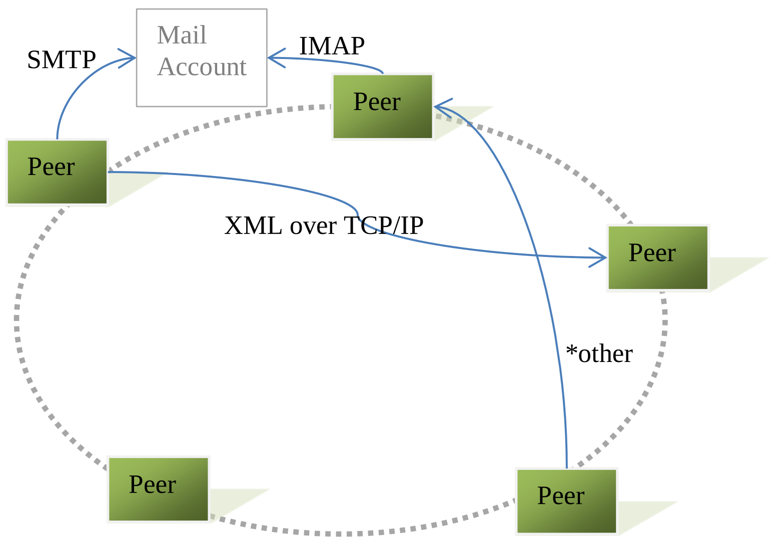
A Darwinet domain network is made up of Darwinet clients that interconnect as equals in a Peer-to-Peer network.

The Mechanism to make this work is called the Peer-to-Peer mechanism.

The Mechanism requires services provided locally at each node and services provided to and from other nodes of the network.

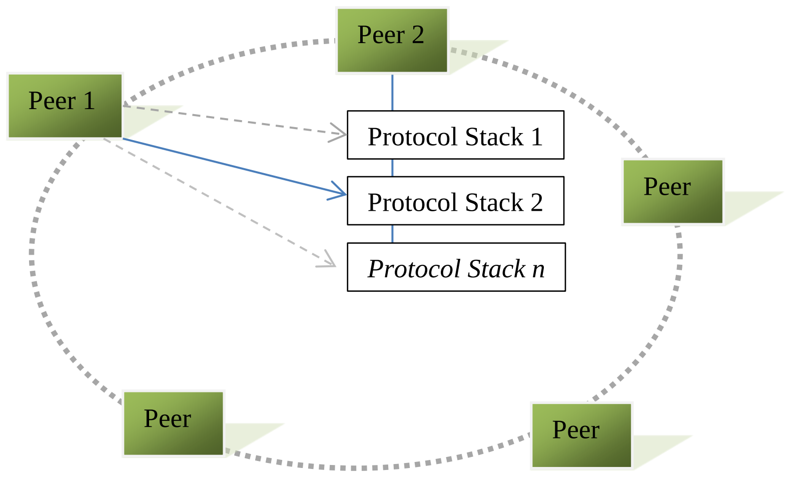
P2P using different messaging protocol stacks

A Darwinet node may be configured to use a set of messaging protocols through which it is accessible. And nodes of a Domain may use a mix of protocols as the accessibility changes for example with the node location.



P2P internode messaging protocol stack option

A Node shall select the most appropriate Messaging protocol stack supported by the other node when connecting to it.

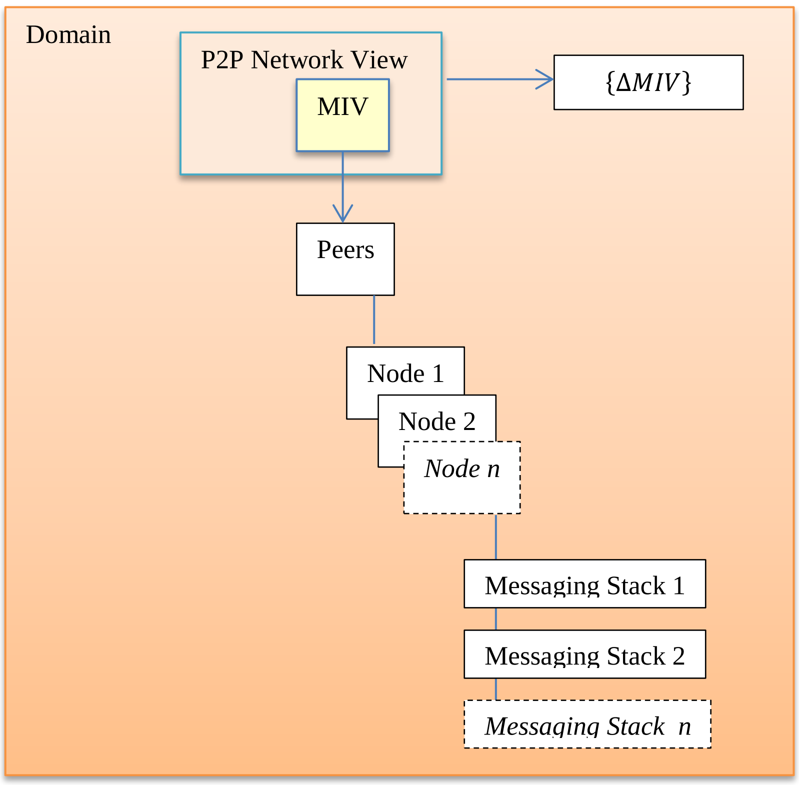


Node Protocol Stack model

The Protocol Stack model shall be layered.

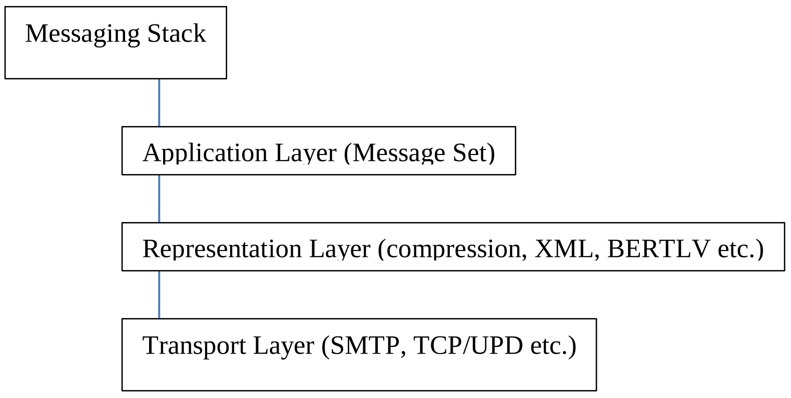
- It may contain an Application layer defining what Darwinet P2P messages to exchange. *This enables new messaging sets to be introduced over time.*
- It must contain a representation layer that defines how exchanged messages are to be represented (e.g. as XML, BERTLV, Text in Email etc.). *This enables new representation forms (e.g. different types of compression algorithms) to be introduced over time*.
- It must contain a Transport layer that defines how the data is transferred to the receiver (e.g. SMTP (Email), TCP, UDP etc.). *This enables new data exchange protocols to be introduced that best suites new node platforms and portable device service providers (e.g. use some push notification service of iOS, Android, UNIX etc.).*

Node Messaging Protocol stack support modelling



A Darwinet Domain instance shall create a P2P Network view with a MIV modelling the current state of the Domain. The State shall contain aspects like Users, Nodes, and Applications etc. that are members of the Domain.

- The model shall contain a list of Peers that are members of the Darwinet Domain view.
- Each Peer shall define the Peer Node.
- Each Node shall define a set of incoming Messaging Protocol stacks it implements for incoming connections.

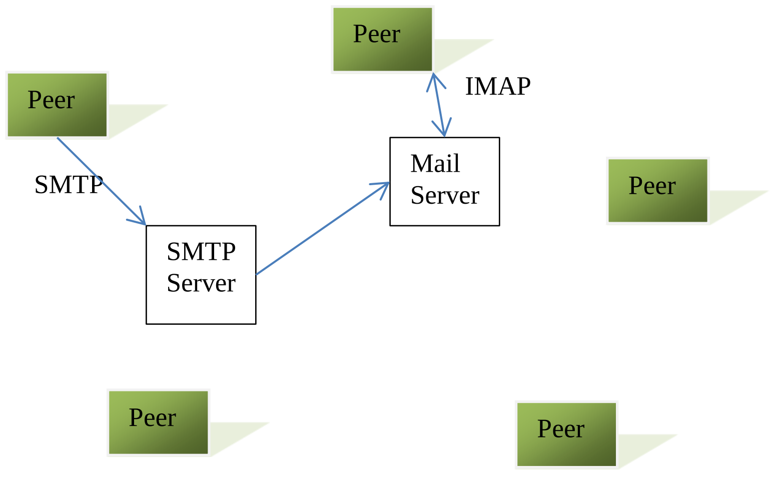

Messaging Stack
Application Layer (Message Set)
Representation Layer (compression, XML, BERTLV etc.)
Transport Layer (SMTP, TCP/UPD etc.)

The Messaging Stack shall define each required layer of the stack.

P2P using e-mail

A Darwinet node may interchange data with other nodes using an available e-mail account. This may prove to be an easy way to introduce a Darwinet network into an existing infrastructure where users already have e-mail accounts.

A company may go for this option to allow nodes to interchange data through fire walls in a safe way where other options are cut off. A company may apply data exchange supervision may adding a Darwinet node to the mail server to supervise that the data exchange complies with an existing policy.



A Darwinet node accessible in the P2P domain through e-mail shall be configured to use an e-mail account accessible using IMAP. IMAP enables the node to use an e-mail account of a user. The node will fetch only e-mails that contain Darwinet node messages. A user sharing account with a Darwinet node has to get used to receive e-mails carrying Darwinet messages and just ignore them. The node will use the IMAP access to remove those mauls when they are consumed.

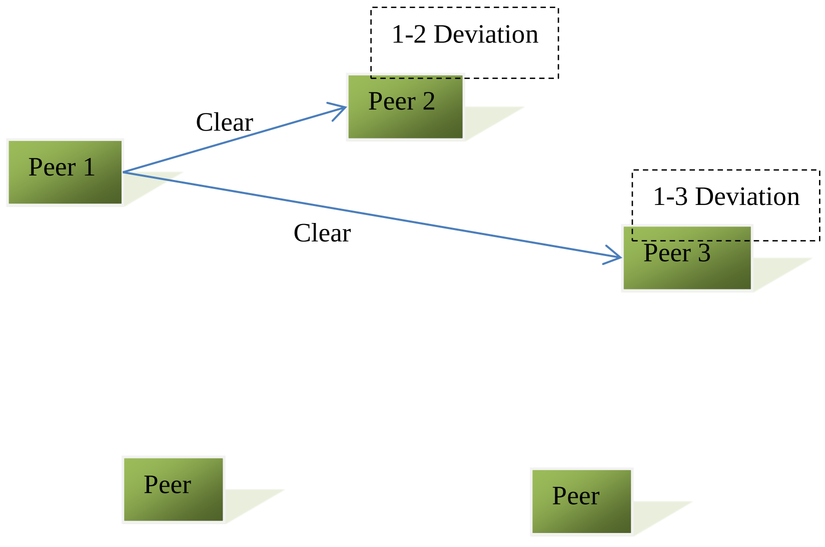
A node assigned a dedicated e-mail account (not shared with a user) may access the account using POP3.

MIV cross node synchronization

Changes to the state of the main and branched MIV shall be synchronized between the nodes of the domain in a process called State Clearing and State deviation settlement.

Clearing MIV states between nodes

The result of a state clearing process shall be a record of the state differences for node pairs involved in the clearing. Any detected differences shall be settled by a state difference settlement process.



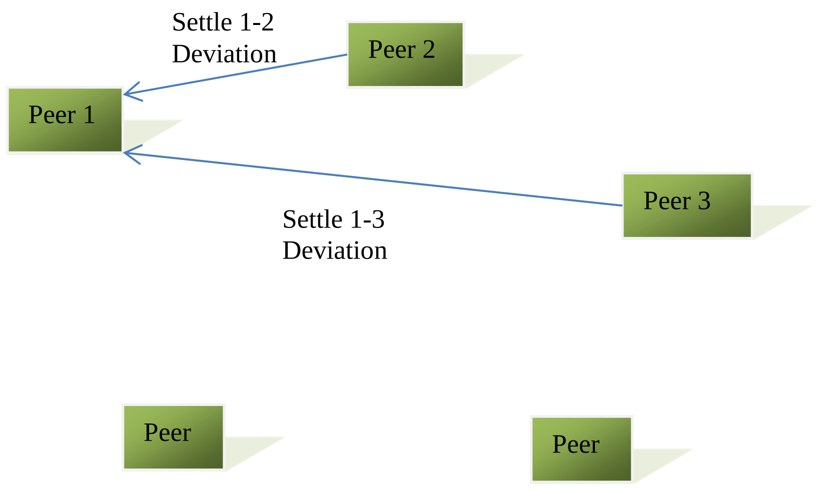
The State Clearing process shall detect conflicting deltas and create branches when needed.

Small domains may always try to clear its state with all available online nodes for each state clearing initiative.

Large Domains shall not let nodes clear its state with all other nodes. In such domains nodes must clear its state only with a sub-set of randomly selected nodes in the Domain. It must then also initiate regular state clearing. In this way new Deltas may ripple through the Domain over time.

Settle MIV states deviation between nodes

A state deviation between nodes of a node pair shall be settled by a settlement operation.



In the settlement operation the node that has newer deltas must send them to the node that lacks them.

A node that produces local changes (deltas created by the node) shall settle its new state with currently online nodes.

The state deviation settlement shall handle branching when required.

Darwinet P2P Message

Protocol independent Darwinet Message Definition

The actual representation of a message depends on the messaging stack used in the interaction.

A Message shall be a Component of fields, each field being a primitive field or a composite of new components

- <Message> :== <Component>
- <Component> :== <Tag><<Leaf Field> | <Composite Field>>
- <Leaf Field> :== <Value>
- <Composite Field> :== ARRAY OF <Component>

Examples of standard representations of Components are XML and BER (Basic Encoding Rules) but any representation may be used as long as sender and receiver agree on the interpretation.

A Message shall contain

- Sender Node Id
- Receiver Node Id (enables routing of messages)
- Message content

Message Content shall be

- Delta
- … other (Extendable as needed)

The Service Provider Mechanism

Each node of a Darwinet Network implements a set of required services needed to enable the Darwinet network to operate as defined.

In Addition to this, the network allows optional Service Provider nodes to be allowed to operate within the network. These optional nodes may be gateways to web-services to provide for example Storage services, Payment services, Application store services, data market services and so on.

The Service Provider mechanism defines how these services may be integrated into the network. It defines Service Provider restrictions, supervision and control.

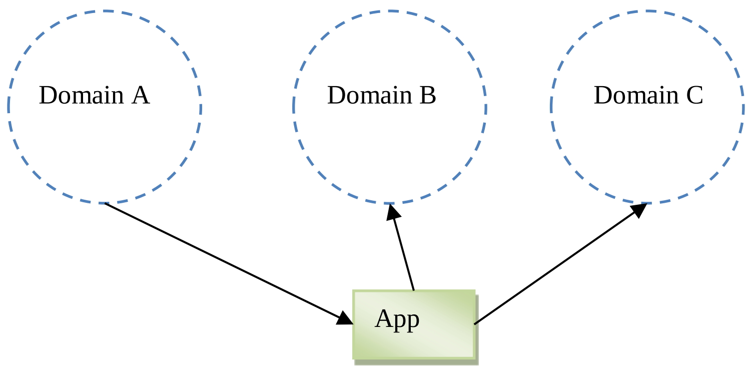
The Domain bridging mechanism

An Application may need access to data in multiple Darwinet Domains.

To see why this is important consider the following applications:

- Corporate invoicing application in economics department domain needing time report information from division domain.

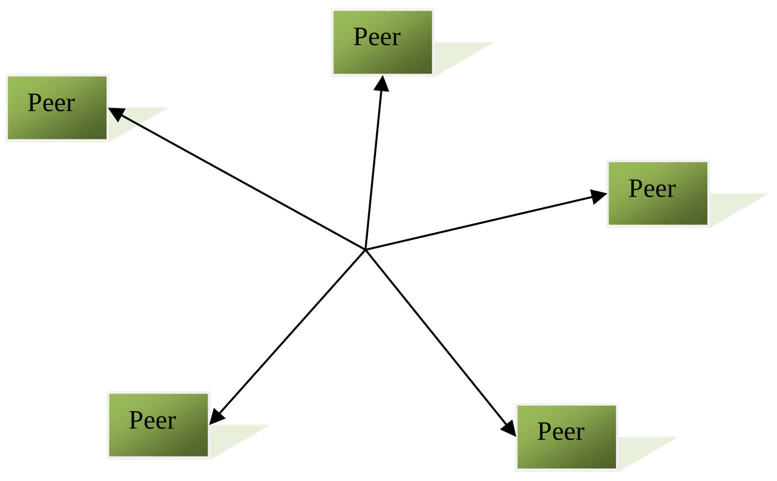
- Time report application of a customer needing access to consultant time report data.
- Social network application needing to share information in your relative’s domain with the members of the family domain.



The Darwinet network

Darwinet network introduction

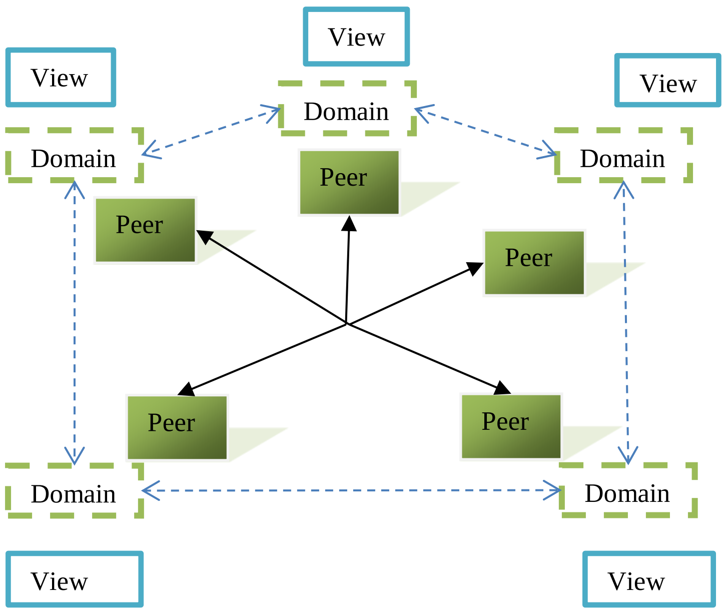
A Darwinet network is a network of symmetrical peers exchanging data within a defined domain.



The network operation is performed by the Darwinet clients of the network.

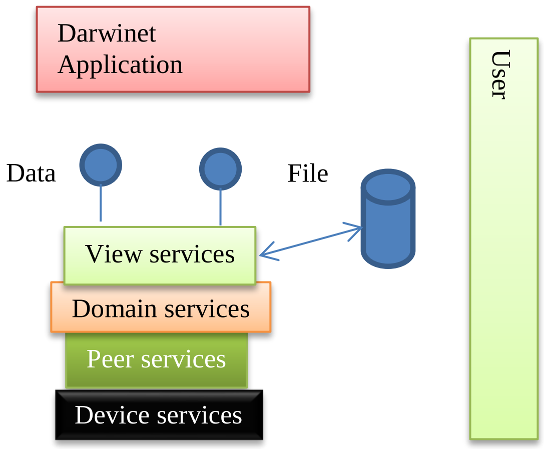
Networking operations are performed on these basic concepts:

- A Darwinet Domain is a defined set of Users, devices and peers.
- A peer is implemented by a Darwinet Client.
- The Network exchanges data between Darwinet peers.
- Each Physical or virtual device is allowed to instantiate multiple peers.
- A View exposes data and services of a Domain at a specified View Point.
- Each Peer may instantiate multiple views.
- The Peer distributes synchronizes its Views locally.

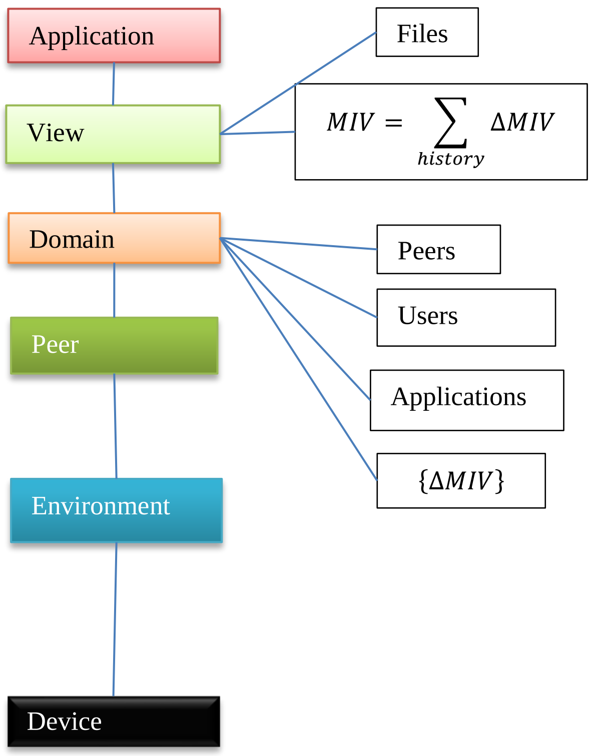


The Darwinet Peer service stack

Each Peer implements the Darwinet Domain services as a local stack.



Each layer maintains a subset of states.



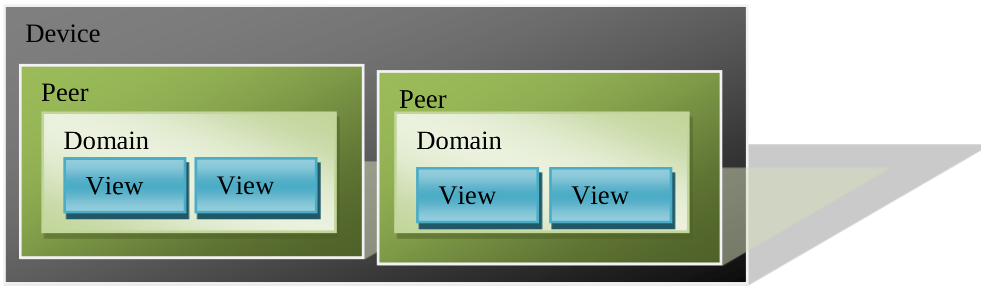
Enabling sandboxed applications to interact

- The network must allow sandboxed Darwinet Applications to interact within a Domain. They do so by acting as separate Peers on the same device. For example, iOS applications are sandboxed.

Domain view instance management

A Darwinet Domain controls what Domain views that are allowed. The Domain view instance control checks the:

- Physical Device where the Instance is created
- The user creating and using the instance
- The data and services allowed on the Device
- The applications accessing the view



Bridging domains

Domain bridging introduction

See “The Domain bridging mechanism”.

The Darwinet Client

Darwinet Client overview

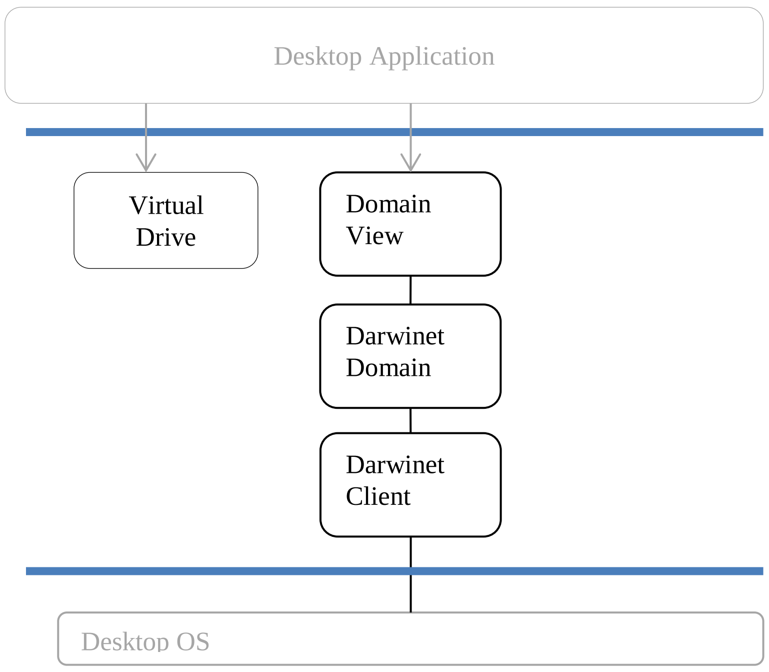
As a Darwinet network is a Peer-to-Peer network it needs a client application that executes at each node of the network.

Now, as the goal is to provide any device on any platform the possibility to interact within a Darwinet domain, the clients may have to be implemented and provide Darwinet services quite differently.

Desktop Darwinet Node (Windows, MAC, UNIX etc.)

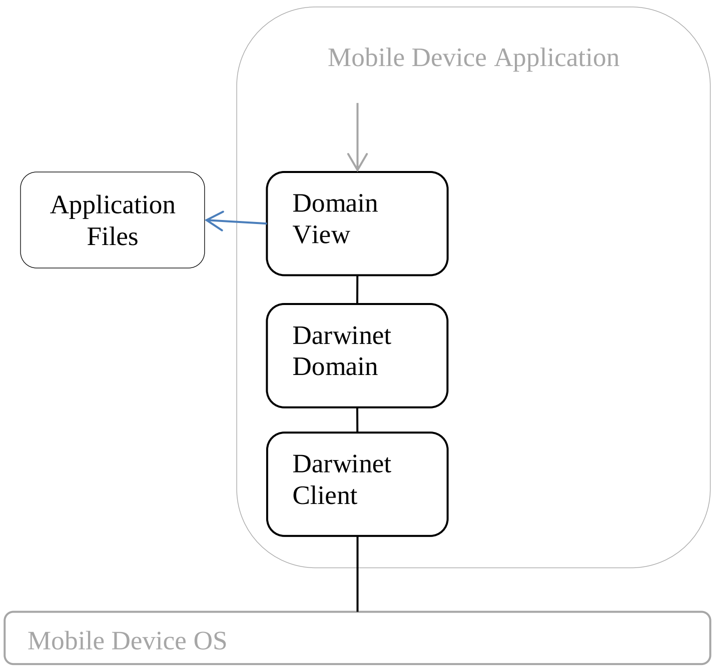
A Darwinet node hosted on a desktop client may provide a rich user experience and interact with both Darwinet applications as well as desktop applications.

On the major desktops like Windows, MAC and UNIX desktops the Darwinet client shall have a common set of services.



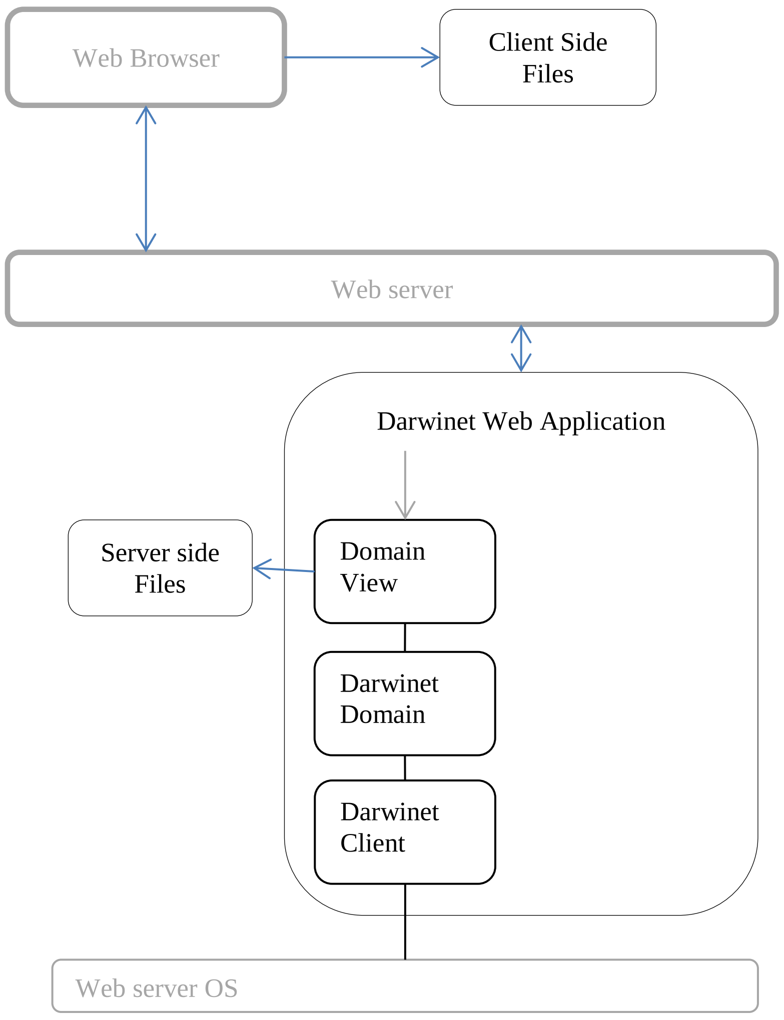
Mobile Device Darwinet client

Mobile platforms like iPOS, Android, Windows Phone may require each application to host a complete Darwinet Client in itself. In effect each application them becomes a Darwinet Node of its own.



Web server hosted Darwinet node

A Darwinet Domain may be hosted on Web servers by using Web server Darwinet clients. In this way it is possible to operate a Darwinet Domain entirely through web browsers and use no local Darwinet Client at all.

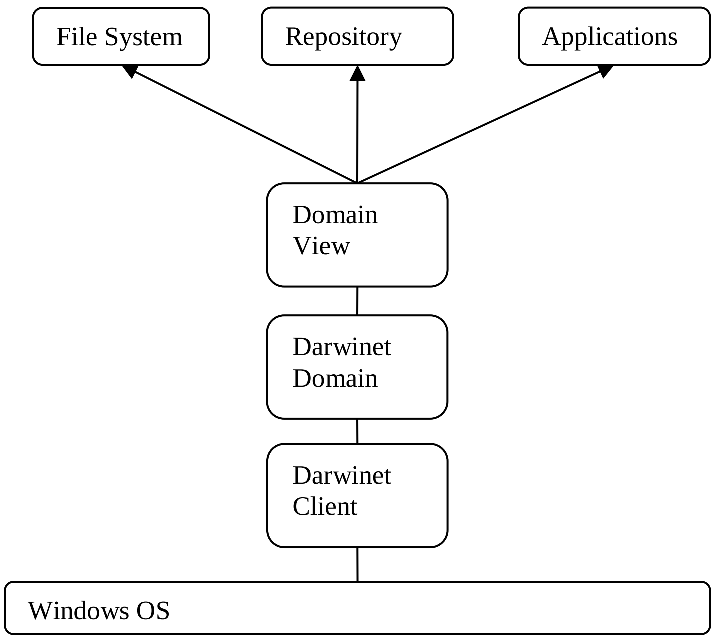


In this case all Darwinet mechanisms have to be implemented into the code executed on Web access.

- A user interface to a view into the Darwinet Domain including login, view selection and so on unless hard coded by the web access.
- The messaging protocol with other nodes may be restricted by the available services of the Web browser. But SOAP, e-mail and other common Web protocols may be used and must then be supported by the other nodes of the Domain.

If the web hosted nodes are seldom accessed the amount of work needed to update the node may slow down the user experience. The user interface should show current state and then update the state while the node is “catching up” (clearing and settle its state with the Domain).

The Windows Desktop Darwinet Client



The Windows desktop Darwinet client may be an ordinary windows application that, when run, provides a user interface to open a domain.

The Domain has a default view showing the current domain state through a domain view.

The Domain view in turn exposes three standard interfaces.

1. The File Handler exposing the files of the domain in the state defined by the view.
2. The data repository exposing the current key, value tree of the domain as defined by current recorded Model, Instance, and Value delta stream.
3. An optional Application list interface of the Domain. Through this interface Darwinet applications may contact each other for data and service exchange.

The Darwinet Domain View

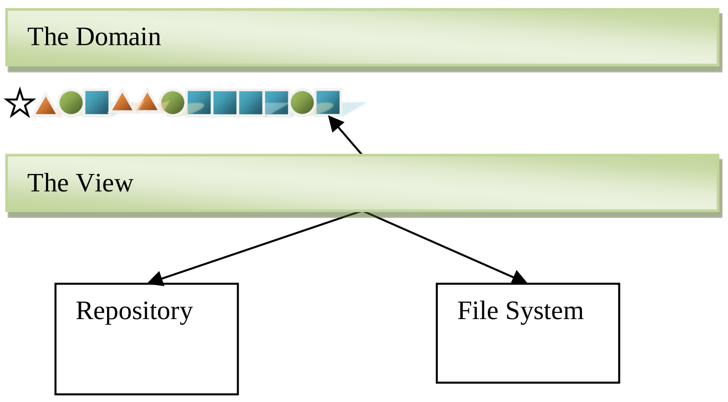
Darwinet Domain View introduction

Any interaction with a Darwinet Domain is done through a Darwinet Domain View. The View defines:

- A viewpoint in time
- A subset of what data is exposed
- A data and services “world” that maps Domain Data into a graphical user interface view-port and Application API viewport.

The data exposed by the View is:

- The Domain Repository contents as defined by the View Point.
- The Domain File System as defined by the View Point.



The Domain contains an evolution line of Model, Instance, Value change events. In this figure visualized as follows:

- ☆ denotes a creation event
- ▲ denotes a model change (Add or delete a type)
- ● denotes an instance change (create or destroy a value)
- ■ denotes a value change (add or subtract a value)

The View defines a point in the Domain history. It creates the Domain repository and the File System at the state defined by the View Point. The view creates this state by integrating the Model, Instance, Value change history up to the View Point.

The Darwinet Domain view repository

Darwinet Domain View Repository introduction

The Darwinet Domain View Repository is where all the data of a specific domain view “lives”. The repository contains the current value instances stored in a Key-path, value manner as defined by the current Model.

The repository is built for the view by integrating the Model, Instance, Value change history for the view.

Darwinet Applications

Darwinet applications overview

A Darwinet Application is an application that may be brought to execute on the contents of the repository of a Darwinet Domain View.

To enable this on multiple platforms is quite tricky and requires different schemes.

Darwinet Applications on iOS

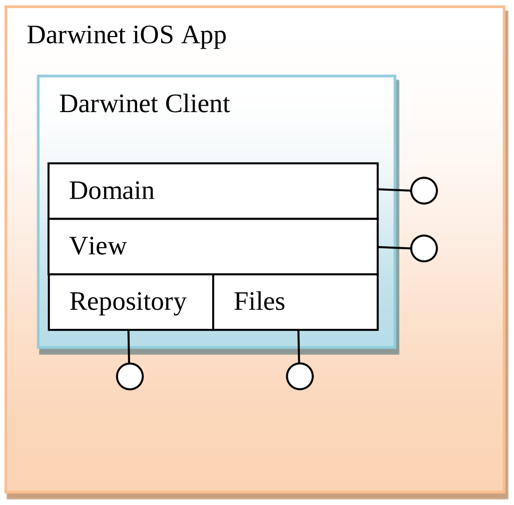
Darwinet Applications on iOS introduction

The iOS implements “sandboxing” in the sense that each iOS application really does not share any data or file system with any other application. Instead applications may interact over TCP/IP using a local URL-scheme.

So how should an iOS-application access the repository and the file system of a Darwinet Domain view?

Stand-Alone iOS Darwinet application

An iOS Darwinet application is a stand-alone Application. This means it acts as a separate Peer in the Domain.



Note: This means that each app on the iOS device may mirror the same data within the Domain. The amount of data actually mirrored should be minimized to the data actually shared.

The Darwinet domain development and deployment tools

Darwinet development and deployment tools overview

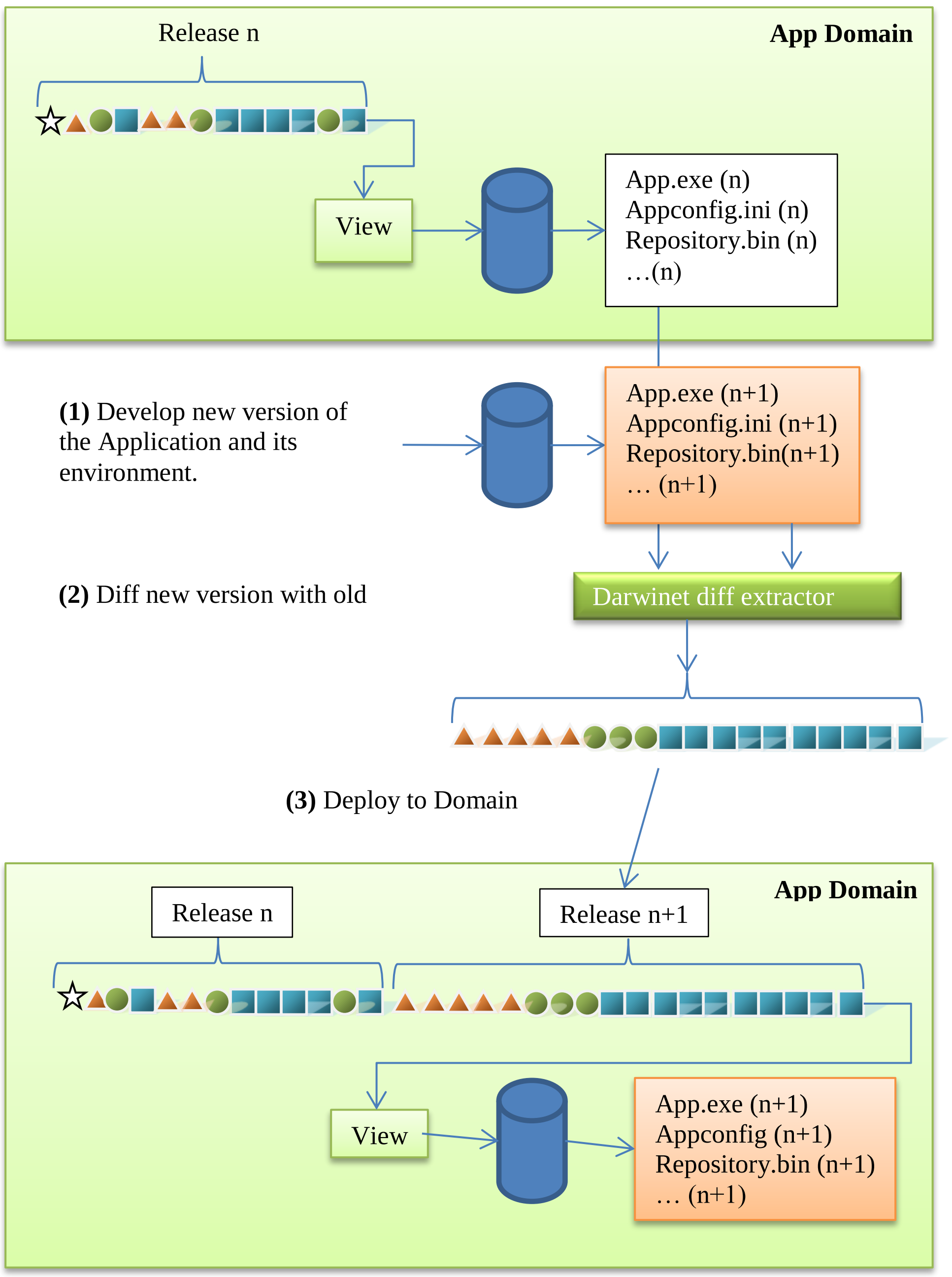
Some changes to a domain may need to be deployed as a pre-defined set of changes where all intermediate changes have been eliminated.

- When deploying a new or an updated application.
- When deploying an update to a sub set of the environment of one or several applications.
- When deploying large architectural changes to a domain data model or state.

The Darwinet Environment diff extractor

The Darwinet Diff Extractor overview

The Darwinet Diff extractor is a tool that calculates the shortest evolution path between two views. It may be used to deploy an update to an application and its environment to a Domain. The Old Application and its environment are in one view and the final state of the developed application and its environment are in another view. The Darwinet diff extractor is then used to create the shortest evolution path from current application release view and the new updated one. The Evolution path is then deployed to the Domain causing the default Domain view to evolve to the new updated application and its environment.


Release n
App Domain
View
App.exe (n)
Appconfig.ini (n)
Repository.bin (n)
…(n)
(1) Develop new version of the Application and its environment.
App.exe (n+1)
Appconfig.ini (n+1)
Repository.bin(n+1)
… (n+1)
(2) Diff new version with old
Darwinet diff extractor
(3) Deploy to Domain
App Domain
Release n
Release n+1
View
App.exe (n+1)
Appconfig (n+1)
Repository.bin (n+1)
… (n+1)

The Darwinet Local API Independent Platform model

Overview

A Darwinet network must be specified in a platform independent way. This enables Darwinet nodes to be implemented on any web-connected device.

Each web-connected device may have its own platform, framework and operating system. And the installed Darwinet node executable may implement the application API using any suitable framework on that platform. This must not disable these applications to interact with other Darwinet nodes.

TBD.

Roles and actors of the Darwinet network

- A Darwinet node. This is a single executable with a unique node Id that provides Darwinet applications with access to other nodes in a Darwinet network.
- A Darwinet Domain. This is a collection of defined users and nodes. Only users and nodes that are members of the Domain may share information.
- A Darwinet User.
- A Darwinet Application. This is an executable that uses a Darwinet node to share information with other Darwinet nodes.

The Darwinet network architecture

The Darwinet service provider holds user accounts and provides Darwinet services to Darwinet domains.

Services of the Darwinet service provider are:

- Provides the Darwinet framework nodes that are applications that may run on the platform of the Darwinet node and enables communication with other Darwinet nodes.
- Provides Darwinet framework updates and configuration.
- Provides payment services to Darwinet applications to enable them to charge for provided services. The service provider forwards payments to the application developer.

    - Service providers are allowed to charge 30% for their services and pay the rest to the application developer (the apple way)

A Darwinet service provides service a Darwinet domain. This domain is base domain in which users may create user domains of their own.

A user domain is a set of users set up to work together using a defined data domain and a set of Darwinet applications.

Darwinet nodes data exchange scenarios

Overview

The Darwinet must support asynchronous evolution of the data in the network domain. One user may change data off-line at one node of the network while another one changes data on-line at another node. A third user at a third node will then at some future time receive the changes made online and off-line and end up with the correct state of the data. Also – the third user must be able to work with known state of data until the changes made by the other two users are received.

The solution to create this behavior is the following:

- All changes are recorded as changes (diffs) from current state and sent to the other nodes as changes to the defined state.
- Changes may be:
    - Data model changes (declaration changes)
    - Data instance changes (definition changes)
    - Value changes (assignments changes)
- All changes must be reversible based on the change information only. This enables backward changes to be created from received forward changes. Local storage may be based on saving latest state and a series of reverse changes although changes are exchanged from older to newer state.

Darwinet network system scenarios

Creating the first node of a user domain

A user that wants to create a Darwinet user domain starts of by creating the first Darwinet domain. The steps are as follows:

- Contact a Darwinet service provider; register an e-mail address to get an account.
- The Darwinet service providers mail a “Darwinet node seed” to this mail address. The seed may be one of the following:
    - A file with a configuration of the Darwinet node.
    - A small executable that will download the Darwinet framework.
    - Other? The seed must enable installation and configuration of the Darwinet node framework on the platform at hand. Examples of platforms are Windows PC, Macintosh PC, Mobile phone, other portable device.

- The user uses the seed to install the Darwinet node framework on the platform where the node is to execute.
- The user starts the Darwinet framework administrator interface and creates the user domain.
- The user domain defines the following:
    - The user domain name
    - The users that are members of the user domain
    - The physical nodes of the domain. This is the locations where there will be user nodes executing that are allowed to participate in this domain.
    - Darwinet Applications which are available within the domain.
    - The folder structure and file contents of the shared files.

Note that all aspects except the domain name may be altered during the life-time of the domain. New users may be added. New physical execution locations of nodes may be added. New applications may be added. The folder structure of the shared files may be altered.

And all alterations are recorded in a time-line allowing each user to access change history of any aspect of the domain.

Adding applications provided by the Darwinet service provider

The Darwinet service provider provides Darwinet applications to be installed and used in Darwinet user domains.

1. Darwinet applications are developed by third party developers. Anyone is allowed to develop and distribute Darwinet applications.
2. Darwinet applications are provided by the Darwinet service provider. Anyone is allowed to set up a Darwinet service provider.
3. A Darwinet application developer cuts a deal with one or many Darwinet service providers to provide their application. The deal defines the price (if any) the user must pay to use the application in their user domain.

When the Darwinet service provider is set up with the new Darwinet application it is instantly available to be installable into Darwinet user domains.

- A user of a Darwinet user domain may open the “Darwinet application store” and select to buy an application.
- The user pays for the application and the application is installed for use by the user in the user domain.
- Applications bought by a single user are only available to that user. But it is available on all physical nodes of the user domain.
- A user domain administrator may select to buy and application to be used by all users of the domain. In this way the administrator may populate a user domain with the applications of choice.

Example of a Darwinet user domain usage scenario

A company starts a project to develop a new product.

The project needs to share a project plan, project files and the ability to report time they spend in the project.

The project leader decides to create a protected Darwinet user domain for the project.

- The project leader visits the web-site of a Darwinet domain service provider and creates a user account using an e-mail address.
- The service provider provides the domain seed to the mail-address selected.
- The project leader confirms the e-mail address and installs the Darwinet node framework using the seed.
- When the node is installed the project leader opens the Darwinet domain administrator interface and creates the Darwinet user domain giving it a unique name.
- The project leader then opens the administrator interface of the created user domain and performs the following:
    - Buys a project administration application and adds it to be used by all users of the domain.
    - Buys a time report application and adds it to the domain to be used by all users of the domain.
    - Creates a project folder in the shared files folder for the domain.

The project leader now starts to use the applications of the domain to set up the project.

The project leader then clones the node for all the users of the project.

The node cloning is made as follows:

- The project leader opens the user domain interface and selects to invite new users to the domain.
- Each user is defined using an e-mail address of the user.
- The Darwinet framework creates a node seed for each user and mails it to the user.
- Each user installs the user domain node using the seed received on mail.

Installing a Darwinet user domain node using a seed received on mail

The Darwinet domain service provider holds the user accounts of all users of the serviced Darwinet domain.

When a new user is invited to a user domain the Darwinet service provider provides the Darwinet node framework to the new user.

This scenario is the following:

- The user receives a user domain seed on mail.
- The seed is a special link to the Darwinet service provider to manage the user invitation
- The user clicks the link and is brought to a web-page.
- The user has to log in to the user account, creating the account if the user is not yet a member.

- If the user creates a new account a new mail will be created where the user has to verify the mail-address and activate the account.
- When the user logs in it is able to install the Darwinet framework and the user domain configuration.
- The user performs the installation and the result is a Darwinet domain node configured to act as the user domain node in the user domain to which the user was invited.

Also see other scenario where an existing member of a user domain clones the existing node to be used on another location.

Domain growth through new node scenario

The Domain should grow with new nodes in a way controlled by mechanisms within the Domain.

Member of the Domain with the right privileges are allowed to clone an existing node for a new or existing user. And the new node must be given a unique ID when installed to distinguish it from other nodes.

This would impose the following scenario when a new node is cloned, installed and authorized.

4. Node N1 is used by User A to create a Domain node clone for User B. We call this node for Node N*.
5. User B installs node N* in a lap top.
6. The User B starts node N* and logs in.
7. The Node identifies it has not yet been authorized and asks for access to node N1 through the nodes it has record of in the installation.
8. N1 receives the authorization request and produces a task to user A to authorize the new node.
9. User A sees the task and attends to it. User A is then presented with a description of node N* including a description of the location, the machine etc. that identifies where the node is installed.
10. User A fins all satisfactory and grants N* to be member of the Domain. N* gets the Id N73.
11. Node N73 to allows User B to launch any of the applications of the Domain.

Triple A Scenario

Consider to implement the Triple A scenario to all changes within the Darwinet.

- Authenticate the change – Meaning all changes have a producer that may provide an authentication of the produced change.
An example of this may be a weather forecast producer that authenticates all produced data to be correct and produced by them.
- Authorize the change – Meaning that no change is accepted unless the producer are Authorized to publish the change.
- Account the change – Meaning the change actually takes effect in other nodes. It also means that information may be charged for and debited the consumer of the information.

The Darwinet framework

Darwinet data management

The Darwinet data management is the technology to store and distribute data within a Darwinet network.

Data is stored in a Model, Instance, and Value layer model built from deltas (changes) within each layer.

The MIV (Model, Instance, Value) data structure

All data within a Darwinet domain are defined in a MIV (Model, Instance, and Value layer model).

The Model is the highest structure and defines available data structures or "types" within the domain.

The Instance structure defines current instances of types in the Model.

And finally the Value structure defines the current value of model instances.

Model, Instance, Value deltas (changes)

All data of a Darwinet domain are built from recorded changes within the Model, Instance, Value layers. A change is called a "Delta".

Thus a change to the Model is recorded as a Model delta (ΔM), an Instance Delta (ΔI) or a Value Delta (ΔV).

The current state of the Model, Instance, and Value layer stack is then defined by the sum of all Deltas up to the current time.

Data exchange

Data between nodes of a Darwinet domain is exchanged in the form of recorded changes. Thus the data that flows between the nodes are streams of deltas.

The Darwinet API stack

The Darwinet API stack is the architecture of Darwinet services of a Darwinet node that provides access to Darwinet controlled data and services.

As the name implies these services are organized into layers, one layer working on the data and services of the layer below.

Darwinet data distribution policy layer

This is the Darwinet stack layer responsible for implementing the data distribution policy.

Data is distributed based on:

1. Locally assigned storage size. Nodes configured to not store the whole body of data are assigned to store a selected fraction and fetch needed data from other nodes on demand.
2. Redundancy requirements. Data may be marked to be stored with a degree of redundancy like “At least two nodes” or “maximum number of nodes”.
3. Access time requirement. Data may be required to be stored on nodes accessible through high data rates or with high online time.
4. Etc...

Darwinet data protection layer

The Darwinet data protection layer is responsible for protecting the data from unauthorized access.

Data access may be granted based on a combination of:

- Current node
- Current application
- Current user
- Current time
- Current location
- Current communication line properties
- Current environment OS
- Etc...

Darwinet Authentication service

The Darwinet Authentication service is a layer of the API stack that checks the authentication of Darwinet network role players.

- Node authentication
- Application authentication
- Domain authentication
- User authentication
- Etc…

Darwinet encryption service

The Darwinet encryption service is the API stack layer handling encryption of data in a Darwinet domain.

Data may be encrypted for access only by:

- A domain

- A Node
- A user
- End-to-end between two nodes
- An environment OS
- An Application
- Darwinet networks
- Etc…

Darwinet Encryption Key infrastructure

Data within Darwinet networks manage a number of different keys for data protection using encryption.

- Each node creates a key pair by using properties of the local hardware. This key is never exposed nor stored. And it is unique to a physical or virtual hardware. The public part of this key is exposed within the domain and used by other nodes when sending data to the node.
- Each domain uses a key pair for all data within the domain. This key is created at the node where the domain was created and is maintained and updated of the authorized nodes of the domain. The public part of this key is used by other domains when sending data to the domain.
- Each user is assigned a key within a domain used to protect data of this user. User data at a node of the domain is not accessible by other users unless they are authorized. This key is stored at each node where the user has logged in and is protected by the domain key and the node key.

The Darwinet API based on XML over TCP/IP

Draft scenarios

Building a Darwinet Data View

A Darwinet application will access data of a domain through a Data view. This data view defines the data model, the data instances and the data values at a specific time of the history (including current time).

The data model, the data instances and the data values are results of a series of changes that lead to current state (state at the time defined by the view).

These changes are exchanged between nodes of the Darwinet domain nodes using XML.

Here follows a draft of how a view may be built using changes.

- The builder starts of by creating an empty data model
- It then retrieves the sequence of changes (model, instance and value) and starts of processing them.
- When all change has been processed the data body of the view is built.

A model change may be the following in XML.

```
<model_change operation=new>
      <scope>time_account<\scope>
      <id>time<\id>
      <type>integer<\type>
<\model_change>
```

The result of this change is that the existing type time_account now aggregates the field time of type integer. This corresponds to the c++ declaration:

```
struct time_account {
      int time;
}
```

A data instance change may be the following in XML

```
<instance_change operation=new>
      <scope>root<\scope>
      <type>time_account<\type>
      <id>TimeAccount<\id>
<\instance_change>
```

The result of this change is that there is now an instance called TimeAccount in the root scope which is of the type “time_account”.

A data value change may be the following in XML.

```
<value_change operation=add>
      <instance>root.TimeAccount.time<\instance>
      <value>7,5<\value>
<\value_change>
```

The result of this change is that the value of the field “time” of the composite instance TimeAccount is incremented by 7,5. In C++ this would correspond to the code:

```
…
      TimeAccount.time += 7,5;
…
```

NOTE: All changes must be reported using operations that are reversible based on the contained change information. This means that the operations new, delete, add, subtract etc. are allowed change operations. But the “set” or “assign” operation are not (any previous state will be lost in the operation so it is not reversible)!

TBD.

Overview of Darwinet API based on XML over TCP/IP

XML data compression

XML documents are sent between Darwinet nodes using the ISO/IEC 24824-1 (Fast Infoset) compression algorithm.

Model change record

```
<darwinet_message>
      <domain> itfied.acquiron </domain>
      <user> itfied.acquiron.users.kjell-olov </user>
      <from_peer> 1 <from_peer>
      <to_peer> 2 <to_peer>
      <evolution>
            … evolutions …
      </evolutions>
</darwinet_message>
```

```
<model_evolution>
      <add>
            <target> root.user </target>
            <type> darwinet.types.string </type>
            <caption> name </caption>
      </add>
</model_evolution>
```

```
<instance_evolution>
      <create>
            <type> root.user </type>
            <instance> user[23] </instance>
      </create>
</instance_evolution>
```

```
<value_evolution>
      <replace>
            <instance> user[23].name </instance>
            <value> kjell-olov </value>
      </replace>
</value_evolution>
```

The Darwinet API based on C++ class interface

Overview of Darwinet API based on C++ class interface.

TBD.

Darwinet Open P2P Service Cloud Specification The Darwinet API based on C++ class interface • 81

Project 1 Design specification

Project summary

Project Goal

Have two Darwinet nodes share a MIV.

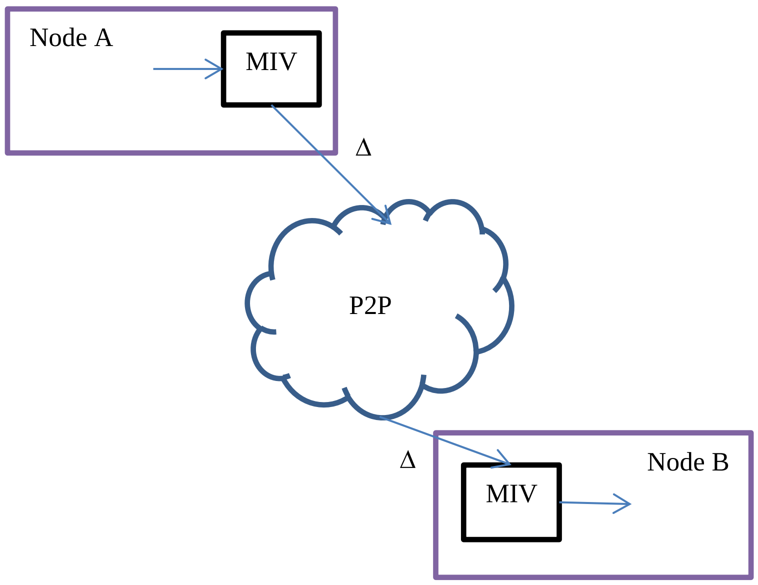
Motivation: A P2P Network needs to share common data and state without the uses of a central server. Sharing a MIV between nodes P2P is the core functionality of Darwinet that enables this.

Runtime Environment

Runtime environment

Two separate nodes running separately and interacting only using Darwinet P2P messaging.

The Nodes shall share a MIV with at least one value instance. Change of a value in the MIV of one node must be distributed to the other Node. The node must be noted when the MIV state changes.



Characteristics:

- The node shall be an executable application.

- The Application shall have a user interface to change a MIV value and show when a MIV value is remotely changed.
- The application may be a console application or a GUI application.

Excluded functionality:

- No MIV persistent storage (same lifetime as executable)

Functional description

Initial MIV state

The MIV is created with one integer having a fix default value.

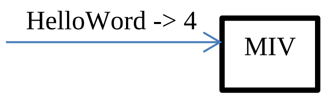
- Integer HelloWorld.
- Default value 2.

Change Exchange message flow

Use the user interface to set the HelloWord MIV variable to 4.

- A Console application could provide command option >SET HelloWord=4

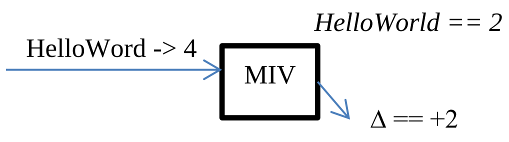
The Application calls the MIV API to set the value



```
// C
If (SetMIVValue(pMIV,” HelloWord”,”4”) == false) {
    ReportDesignInsufficiency(GetLastError(pMIV));
}

// C++
Try {
    pMIV->setValue(“HelloWorld”,”4”);
}
Catch (std::runtime_error& e) {
    LOG_DESIGN_INSUFFICIENCY(e.msg())
}
```

The MIV shall compare the set value with the existing value. If the value differs it shall produce a delta.



```
// C
Bool SetMIVValue(t_MIV* pMIV,char* szInstanceId,, char* szNewValue)
{
    …
    T_SequenceNumber sequence_number;
    GetCurrentSequenceNumber(pMIV, sequence_number);
    Char* szCurrentValue = GetMIVValue(pMIV,szInstanceId);
    If (strcmp(szCurrentValue, szNewValue) != 0) {
        szDeltaValue = // New Value - Current Value
        Result = ReportMIVValueChange(
             pDomain
            ,&sequence_number
            ,szInstanceId
            ,szDeltaValue);
    }
    …
}
```

Note: remote changes may be synchronized.

The Domain now distributes the Delta to the other nodes of the Domain.

Assemble a list of nodes to send the Delta to (*in small domains this is the complete list. In large domains it is a subset*). Use a “hard coded” list of nodes.

Note: The Final implementation shall store the Node List in a MIV of the Domain. In this way update of the Domain state (member nodes, users etc.) uses the same state share mechanism as any other Darwinet data.

Delta Processing Sequence Diagram

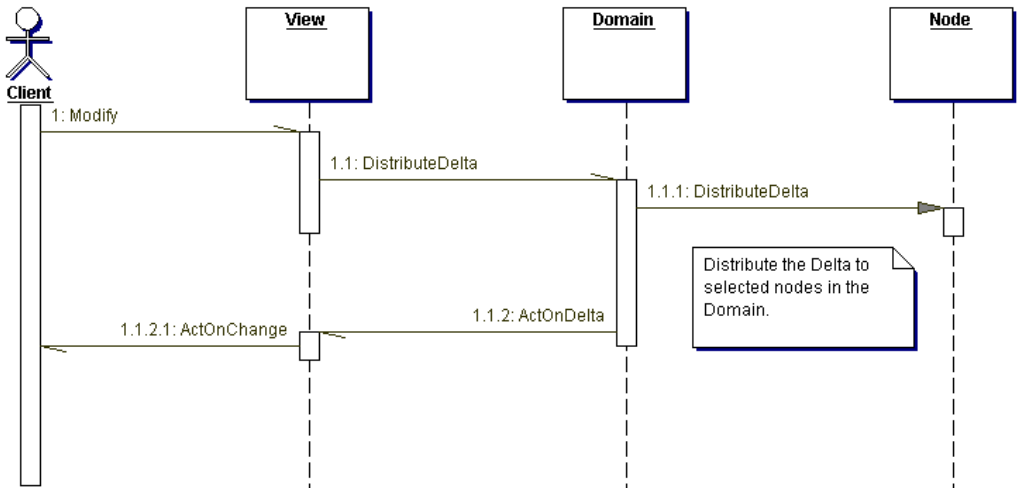
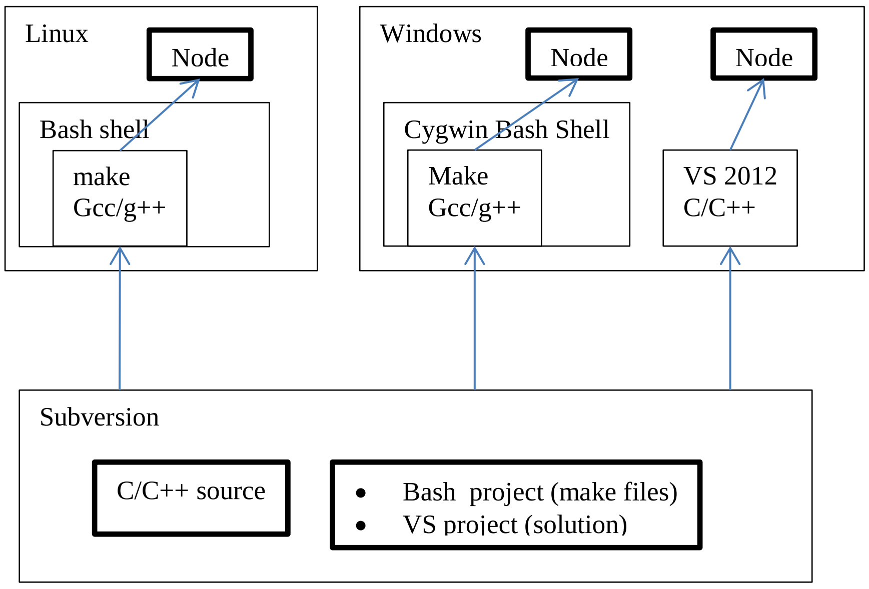


Figure 1 – Delta Processing Sequence Diagram

A Client accesses data in a Darwinet Domain though a View. The View provides the client with an interface to modify the state of the viewewed MIV. The View produces Deltas that reflects the actuakl changes made to the MIV state. The View distributes the produced Deltas to the Domain and applies them to itself. The View reports each MIV state back to the Client. In this way a locally and a Remotly produced Delta is applied by the View using the same mechanism.

Development Environment

UNIX and Windows target



Linux development environment

- Bash shell (or compatible)
- make
- gcc/g++

Windows Cygwin environment

- Cygwin bash shell
- make
- gcc/g++

Windows Desktop environment

- Visual Studio 2012

Darwinet Project Plan

Mile stones

1. A value sharing Darwinet node for windows. Simplest possible hard wired implementation.
2. A file sharing Darwinet node for windows. Simplest possible hard wired implementation. Just enough to distribute the Darwinet exe-file.
3. An “Expand network” enabled Darwinet node for windows. Simplest possible hard wired implementation. It must only be able to allow new nodes to be added to the network from existing nodes.

At this stage we have a very simple Darwinet node for Windows that are able to distribute updates to its own exe-file and to expand the network with new nodes.

4. A Darwinet node and a Darwinet Application for windows. Simplest possible implementation. The API between the Application and the Darwinet node must be extendable to iOS, Android, Mac and UNIX.
5. Add of Domain handling. Two domains. No user handling yet.
6. Add of User Handling. Two users.
7. Add of Domain data protection through encryption. No non-domain node should be able to read data in the domain.
8. Add of “New node Authorization” mechanism. A new node must retrieve authorization from existing node to be able to interact in the network.

Suggested Sprints

1. Share one value in a MIV between two nodes. Value change is exchanged.
   Simplest communication and implementation possible used. This implementation only serves as a core to iterate from.
   a. No domain. No user functionality.
   b. No p2p addressing or messaging. Nodes are hard wired to each other.
   c. No specific messaging protocol. Send value as simple as possible. Receiver hard wired to know the value.
   d. Value is transmitted as a value change! Receiver update value according to change!
2. TBD.

Sprint One

Overview

This sprint is finished when the test cases defined for sprint one passes by the system built based on the design goals.

Test case

Sprint Two

Overview

This sprint is finished when the test cases defined for sprint one passes by the system built based on the design goals.

Development goals

The design must support the following mechanism.

- A user logs in at a node.
- The user changes model definition, variable instance or variable value.
- The change propagates to the other nodes of the domain.

Specification update goals

All Darwinet specification documents must be updated according to findings in this sprint.

Design suggestions

The following design is suggested at sprint start. Final design is documented during sprint and at sprint end.

- At start of an application in a domain, the application asks the Darwinet node to build a) The Model Declaration, b) The Variable definitions and c) the variable values.
- The Darwinet node builds declarations, definitions and values from files of changes.
- All changes contain a change identifier consisting of the following information.
    - The domain identifier in which the change was made
    - The Application identifier that made the change
    - The date and time of the change
    - The user who made the change
    - The node at which the change was made
    - The user change entry sequence number

- The user change entry sequence number is a unique unbroken number 1...n of one user for a Domain and application. The sequence number enables nodes to detect if all changes of a user within a domain and application are received.
- The Darwinet node builds the model declaration from the file of model declaration changes.
  - A declaration defines a type.
  - A declaration change operation is Add or Remove.
  - The Add declaration syntax is:
    Add <added declaration identifier> <type identifier>
  - The Remove declaration syntax is:
    Remove <removed declaration identifier>
  - The declaration identifier is:
    declarations.<identifier>.<identifier>…<identifier>
  - Example: Add scc.chronos.declarations.time_account int
- The Darwinet node builds the variable definitions from the file of definition changes.
  - A definition defines a variable.
  - A definition identifier is
    definitions.<identifier>.<identifier>…<identifier>
  - A definition change operation is Add or Remove.
  - The Add definition syntax is:
    Add <definition identifier> <type>
  - The Remove definition syntax is:
    Remove <definition identifier>
  - Example: Add scc.chronos.definitions.0001 scc.chronos.declarations.time_account
- The Darwinet node builds the variable values from the file of variable value changes.
  - Variable value change defines the contents value of a variable
  - A variable value change operation is Add or Remove.
  - Example:
    Add scc.chronos.variables.0001 4

Test case 1

- Log in as user one (U1) on node one (N1) and report one time entry on time account 0001.
- Log in as user two (U2) on node two (N2) and report one time entry on time account 0001.
- Now log in as U2 on node N1 and check that U2 sees the time reported at N2.
- Now log in as U1 on node N2 and check that U1 sees the time reported at N1.

Document Revision History

Requirement change history

The following changes have been made that affect the requirements defined in this document.

Newest changes are at the top of the list.

VERSION	DATE	DESCRIPTION	ISSUED BY
1.0 Revision 10	131230	• Added “Delta Processing Sequence Diagram” with an UML diagram.	
1.0 Revision 10	131122	Updated “Delta Synchronization” chapter to describe the latest thoughts on how Darwinet Nodes shall exchange and synchronize deltas generated within the Domain (algorithm, still not complete)	
1.0 Revision 9	130217	Added Licensing information. • Licensed this document under Creative Commons Share Alike licence in chapter “Licensed for use and contribution”. • Discussed Open Source licensed in “Darwinet Licensing”.	Kjell-Olov Högdahl
1.0 Revision 8	130206	Created this Revision History.	Kjell-Olov Högdahl

Editorial change history

The following editorial changes have been made to this document. Editorial changes do not affect the specified requirements.

Newest changes are at the top of the list.

VERSION	DATE	DESCRIPTION	ISSUED BY
3.1 Revision 8	130217	• Merged contents of “The evolution mechanism” with contents of “The MIV evolution history mechanism”.	
3.1 Revision 8	130211	• Added “Introduction to Darwinet P2P network versus other networks” with sub-chapters about a) Server centred Domain, b) Network file folder network and c) Darwinet Service Cloud network.	
3.1 Revision 8	130209	Illustrated User, Link and Service provider roles of Darwinet nodes in “Small set of base Darwinet services used to build all complex services”. Described node support for multiple messaging protocol stacks • “P2P using different messaging protocol stacks” • “P2P internode messaging protocol stack option” • “Node Protocol Stack model” • “Node Messaging Protocol stack support modelling” Totally revised “Darwinet usage scenarios” • All previous chapters deleted (deprecated)	

		• New chapters added. They are mostly empty but define a structure into which to add text. Removed “Darwinet data API usage scenarios” (Deprecated) Added “Seeding over Darwinet” • Described how a Darwinet Domain may “Seed” over Darwinet to nodes already a Darwinet member.	
3.1 Revision 8	130206	Created this Revision History.	Kjell-Olov Högdahl

Glossary of Terms

Darwinet

A Darwinet is a P2P network of Service providing nodes.

Darwinet Application

A Darwinet Application is an application that interacts with a Darwinet Domain data or file.

A Darwinet node that provides a virtual drive containing files of a Domain enables any standard application to act as a Darwinet Application.

An Application that interfaces with the services of the Darwinet node is a Darwinet enabled application.

MIV

A MIV is the Model, Instance, Value data instance of the current recorded state of a Domain View.

∆MIV

A Delta MIV is a change to a MIV. The change may be a data model (type) change (∆M), a value instance change (∆I) or a value change (∆V).

Index

No index entries found.