REPORT

Define a computation trust approach over the social graph

- project deliverable 4.5

Authors: Kevin Koidl (TCD) Kristina Kapanova (TCD), Barbara Guidi (UNIPI), Giulia Fois (UNIPI) Andrea Michienzi (UNIPI), Laura Ricci (UNIPI), Kuusijärvi Jarkko (VTT)

Confidentiality: *Public*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825585.

Heterogeneous Social Network Graph topology and lifecycle	
Project name HELIOS	**Grant agreement #** 825585
Author(s) Kevin Koidl (TCD), Kristina Kapanova (TCD), Barbara Guidi (UNIPI), Andrea Michienzi (UNIPI), Giulia Fois (UNIPI), Laura Ricci (UNIPI), Kuusijärvi Jarkko (VTT).	**Pages** 42
Reviewers Carlos Alberto Martin Edo (ATOS), Javier Marín Morales (UPV)	
Keywords Trust, Trust Models, Online Social Networks, Decentralized Online Social Networks	**Deliverable identification** D4.5

Summary

In this deliverable, we first show how the problem of trust is framed by state-of-the art research, by giving a brief survey of the studies carried out in this field; based on this survey, we then outline which are the main properties of trust in social environments. Thereafter, we introduce the HELIOS trust model we theorized, by defining and describing the features that are exploited, and we provide the actual formula that we use in the module for trust computation. Finally, we give an overview of the APIs that can be employed by the other HELIOS modules or the client application to instantiate the trust module, update it in case of new events happening in the Conceptual Ego Network (CEN), or retrieve the last trust value that has been computed for an alter in a specific context.

Confidentiality	Public

29.10.2020
Written by

Barbara Guidi (UNIPI, Task Leader),
Kristina Kapanova (TCD)

Project Coordinator's contact address
Ville Ollikainen, ville.ollikainen@vtt.fi, +358 400 841116

Distribution
HELIOS project partners, subcontractors, the Project Officer and HELIOS web site

Contents

List of Acronyms

Acronym	Description
API	Application Programming Interface
BOSN	Blockchain-based Online Social Network
CEN	Contextual Ego Network
FOAF	Friend Of A Friend
DHT	Distributed Hash Table
DOSN	Decentralized Online Social Network
DTN	Delay Tolerant Network
HSG	Heterogeneous Social Graph
NGI	Next Generation Internet
OSN	Online Social Network
P2P	Peer-to-Peer

1 Introduction

HELIOS is a decentralized social media platform that addresses the dynamic nature of human communications in three dimensions: contextual, spatial and temporal.

HELIOS will advance the current approaches to social media by introducing novel concepts for social graph creation and management by exploiting trust and transparency. Indeed, HELIOS introduces a novel way to create, maintain and configure personal social graphs by exploiting context social data that are available when the application is running. The platform is principally based on the principle of trustworthiness, and it provides all necessary means and features to ensure the highest level of trust and control.

Privacy and trust are today important requirements in Social Media. Social media greatly enables people to participate in online activities and shatters the barrier for online users to create and share information at any place at any time.

Social Media refers to platforms that are designed to allow people to share content and to communicate with people. There are many different types of Social Media, and Social Networks are one of the most well-known social media. Social Networks are mainly based on communication between users. They propose services to connect users to other users. The main difference between Social Media and Social Network is the goal they try to achieve. In Social Networks the goal is to build a network of relationships and share with them content. Instead in Social Media the goal is principally to share content with the others.

In the Social Network environment, trust is crucial in helping users collect both relevant information and relevant relationships.

As concerns content, fake news can be considered an issue in terms of content trust. The “fake news” phenomenon is critical to combating the ongoing global erosion of the trust in the Social Networks. Fake news has acquired a very special significance in the wake of the latest US elections, but also during the diffusion of the COVID-19.

In Online Social Networks, trust affects relationships and content. Indeed, meaningful relationships are characterized by a certain level of trust between the two parties. An individual that has a trustful relationship with another one will be likely to have a positive and frequent communication. Moreover, one will be probably willing to share some social information and content that is made unavailable to others (the relationship with whom lacks of trustfulness).

Trust has been studied in many research fields. Each of these research fields has defined and considered trust from different perspectives. For this reason there is no simple definition of trust which can be used, but the definition depends on the scenario. In general, trust is a measure of confidence that an entity or entities will behave in an expected manner.

Trust takes the form of a binary bond where the two parties are the trustor (that is the one that has a certain level of trust towards a target entity) and a trustee (that is the target entity that is trusted). In virtual environments, like the one of Online Social Networks, trust has the additional property of representing a meaningful element whenever it is necessary to decide with which entities information can be shared. Moreover, trust also plays an important discriminant when it comes to allow another entity to share information without any further verification.

Trust plays an important role in how users act on social networks. Mechanisms that enforce different levels of privacy have been defined for the most used OSNs, like Facebook or Twitter. A Social Network platform where trust is one of the primary principles is important today, and HELIOS faces this challenge by providing a new trust model.

This new trust model represents a novelty in the field of Social Networks trust research. The reason is that while many studies have been carried out with reference to trust in OSNs or trust in P2P networks, almost no studies have been dedicated to the matter of trust in DOSNs. In DOSNs it is not enough to simply adopt the same considerations that are made for OSNs, because of the fact that decentralization, that is the main feature, carries along a whole set of problems that have to be addressed. The main concerns are related to privacy aspects: while in OSNs trust is computed by a central authority, that disposes of all the information coming from the nodes in the network, in DOSNs peers most likely don't want to share their personal information with others. Moreover, computational power limitations arise as well, given the hardware where DOSNs clients are installed and run.

Trust computation has thus its own set of new concerns that had not emerged in the other mentioned matters regarding OSNs. Our model does tackle those problems, and moreover introduces the element of Sentiment Analysis, that characterizes relationships based on media content and messages exchanged by two parties. We propose a trust model that exploits Sentiment Analysis as a feature, which has not been done by any other models.

In this deliverable, we first show how the problem of trust is framed by state-of-the art research, by giving a brief survey of the studies carried out in this field; based on this survey, we then outline which are the main properties of trust in social environments. Thereafter, we introduce the HELIOS

trust model we theorized, by defining and describing the features that are exploited, and we provide the actual formula that we use in the module for trust computation. Finally, we give an overview of the APIs that can be employed by the other HELIOS modules or the client application to instantiate the trust module, update it in case of new events happening in the Conceptual Ego Network (CEN), or retrieve the last trust value that has been computed for an alter in a specific context.

2 State of the art

Trust is a multi-disciplinary, multi-faceted concept. Definitions of trust have been given in multiple fields, ranging from psychology to social sciences, from economics to law, from communication science to information technology. Within the computer science domain there is no combined and straightforward definition of trust. For instance, trust can be defined as the expectations built towards another entity based on previous interactions [1]. Other definitions are more related to the "service side" of computational relationships, and frame trust as the belief of an agent behaving in a certain way for a specific period of time with relation to the offer of a specific service [2].

Netrvalova et al., in [3], define trust as "*the confidence in the ability or intention of a person to be of benefit to something or someone at sometime in future*", based on the definition originally given by Gambetta [4].

In [5], trust is defined as "*a measure of confidence that an entity or entities will behave in an expected manner despite the lack of ability to monitor or control the environment on which they operate*", and the concept of trust communities is introduced as being "*communities that create an environment where its members can share their thoughts, opinions and experiences in an open and honest way without concerns about their privacy and fear of being judged*". From these former definitions it is already possible to notice that trust is strictly related to the confidence in another person or entity's future behaviour, and is therefore subjective; moreover, in a social context, the higher is the trust towards another agent, the more a user is willing to share personal content with that agent.

Liu et al., in [6], refer back to Goldbeck's original definition of trust in a person [7] as the "*commitment to an action based on a belief that the future action of that person will lead to a good outcome*". Here again the component of future behaviour is present; a person is trusted if its future behaviour will be beneficial to the trustor.

Adali et al. give a difference of trust more in relation to the role it plays in a social network [8]. The role of trust is in fact defined as important "*in the formation of coalitions in social networks, in assessing quality and credibility of information as well as in determining how information flows through the network*".

From these definitions the bond between the concept of trust and the social aspects of computing appear very clear, but trust isn't exclusively related to human ties: as stated by Hoffmann, trust of

human agents in technology and automation needs to be analyzed as well in this era of cybersecurity vulnerability [9].

All these definitions of trust in relation to a social network context are different but complementary. It is important to also frame the concept of trust with respect to its properties that can be derived by reasoning on the definitions themselves and by thinking about how real world human trust can be translated into virtual trust.

2.1 Trust Properties

Fuzziness
Trust is not a binary value: there is no such thing as “complete trust” or “complete distrust”. A user in a social network can trust another one to a certain extent, and the trust value lies into a continuous interval.

Dynamism
The trust value is subject to incessant change, depending on various parameters, for example the positive/negative outcome of interactions.

Non-transitivity
Let A, B and C be three users in the social network. The fact that A trusts B with trust value t and B trusts C with a similar trust value doesn’t necessarily imply that A will trust C with a trust value around t as well; it may be the case that A doesn’t actually trust C at all. This is true in general for non-social contexts (like file sharing or e-commerce), but it is even more emphasized in social networks, where relationships are unique and closely linked to personal matters.

Asymmetry
The trust in a relationship between two people is not necessarily bidirectional. Let A and B be two users in the social network: it may be the case that A has a high trust value towards B while B doesn’t trust A at all. As non-transitivity, this property too generally holds for networks having different purposes.

Subjectivity
Reconnecting with what has just been said, trust is highly related to the trustor and the unique

relationship between them and each trustee. Some people may be inclined to give more initial trust to strangers, and increase it significantly for each positive interaction; others, on the other hand, may be less prone to trust others, and this translates to a slower process of trust building.

Context dependency

Let A and B be two users in the social network, and let B be specialized in a specific job field. A may give B a high value of trust for matters that concern B's job, but A and B may not be close friends, and therefore A wouldn't want to share personal information with B. This example shows how, according to the different contexts relationships can be based on, trust between two users may assume multiple values, depending on the context we are referring to. In a social network built around B's job, for example, A may disclose their information related to that (for example, if B is a doctor, A would share its medical information), but they may not even be friends on a social network built on friendships (e.g. Facebook).

Trust plays an important role in how users act on social networks. Mechanisms that enforce different levels of privacy have been defined for the most used OSNs, like Facebook or Twitter. Different privacy settings are clearly connected to different levels of trust in users we want our private contents to be accessible to or, on the contrary, users we do not want to share anything with. The definition of trust models in centralized OSNs often involves the acquisition and exploitation of recommendations and reputation. In centralized contexts this is clearly possible, because of the fact that the computation itself is probably not done by the client, and this constitutes an advantage both in terms of storage and computational power and in terms of privacy matters (since the trust and reputation information is only shared with the central authority).

When dealing with DOSNs, defining a trust model gets more complicated. The reasons are related to the decentralized nature of the network itself that has no central authority and therefore makes privacy, information gathering and computational power usage significant problems. Users of DOSNs most likely do not want to share their private trust information to other peers, and this results in a challenge when it comes to assigning a certain reputation score. Moreover, due to the hardware where DOSNs are installed, users usually do not have unlimited computational resources; the social network client application needs to be lightweight, the same being true for the computations, as to not be a burden on users' devices. Detecting malicious nodes that are likely to deceive in the future is also an aspect to take into account. This aspect is also important in centralized OSNs, but is easier to tackle due to the presence of a central authority that is able to possibly take care of negative behaviour.

These challenges require trust models for DOSNs to consider different information and to be based on a different set of features with respect to standard OSNs. The sets of trust features for centralized OSNs and for DOSNs are not disjoint: some features of centralized OSNs' trust models are still valid and determine feasible computations even in decentralized environments. Some, instead, need to be discarded or heavily adapted to a decentralized context, in particular the ones that rely on information coming from all over the network.

In the literature, several trust models for distributed networks have been proposed, although currently there is a lack of models with specific focus on decentralized social networks. Works on trust models for decentralized networks can be divided into three categories, based on the specific purpose of the network they are geared towards:

- Trust models for file sharing (or generally **resource sharing**) networks. These models are particularly designed for common situations in which multiple nodes may be offering the same resource, hence the necessity of evaluating which node to choose to fulfill a particular request. Trust in this case allows the avoidance of malicious nodes that may cheat or supply a resource with a different quality from what is expected.
- Trust models for **general** P2P networks. This is the category that so far has the greatest number of models to the asset. Models of this kind are general purpose and can possibly be adapted to all contexts, but lack of specificity and thus some level of integration of specific features needs to be pursued.
- Trust models for **social** networks. This category contains models that address social issues more predominantly: who is worth sharing profile information with? Are there any unknown nodes that can be trusted or distrusted based on information received from the social overlay? In decentralized social networks node connections have to be considered at the application level as well: links in the social overlay, created by manually adding an edge to another user or by automatic processes (i.e. link prediction), may be considered as preferential channel to get trust information.

The following sections further analyze the trust models proposed in the literature for each important category.

2.2 Trust models for resource sharing networks

Decentralized resource sharing networks are characterized by an initial complete distrust between nodes. Nodes' neighbours are drawn in a way that is guided by the underlying DHT's policy, and in general the identities of peers are completely unknown. While certain security mechanisms have been provided to address some threats/vulnerabilities (such as the more robust identifier

generation of S/Kademlia, that helps preventing the Sybil attack[1]), it is still fundamental to have trust models that allow users in the network to avoid stumbling into harmful resources (such as viruses, Trojan Horses, or general malicious executable files) when trying to download some legitimate resource.

Damiani et al. [10] propose P2PRep, a trust model based on reputation and on the presence of opaque identifiers for nodes that is stable identifiers that do not compromise anonymity but allow the possibility of associating transactions to a specific resource servant. Whenever a peer p is looking for a resource that is available to download by a certain set S of servants, in order to choose the most trustworthy one p sends a poll request to its peers, to which they have to reply with a set of trust votes V of each servant in S. To be sure of the reliability of the votes, p also keeps track of the credibility of its peers, by making use of a reputation manager. After receiving all the votes, p computes a reputation score of each servant by putting together all the votes received about them. Votes of other peers are based on servants' behaviour as resource providers.

TACS [11] is a model proposed by Mármol et al. and it is based on the ant-colony optimization heuristic. This allows identifying the most covered paths in the network, and therefore assigning a routing preference to them with respect to less covered paths. For each service s offered by nodes in the network, the edges are weighted according to the pheromone (that represents the trust in reaching the optimum server travelling across that edge) and the service heuristic (that represents the similarity between the service offered by the tail of the edge and s). For a client c requesting for a specific service s, the algorithm selects the optimum path (and the target node t where s is available) according to the weights-state of the network, updates the pheromone state of the percolated edges, requests the service to t and then evaluates the service that has been provided (if c is not satisfied, the pheromones on the path leading to t are evaporated according to a certain policy). Experiments show that, with specific pheromone updating rules and starting parameters, the chosen benevolent servants percentage is higher with respect to what happens with other algorithms.

These works strongly highlight the correlation between trust and reputation, based on peers ratings or on most crossed paths in the network). Stakhanova et al., describe in [12], a model that combines reputation with anomaly detection since considering solely reputation, in fact, may lead to not detecting situations of abnormal or suspicious behaviour of peers. Reputation of a peer p is computed by taking into consideration the percentage of "bad actions" carried out by p, and

[1] In computer security, the Sybil attack is an attack that affects reputation systems by creating several identities, that are pseudonyms for the attacker. It is aimed at gaining a certain influence over the whole network, by acquiring a disproportionate level of control on it.

anomalies of p's behaviour result in an increase of this percentage. Anomaly detection employs an unsupervised learning algorithm trained on the normal behaviour of the peer (in terms of network and temporal activity), that is in turn evaluated by profiling the peer based on its sessions' data.

Tang et al. [13] propose a model based on fuzzy theory. The thesis of the authors is in fact that fuzziness is a feature we should consider when reasoning about trust, because of the fact that it's really tied to the observer's criteria and therefore highly subjective. In particular, trust can be modeled as an aggregation of different parameters having different weights depending on the observer. A recommendation from another peer should be considered according to the similarity between the trust evaluation criteria of the recommender and the ones of the peer who receives the recommendation. Acceptance of a recommendation should therefore be modeled with fuzzy theory as well. The process of trust deduction is based on these considerations. Moreover, a component of trust revision should also be present, because of the dynamicity of trust itself. In this model, trust revision is based on fuzzy logic rules that take into account past and current experience.

From the features that emerged from the analysis of this category's models, reputation is the one that for sure cannot be applied to our context, because of privacy issues, as we discussed. One feature is instead more related to the necessities of peer to peer file sharing: stable identifiers are in fact needed in a non-social context where the peers one is connected to change all the time, but in a social context this feature is trivially present and therefore not worth mentioning. For what concerns anomalous behaviour detection, this feature needs to be re-mapped to a social context, where it may be worth detecting social anomalous behaviour (e.g. the behaviour of bots, or of spammers) rather than network anomalous behaviour itself.

2.3 Trust models for general P2P networks

The following trust models have been defined for P2P networks with no specific purpose (unlike the ones of the first category, that are related to file sharing networks). For this reason, some may present features that are suitable for DOSNs, while others may not.

In [14], Wang and Vassileva propose a trust model based on Bayesian networks. The model is tested on a file sharing application but, according to the authors, it can be generalized to other kinds of networks as well. The model is based on the fact that the trust in a peer can depend on a lot of different parameters, and each one has a different weight according to the trustor's needs in a particular moment. For each other peer that peer p interacts with, a naive Bayesian Network is built: the root is the trust T from p to the other peer, and the children nodes are all the parameters that, for each interaction, need to be evaluated in order to determine the general interaction quality and update the trust value accordingly.

Aberer and Despotovic [15] illustrate a trust model based on reputation drawn on behavioural data, that is the set of assessments made by peers about the interactions with a specific peer p. In order to assess the trust value to be given to p, a peer needs to know its own behavioural data about p (i.e. its evaluations of interactions with p) and possibly a subset of the global behavioural data about p, that is a set of evaluations of interactions with p shared by other peers, called “witnesses”. This constitutes the reputation component. More specifically, trust is modeled as a binary value, and the behavioural data set only contains the complaints, that are negative evaluations of transactions with a peer. The trust value to give to p is assessed by applying an heuristic on the number of complaints about p (considering also witnesses) and the number of complaints filed by p: if those two quantities exceed the average values of more than a certain factor (in turn assessed with a probabilistic analysis), the peer is to be considered untrustworthy. This trust model heavily relies on the availability of transactions evaluation data of other peers.

Another model that relies on both “first-hand experience” and on “second-hand experience” (i.e. reputation evaluation) is the one proposed in [16], where these two types of information are updated and combined by exploiting a Bayesian framework.

VectorTrust [17] is a model proposed by Zhao and Li and it is based on a trust aggregation algorithm built over a trust overlay. The latter contains directed edges, called trust vectors, labeled with the trust score the source node has assigned (or updated) to the destination node following an interaction between the two. When a node n wants to infer the trust value towards another one it has never interacted with, n percolates all the possible trust paths (made of trust vectors) directed to the other node, and then chooses the most trustable path, that is the one that corresponds to the maximal product value of all trust vectors along each possible trust path. Trust is then evaluated as the value of the product that corresponds to the most trustable path.

In general, we can say that all the trust models defined for general P2P networks in the literature rely heavily either on reputation or in any case on interactions evaluation data obtained by other nodes. This is a feature that is not suitable for our model, as discussed already multiple times. An interesting feature that did not emerge before is the weighting of parameters: trust, in fact, depends on multiple factors and it should be up to the specific peer to decide which ones should have bigger importance and which ones, instead, do not matter much to them. Moreover, we find again the interactions evaluation feature that rightfully seems to be a constant in all kinds of models: considering the history between two users is crucial when computing the trust between them.

2.4 Trust models Decentralized Online Social Networks

There are not many works in the literature that deal with the concept of trust in DOSNs and which define models to compute it. Modelling trust in a social decentralized context is for sure harder compared to the cases we have treated before, because one has to deal with privacy matters. Moreover, it is difficult to define what social trust is, which are the parameters that have to be taken into account to compute it, and what can be classified as “malicious behaviour”: while in the file sharing case it was sufficient to analyze the other nodes’ network activity and the quality of resources offered, the social environment is way more polyhedric, because it involves real human behaviour and feelings.

Juan Li and Qingrui Li [18] design a model specifically targeted to mobile ad hoc social networks. This model deals with a problem we dealt with before, which is the possible data unavailability. In fact, they take into consideration two possible cases: first, where a user can collect sufficient “hints” from the network, i.e. if there are users willing to share trust information with them, and a second one where a user, instead, cannot dispose of any other users’ private information at all. In the first situation, to compute trust towards an unknown user, a top-k ranking (based on profile similarity) of users available to share trust information is drafted. Trust towards an unknown user can be then computed as a weighted average (where the weights are the profile similarities) of the trust values $t(u_i,v)$, where v is the unknown user and u_i is each of the users in the top-k ranking. In the second situation, that is more comparable with ours, the user does not collect any more information than its own, and thus has to rely only on its local data to compute the trust value towards another unknown user. The three parameters that are taken into account are profile similarity, local reputation (that can be in turn computed by taking into account the history of interactions), and the number of common friends.

$$Trust(A,B) = \alpha \times Sim\big(Prof(B), Prof(A)\big) + \beta \times Rep(A,B) + \gamma \times fof(A,B)$$

Equation 1. Trust Model described in [19]

The formula shows in Equation 1 comprises the model we have just described. The first parameter is the profile similarity, weighted by α; the second one is the local reputation, weighted by β; finally, the third parameter is the number of common friends, weighted by γ. These three weight factors can be tuned according to the importance of each parameter for the effective computation.

Qureshi et al. [19] propose a trust model used to form trust communities in peer to peer mobile social networks. Trust values are expressed as discrete values in the interval [0,3]; moreover, each peer has an opinion towards other peers, that is based on the evaluation of previous interactions and is expressed as a discrete value in the interval [-1,1]. The computation of trust is reputation-based: whenever a node encounters an unknown other node x, it asks other known peers for their trust values and opinions towards x, and uses those values to compute its own trust value. This work does not take into account the privacy matter, and while presenting itself as a model tailored for social networks, its features do not differ much from the ones defined for other kinds of P2P networks.

2.5 Trust models for centralized Online Social Networks

As previously stated, there are not many works in the literature about trust that are centered specifically on decentralized online social networks; the models proposed for other kinds of networks present features that cannot be really adapted for our purposes. For this reason, we now review some trust models proposed for centralized online social networks, in order to understand which are the features that emerge from a social context and discuss if we can adapt them to make them suitable for a decentralized environment as well.

In [20], Netrvalova and Safarik introduce a centralized trust model based on the reputation of the trustee, trusting disposition of the trustor and recommendations. A variation of the trust value is thus determined by these parameters and by the mutual current trust values. This model does not really consider features that are peculiar of social networks.

STrust is a model proposed by Nepal et al. in [21] aimed at computing trust in order to build trust communities inside social networks. According to the authors of this work, the most important properties of trust are time and context dependency, while the main parameter for its computation is user behaviour. User behaviour is expressed in terms of social capital that is constituted by interactions between nodes of the network. Based on the type of interactions, two different kinds of trust are defined. The first one is popularity trust, that refers to the trustworthiness of a node from the perspective of other members in the community and is therefore affected by metrics like positive/negative feedback on posts or comments, number of friendship requests and so on. The second type of trust is engagement trust, that is tied to the involvement of the node in the community, and is thus related to the feedback provided to other posts, the friendship requests sent, or other more passive activities, like the amount of time spent reading the posts or observing the discussions. These two kinds of trust are computed by keeping into account positive and negative interactions in each context, and then by aggregating over the contexts; the final value of trust is computed as a weighted sum of these two quantities.

In [22], Yu and Singh propose a trust model that is based on trust ratings. A trust rating depends on direct behavioral observations and reputation, that aggregates trust ratings of other users, weighted depending on the trustworthiness of the users themselves. It is defined in the interval [-1,1]. For each interaction between two nodes, the trustor updates the trust value towards the other node by selecting a specific rule depending on the rating of the interaction itself, and depending on the trust testimonies of neighbours. This model is not interesting for our purposes, because it relies significantly on reputation.

In [23], Liu et al. propose a probabilistic trust inference model based on two factors: the intimacy degree between users, measured as a real value in the interval [0,1], and the role impact, also mapped as a value in [0,1] and representative of the expertise of a user in a specific domain, and therefore the goodness of the user's recommendations in that domain. Trust is then computed with a posterior probability estimation based on the Bayes theorem. This model is built on the assumption of the presence of trust transitivity up to a certain extent, and is therefore incompatible with our research, because according to the properties we defined above trust is definitely non-transitive. Moreover, this model makes use of an evaluation of users' recommendations that can be seen as a form of reputation: this too is incompatible with the definition of our model.

In [24], Goldbeck and Kuter propose SUNNY, which is also a probabilistic trust model based on bayesian networks. The aim of this work is to find a way to infer trust between two nodes (the source and the sink) of a trust network. Trust is estimated by taking into account nodes as information sources with a high level of confidence, defined as the belief of other nodes on the correctness of information provided. No other social information is taken into account for the computation of trust. Moreover, this model would be computationally overly complicated if it were to be implemented in a decentralized context, because it would need a view of the social overlay that goes beyond the local view of our nodes in HELIOS.

Maheswaran et al. [25] define a trust model that takes into account the possibility of having different trust relationships between two nodes depending on the contexts they are in. Trust is therefore not modeled as a real value but as a m-dimensional vector of real numbers, where m is the number of contexts the two users belong to. This vector represents the trust distance between the two nodes in the m-dimensional trust space, and is updated according to the occurrence of positive/negative interactions among them. The final summarizing trust value of node i towards node j is computed by taking into account this trust distance vector and the age of node j as evaluated by node i.

2.6 Summary

In this section, we summarize the presented State of the art in order to highlight the main features of each work. The following tables (Table 1, Table 2, Table 3, Table 4) summarize the works we cited, following the partitioning we considered before.

Trust models for centralized OSNs	
Work	**Features**
Netrvalova and Safarik [20]	Reputation, recommendations, trusting disposition
STrust - Nepal et al. [21]	Popularity (positive/negative feedbacks received, friendship requests received…), engagement (feedbacks provided, friendship requests sent, post reading activity…), context
Yu and Singh [22]	Behavioral observations (interactions evaluation), reputation
Liu et al. [6]	Intimacy degree, expertise in a domain
SUNNY - Goldbeck and Kuter [24]	Confidence
Maheswaran et al. [25]	Context, interactions evaluation

Table 1. A brief overview of the mentioned works carried out on trust related to centralized OSNs. For each work, we highligh the main features of the defined trust model.

Trust models for P2P resource sharing networks	
Work	**Features**
P2PRep - Damiani et al. [10]	Reputation, behaviour as resource provider
TACS - Mármol et al. [11]	Path-based reputation, behaviour as resource provider
Stakhanova et al. [12]	Reputation, anomalous behaviour detection
Tang et al. [13]	Subjective parameter weights, recommendations, experience evaluation

Table 2. A brief overview of the mentioned works carried out on trust related to P2P resources sharing networks. For each work, we highlight the main features of the defined trust model.

Trust models for general P2P networks	
Work	**Features**
Wang and Vassileva [14]	Subjective parameter weights
Aberer and Despotovic [15]	Reputation, behaviour evaluation
Buchegger and Boudec [16]	Reputation, behaviour evaluation
VectorTrust - Zhao and Li [17]	Trust aggregation (over graph paths)

Table 3. This table contains a brief overview of the mentioned works carried out on trust related to general P2P networks. For each of the works, we also highlighted the main features of the trust models defined in them.

Trust models for DOSNs	
Work	**Features**
Li and Li [18]	Profile similarity, interactions evaluation, common friends
Qureshi et al. [19]	Interactions evaluation, reputation

Table 4. This table contains a brief overview of the mentioned works carried out on trust related to DOSNs. For each of the works, we also highlighted the main features of the trust models defined in them.

As one can observe, trust computation is characterized, in different settings, by some common features, in particular reputation and interactions evaluation, whether resource sharing interactions or social interactions like comments or private messages. The former cannot be considered nor adapted for our purposes: the computation of reputation score relies on trust ratings coming from other nodes, and this is impractical for two reasons. The first reason is privacy: trust information is private and other nodes in the network must not be able to have access to it. The second reason is that requiring trust information from numerous nodes in the network would constitute a big overhead at network communication level. Moreover, nodes in HELIOS only have a partial view of the entire network, and the nodes in the social overlay may not be the ones who have the information requested. So while this former feature involves a whole set of issues that we can not overlook, the latter is for sure an essential cornerstone to take into consideration for our model as well. It is in fact impossible to talk about trust without reasoning about how it is affected by interpersonal relationships that in a social network environment are translated into interactions between users.

Another feature that emerged both in P2P trust models and in centralized social trust models is the subjectivity of trust, declined in various ways, e.g. trusting disposition or differently weighted trust parameters. As trust reflects reality to a certain extent, it is reasonable to think that some users, under the same conditions, may be prone to assign more trust than others; moreover, a feature that is vital for a user may be less important for another one.

Contextual dependency is an interesting feature that only emerged by two works, but that is nonetheless worth considering, in particular because of the multi-layered nature of our social network, where layers correspond precisely to human contexts.

3 HELIOS Trust Module

In this Section, we provide a description of the Trust module. The Trust Module is one of the core modules of the HELIOS architecture, as visible in Figure 1. The modules highlighted in red are the ones the Trust Module depends on, that are the Neuro-Behavioural Classifier module (D4.6), the Contextual Ego Network Manager (D4.2), and the Proximity (D4.7) and Context Aware Profiling (D4.8) modules, that in Figure 1 are named “Other Modules”.

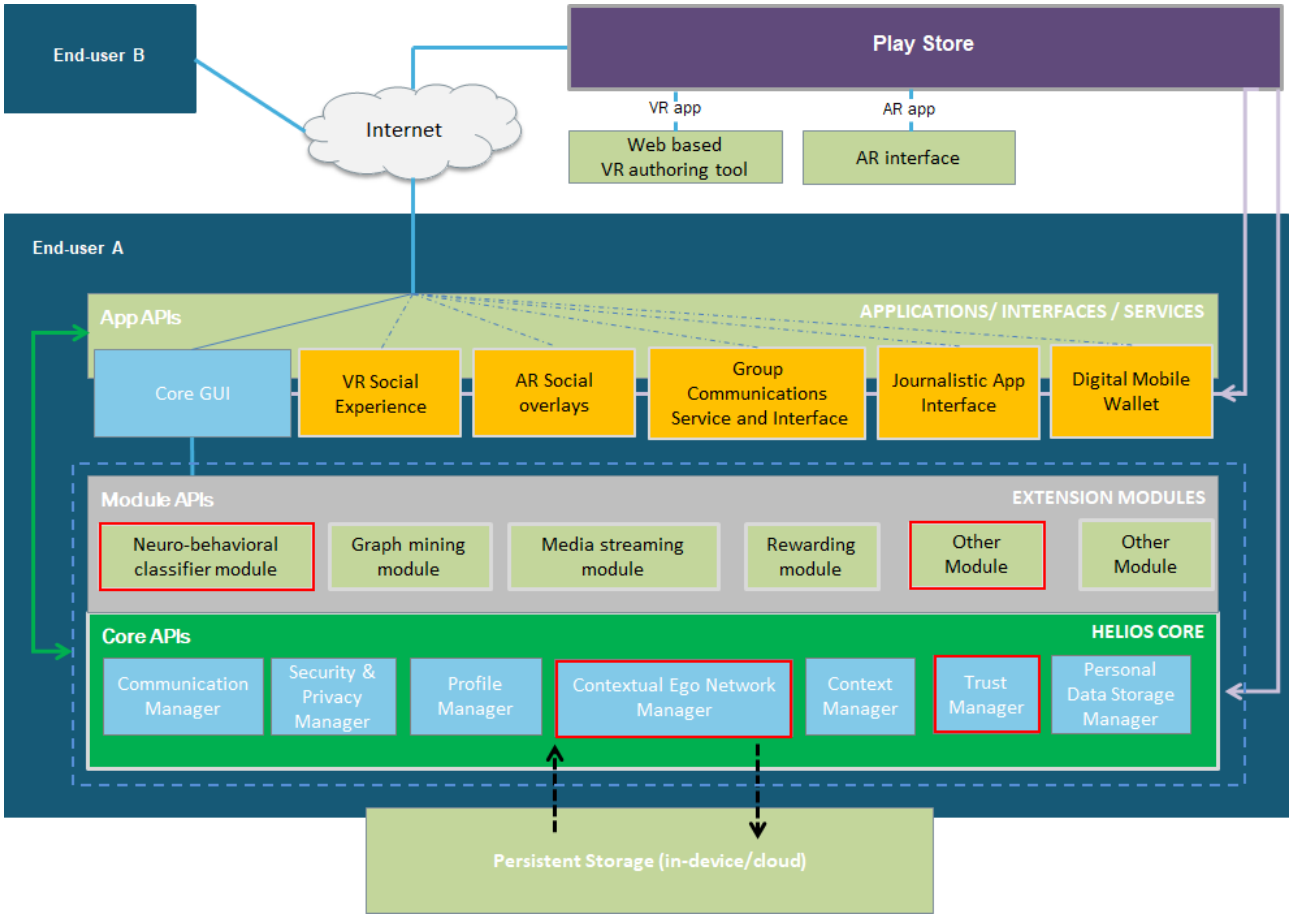

Figure 1. Helios architecture

In HELIOS, the Trust module implements the Trust model that is thoroughly explained in Section 4. It provides APIs to get trust scores related to the relationship between the ego and each alter in the Contextual Ego Network, differentiating them based on the context of reference. For each new alter, the module computes an initial trust value that only takes into consideration the information that is available at that stage; each trust value is then updated every set interval of time by also taking into account the information that can be derived from a sentimental analysis of the interactions, performed by the Neuro-Behavioural Classifier module.

The module, although depends on other modules, is implemented in plain Java and its parameters can be tuned according to the needs of the developers integrating it. The parameters can even be exposed to the end users such that every user can customise their model according to their preference. For instance, one can decide to give more weight to the number of common friends rather than proximity to achieve high trust values towards other users even if they are almost never physically close. If no parameters are suggested to the module, the defaults ones, presented in Section 4 are used. These parameters can be passed during the initialization of the module through an HashMap structure in the constructor method of the TrustManager class. The keys of the HashMap have type String, while the values have type Float. A summary of the keys used to retrieve the parameter of the model is shown in Table 5. Although there is no check that the weights used during the initialization or update phases sum up to 1, it is highly recommended that they do so. to avoid unexpected behaviours and possible errors.

ProfileSimilarityInit	Weight of the profile similarity feature, only used for the first computation.
CommonFriendsInit	Weight of the common friends feature, only used for the first computation.
ProximityInit	Weight of the proximity feature, only used for the first computation.
CommonFriends	Weight of the common friends feature.
SentimentAnalysis	Weight of the sentiment analysis feature.
Proximity	Weight of the proximity feature, used after the first computation.

Table 5. A recap of the keys to retrieve teh parameters of the trust model

The trust module uses a multithreading architecture to manage the updates of the trust values. The correspondence is that each thread manages the updates of the trust scores in a single context,

but only the contexts flagged as active have their correspondent thread active. To make sure that the trust module is correctly set up and working, users must create a TrustManager object, through its constructor, and call the method *startModule()* to create and start the threads that will update the trust scores. Each thread will start by initializing the trust value of each alter in the context according to Equation 2 and then, after that, it will enter in the loop for the trust update procedure. The loop is made of three steps:

- **Wait** for Δt seconds. The value of Δt should be adjusted according to the activity of the users. Setting a low Δt will update the trust score too often to see any significant change, while on the other hand a high Δt will result in having too old trust evaluations.
- **Update** the trust score using Equation 3, and store the updated value in the Contextual Ego Network. The value is stored in the edge connecting the Ego and the Alter.
- **Check** whether the module must be stopped, and if it is the case terminate the execution.

A recap of a thread lifecycle is shown in Figure 2.

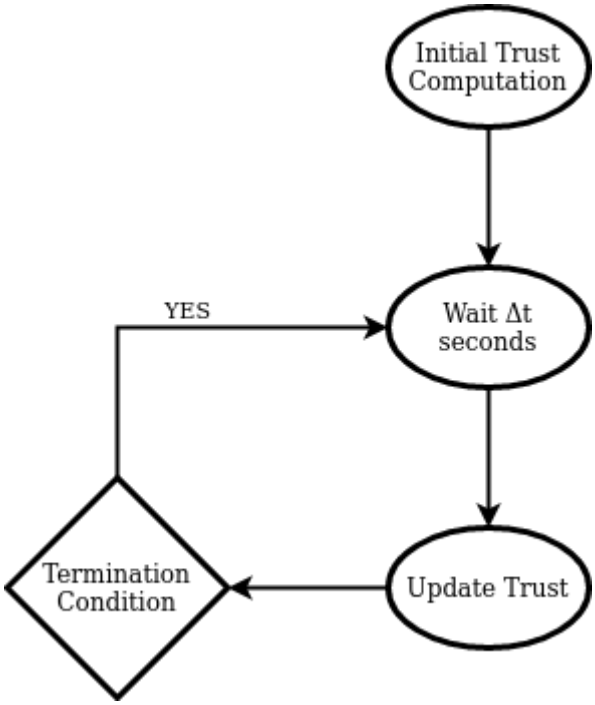

Figure 2. A flux diagram showing the steps performed by the thread of the trust model

The trust module keeps an updated estimation of the trust between the ego and its alters, but only of those in an active context. It is therefore crucial that the module is always aware of the currently active contexts and the alters that belong to them. The trust module exposes a set of methods that update its behaviour according to changes that happen to the Contextual Ego Network, such as contexts that become active/inactive or the addition/removal of nodes from a context. To make sure that these methods are called only when necessary, we implemented a Contextual Ego Network Listener, which implements a callback system when a set of predefined events on the Contextual Ego Network happen. Thanks to the listener, the model can keep track of the updates to the Contextual Ego Network and change its behaviour dynamically.

4 The HELIOS Trust Model

The social overlay of HELIOS is structurally multi-layered: each layer corresponds to a context that is identified by spatial and temporal coordinates and is effectively an ego network where in our case the ego is our user *i*.

4.1 HELIOS Trust Model Features

We first give a description of the features we take into account to compute trust; we then describe how to initialize the trust value and when and how it is updated by providing a formula; finally we show a possible discretization of the trust values, principally aimed at providing a more comprehensible result for the end user.

The following are the features that are involved in the computation of trust according to our model.

4.1.1 Sentiment Analysis

This feature is related to the analysis of interactions between two users, both at a textual level and on a media level (images and voice messages are in fact analyzed as well), in order to provide a score SA_{ij} that is representative of positive and negative emotions that can be extracted from the content exchanged between user i and user j. More specifically, the Neurobehavioural module returns 3 values, that correspond to different emotional analyses elaborated by evaluating textual messages and media sent by i to j. The first one of these values is the Attention component, that is related to the level of interaction between the two users, that is the frequency of messages and media exchanged. It is defined as a discrete value that can take up 3 possible values corresponding to 3 classes, that are *low, medium* and *high* (in relation to a low, mid or high frequency of interactions). Arousal and Valence are related to the general emotion that emerges from the content shared. They are both binary values; the former expresses the intensity of emotions that can be inferred from the messages, and can therefore be labeled as *positive* or *negative*. The latter is related to the general positive or negative emotional outcome, hence can take the values of *positive* or *negative*. Arousal and Valence are computed as an average of all the emotions that emerge from the psychological reaction analysis of different batches of interactions.

These 3 values are related to the analysis of every single message/media sent by i to j, and are therefore to be considered as historical. This allows us to compute a Sentiment Analysis score that takes into consideration possible mutations of the conversation mood, but still considers the previous history as its solid base. It is therefore realistic in representing situations where the emotions slowly become more positive/more negative monotonically, thus portraying a positive

growth or a negative distancing of the relationship; but in case of sudden and isolated mood shifts, the overall sentimental score is not affected as much.

SA_{ij} is computed by combining the 3 aforementioned discrete emotional scores. In particular, the values of Arousal and Valence are taken together: each of the 4 possible combinations of them is assigned a value belonging to the set {0.25, 0.5, 0.75, 1}, that is then multiplied to a separate real value assigned to Attention belonging to the set {0.33, 0.66, 1} (see Table 7). Table 6 summarizes the real values that are assigned to the features returned by the Neurobehavioural moule.

Valence	**Arousal**	**Value in the trust model**
Positive	Positive	1
Positive	Negative	0.75
Negative	Negative	0.5
Negative	Positive	0.25

Table 6. This table shows the score assigned by the trust model to each of the possible couples of values corresponding to the Valence and Arousal features, that are returned by the NeuroBehavioural module.

Attention	**Value in the trust model**
High	1
Medium	0.66

Low	0.33

Table 7. The score assigned by the trust model to each of the possible values corresponding to the Attention feature, that is returned by the NeuroBehavioural module.

4.1.2 Content profile similarity

This parameter, PS_{ij}, is representative of how similar two users i and j are, based on what they share on their profiles. It is a symmetric value ($PS_{ij} = PS_{ji}$), and the Content Aware Profiling Module based on the image files stored on the device computes it. Such images are processed to extract an interest vector, which is then used as the basis for computing the similarity score (with the corresponding interest vector of the other user) by applying cosine similarity.

When computing this value, it is important to be concerned with the information availability problem: it may be the case that a user does not want to share its interests information with others, and this makes content profile similarity uncomputable. For this reason, this feature has to be taken as optional in the model.

This feature will be defined in more detail according to what will be established in Task 4.8.

4.1.3 Common friends

This parameter, CF_{ij}, is the ratio between the number of friends user i has in common with user j and the total number of friends of user i. It is therefore a value in the interval [0,1] ($CF_{ij} \in [0,1]$). Putting the number of common friends in scale with respect to the total number of friends is important, so as not to incur possible overstated values due to the fact that a user tends to accept a lot of friendship requests or tends to send many and therefore the probability of having many common friends with other users is higher. Under the same number of common friends between user i and user j, the lower is the number of friends of user i, the higher is the impact of the common friends.

4.1.4 Proximity

This parameter, P_{ij}, is representative of the physical closeness of user i and user j in the real world (detected through sensors on their devices) and obtained through the proximity module (i.e. organic social graph creation). Considering the fact that virtual relationships are within a specific spatial domain since a context itself has a spatial specification, we assume that two users belonging to the same context are physically close to a certain extent, so this value can be computed. This value is based on how frequently the users find themselves in a close spatial proximity of under 5 meters within a context. The value returned by the proximity module is normalised, and the number of times, the duration, as well as the distance between the devices is stored. Proximity is therefore computed in a certain (this is a value that can be changed) period of time two devices are close divided by the total time (in predefined period of time), and lies in the interval [0,1] ($P_{ij} \in [0,1]$). One should note that there is a decaying factor if two devices are infrequently in the spatial domain. It is important to highlight the fact that proximity per se is not an indicator of possible trust between two people, and therefore has a very low influence on the final trust value. The influence proximity has is more related to the span of time two users have been physically close over the total span of existence of the context itself.

4.2 Trust Model Description

As previously stated when we considered its properties, trust is context dependent. In the HELIOS scenario, which is intrinsically constituted by multiple contexts, trust cannot be represented with a single value: we require a trust score for each context of the Contextual Ego Network, and therefore vectors of trust scores.

For simplicity, without loss of generality, from now on we will describe the computation of the trust score between the ego i and a generic alter j by implicitly referring to a generic context. In the trust module of HELIOS the formulas that follow are implemented by keeping track of all the parameters for each context; the module, thus, computes a trust score for each context.

When an edge is created from user i to user j through a process of link detection, that is out of the scope of this document, or by manual addition by the ego i or the alter j, the trust value T_{ij} needs to be initialized in a certain way. This initial value will label the edge going from user i to user j, and defines the trust that the former initially has towards the latter. Among the features that we just mentioned, the ones that are meaningful in this phase are PS_{ij}, CF_{ij} and P_{ij}. As we mentioned before, content profile similarity can only be computed if the other user is willing to share its content preferences information with the ego. For this reason, this parameter has to be considered as optional. Proximity can be seen as a feature that tells us, in this initial phase, if the two people

have had any physical encounters before they added each other in the HELIOS application; if that is the case, a slight increase in the initial trust score should be given.

The initial trust value is therefore computed as described in Equation 2.

$$T^0_{ij}=\alpha_0 PS_{ij}+\beta_0 CF_{ij}+\omega_0 P^0_{ij}$$

Equation 2. Initial Trust Value

Where α_0, β_0 and ω_0 are configurable parameter weights such that $\alpha_0 + \beta_0 + \omega_0 = 1$ and

$CF_{ij}= cf_{ij} / tf_i$ (cf_{ij} is the number of common friends between i and j, while tf_i is the total number of friends of i).

Once i and j are connected by an edge in a context, it is likely that they are often physically close or virtually interact. For example by sharing chat messages, pictures, videos or voice notes, or by commenting on each other's posts. The trust value between them is updated every determined amount of time Δt: when Δt seconds have elapsed from the previous timestamp t_{k-1}, the trust value is computed over a so-called time window, that encapsulates the interval $(t_{k-1}, t_k]$ ($k\geq1$, $t_k = t_{k-1} + \Delta t$). The sentiment analysis module analyzes the interactions originating from user i directed to user j and when triggered by the trust module returns a sentiment score that is an average of all the sentimental scores of batches of interactions occurred in the current time window.

If the users are physically close most of the time, they probably interact mostly by speaking to each other, without the need of using the social platform. However, there is no way to evaluate these "real life" interactions, so the proximity score, as said previously, is only determined by the actual amount of times of physical closeness.

Equation 3 is the formula used to compute trust over the k-th time window:

$$T^k_{ij} = \beta CF_{ij} + \gamma SA_{ij} + \omega P^{k'}_{ij}$$

Equation 3. Trust computation over the k-th time window

Where SA_{ij} is the sentimental score related to the messages and media sent by i to j, and $P^{k'}_{ij}$ is the proximity score evaluated over the k'-th time frame. As we mentioned before, proximity scores are actually computed once every specific interval of time that is different from the one adopted for the trust computation. This is because the Δt interval for trust is likely to be small, in order to have

fairly frequent updates of its score, while the proximity score is tied to real-life encounters, that to be meaningful need to occur in a bigger span of time.

β, γ, ω are weights such that $\beta + \gamma + \omega = 1$, in particular ω lowers the impact of the proximity score, so it has to be lower than the other two weights (we could for example take it as $\omega = 0.1$).

Since all the parameters lie in the interval [0,1], and all the weights sum to 1, T^k_{ij} will also be in this interval ($T^k_{ij} \in [0,1]$).

The end-user of the application is probably not able to interpret a value in a continuous interval correctly. For this reason, a discretization with natural language labelling is necessary, also to give users the possibility of defining a personal sharing policy, that would require a first phase of discretization in any case.

Table 8 provides a possible discretization for the trust value. A trust label in natural language is assigned to each chosen sub-interval of [0,1].

Interval	**Trust label**
[0, 0.2)	Distrust
[0.2, 0.5)	Partial distrust
[0.5, 0.8)	Partial trust
[0.8, 1]	Trust

Table 8. This table shows a possible discretization over the interval [0,1] of the Trust score.

5 helios.TrustManager: APIs

The Trust Manager is a multithreaded module that periodically gathers the information it needs from the Contextual Ego Network and the NeuroBehavioural module, the Proximity Module and the Context Aware Profiling module, computes a new trust value for each node in each active context, and then updates the Contextual Ego Network with the newly computed trust value by storing it onto the corresponding edge. Each running thread of the Trust Manager corresponds to an active context; whenever a context becomes inactive, the respective thread is put to a wait state through a condition variable. Other HELIOS modules can then utilize the calculated trust value directly from the CEN and based on the user's policy make decisions, e.g., what part of the profile can be shared, how urgent the message from an alter is to the ego, and so on.

Table 9 describes the APIs provided by the Trust Manager module.

Method	**Parameters**	**Description**
TrustManager *constructor method*	ContextualEgoNetwork c *Reference to the module the Trust Manager will retrieve the social information from*	Creates an instance of the Trust Manager. The weights of the model are set according to what is specified in the *modelWeights* hash map. If some weights are not specified in the map, they will be set to default. Since the Profile Similarity component is optional, to decide not to take it into account in the computation, specify a 0 weight for it (that is, put in the map the pair <ProfileSimilarityInit, 0>).
	int deltaT *Time interval each thread will wait before re-computing new values of trust for its context*	
	HashMap<String, Float> modelWeights *Weights of the model. This map has to contain the following keys:*	

	ProfileSimilarityInit *Weight of the profile similarity parameter for trust initialization* CommonFriendsInit *Weight of the common friends parameter for trust initialization* ProximityInit *Weight of the proximity parameter for trust initialization* CommonFriends *Weight of the common friends parameter for trust computation* SentimentAnalysis *Weight of the sentiment analysis parameter for trust computation* Proximity *Weight of the proximity parameter for trust computation* *The keys for which there is no correspondence in the map will be set to default.*	
startModule		Makes the Trust Manager instance run. The contexts for which trust

		will be initially computed are the ones that are active when this method is called. To notify the creation/activation of new contexts the methods *newContext* and *activateContext* have to be called.
newContext	Context c *New context that is added to the Contextual Ego Network*	Notifies the Trust Manager that a new context has been added to the Contextual Ego Network. The Trust Manager will instantiate a thread related to this new context.
addAlterToContext	Node alter *Alter that has been newly added to a context* Context c Context the new alter has been added to	Notifies the Trust Manager that a new alter has just been added to a context in the Contextual Ego Network. The Trust Manager will initialize the trust score towards this alter.
activateContext	Context c *Context whose status is switched from inactive to active*	Notifies the Trust Manager that a context has become active. The Trust Manager will in turn notify the related thread.
deactivateContext	Context c *Context whose status is switched from active to inactive*	Notifies the Trust Manager that a context has become inactive. The Trust Manager will, in turn, put on hold the thread related to that context.

getTrust	Context c *Context in which the trust value towards* alter *has to be retrieved*	Returns the last trust value towards a node in a specific context stored on the Contextual Ego Network.
	Node alter *Alter towards which the trust value in* context *has to be retrieved*	
getProximity	Context c *the context the user finds themselves in*	Returns the last proximity frequency (normalized) of the device/person in a specific context and a specific period of time.
	int delta TP *time proximity window*	

Table 9. The APIs provided by the Trust Manager Module

5.1. Initialization and lifecycle

To start the Trust Manager, it is necessary to instantiate the object, by calling the constructor method, and then call the *startModule* method. The manager will automatically instantiate all the threads related to the active contexts.

The Trust Manager invokes the following methods whenever precise events take place in the Contextual Ego Network. Such methods are invoked automatically by some callbacks that are registered on the Contextual Ego Network.

1. Whenever a new context is added to the Contextual Ego Network, the *newContext* method is triggered: the Trust Manager instantiates a new thread for the new context, and from that moment on (up until such context is deactivated) trust values for the nodes in it are computed every *deltaT* seconds.
2. Whenever a new alter is added to a context in the Contextual Ego Network, the *addAlterToContext* method is triggered: the thread related to that context is notified and gives an initial trust score to the new alter. From that moment on, up until such context is deactivated, trust values for the new alter in that specific context are computed every *deltaT* seconds.
3. Whenever a context's status is switched to active, the *activateContext* method is triggered: the Trust Manager notifies the thread related to that context, that from that moment on (up until such context is deactivated again) starts computing trust values for all the nodes in it every *deltaT* seconds.
4. Whenever a context's status is switched to inactive, the *deactivateContext* method is triggered: the Trust Manager notifies the thread related to that context, that computes a last set of trust scores for all the nodes in it and is then put on hold on a condition variable (up until such context is activated again).

6 Conclusions and Future Works

Trust is a broad concept that finds possible applications in multiple fields of study. In the Information Technology area, many researchers have worked on giving it definitions and developing trust models. Although many works have been written about this topic in the OSNs literature and in the decentralized networks one, only few ventured in the area of DOSNs.

Defining a proper trust model for decentralized online social networks presents its set of peculiar and non-negligible problems, such as data availability, privacy issues and resource consumption optimization. HELIOS belongs to this category of social networks, so it has been necessary to approach this task in a novel way. We first established which features the model has to exploit to compute trust scores towards other nodes in the various contexts, and then defined a preliminary version of it, as the weighted sum of such features. Finally, we implemented this first version of the model and provided the APIs that other modules will be able to call.To compute the trust scores, this module needs to call methods provided by the NeuroBehavioural (D4.6), Proximity (D4.7), and Context Aware Profiling (D4.8) modules.

6.1 Future Works

Future developments of our work will be focused on refining the model. In particular, we need to further investigate the usage of the content profile similarity feature, that is the output of the task T4.8 which is still on hold. Moreover, we will perform a thorough study of parameters, such as the weights, the temporal window of the model and the time frame related to the proximity feature; these will be established at a theoretical level and through model tests.

Including more features to the trust model is also an interesting topic. For example, a feature related to the cybersecurity posture of the other peers (trustees) by performing automatic measurements on the peers without violating their privacy (i.e., mostly from public information). The measurements could be performed partially in the Trust module or in the HELIOS core (security & privacy and/or communication module), although these implementations are not available/planned at the moment. The measurements could include checking available information of the trustee, e.g., encryption/authentication strength, used HELIOS module/application versions, confidence in the peer (IP blacklists, reputations), and observed bad/malicious behavior in the HELIOS network (Denial of Service attacks, flooding). A set of these measurements could provide a cybersecurity posture feature into the model. For example, if another peer is using an outdated

(possibly vulnerable) version of the application, the trust score towards that peer can be lowered accordingly. It should be noted, that in some measurements, e.g. IP reputation, the client would need to contact outside sources in order to compare the information received from the other peer against trusted third party actors. Cybersecurity posture features could possibly affect the overall trust value directly by a weighting factor. Being a more technical feature, its relevance to the calculation of the trust value between peers is more closer to the content profile similarity feature, that also needs more investigations.

References

[1] L. Mui, M. Mohtashemi, A. Halberstadt, A computational model of trust and reputation, in: Proceedings of the 35th International Conference on System Science, 2002

[2] D. Olmedilla, O. Rana, B. Matthews, W. Nejdl, Security and trust issues in semantic grids, in: Proceedings of the Dagsthul Seminar, Semantic Grid:The Convergence of Technologies, vol. 05271, 2005

[3] Netrvalova, Arnostka, and Jiri Safarik. "Trust model for social network." Department of Computer Science and Engineering. University of West Bohemia, Czech Republic (2011).

[4] Gambetta D., 2000. Can We Trust Trust? In Gambetta, Diego (ed.) Trust: Making and Breaking Cooperative Relations, electronic edition. Department of Sociology, University of Oxford, chapter 13, 213-237.

[5] Nepal, Surya & Sherchan, Wanita & Paris, Cécile. (2011). STrust: A Trust Model for Social Networks. 10.1109/TrustCom.2011.112

[6] Liu, Guanfeng, Yan Wang, and Mehmet Orgun. "Trust inference in complex trust-oriented social networks." 2009 International Conference on Computational Science and Engineering. Vol. 4. IEEE, 2009.

[7] J. Golbeck and J. Hendler. Inferring trust relationships in web-based social networks. ACM Transactions on Internet Technology, 6(4):497–529, 2006.

[8] Adali, Sibel, et al. "Measuring behavioral trust in social networks." 2010 IEEE International Conference on Intelligence and Security Informatics. IEEE, 2010.

[9] HOFFMAN, Robert R., et al. The dynamics of trust in cyberdomains. IEEE Intelligent Systems, 2009, 24.6: 5-11.

[10] F. Cornelli, E. Damiani, S. D. Capitani, Choosing Reputable Servents in a P2P Network, in: Proc. of the 11th International World Wide Web Conference, 2002.

[11] Gómez Mármol, F., Martínez Pérez, G. & Gómez Skarmeta, A.F. TACS, a Trust Model for P2P Networks. Wireless Pers Commun 51, 153–164 (2009). https://doi.org/10.1007/s11277-008-9596-9

[12] N. Stakhanova, S. Basu, J. Wong and O. Stakhanov, "Trust framework for P2P networks using peer-profile based anomaly technique," 25th IEEE International Conference on Distributed Computing Systems Workshops, Columbus, OH, 2005, pp. 203-209.

[13] Tang, W., Ma, Y., & Chen, Z. (2005). Managing trust in peer-to-peer networks. Journal of Digital Information Management, 3(2), 58

[14] Y. Wang and J. Vassileva, "Trust and reputation model in peer-to-peer networks," Proceedings Third International Conference on Peer-to-Peer Computing (P2P2003), Linkoping, Sweden, 2003, pp. 150-157.

[15] Karl Aberer and Zoran Despotovic. 2001. Managing trust in a peer-2-peer information system. In Proceedings of the tenth international conference on Information and knowledge management (CIKM '01). Association for Computing Machinery, New York, NY, USA, 310–317

[16] S. Buchegger and J.Y.L. Boudec, "A Robust Reputation System for P2P and Mobile Ad-hoc Networks," Proc. 2nd Workshop on the Economics of Peer-to-Peer Systems, 15 Nov. 2004

[17] Zhao, H., & Li, X. (2013). VectorTrust: trust vector aggregation scheme for trust management in peer-to-peer networks. The Journal of Supercomputing, 64(3), 805-829

[18] Li, Juan, and Qingrui Li. "Decentralized self-management of trust for mobile ad hoc social networks." International Journal of Computer Networks & Communications 3.6 (2011): 1.

[19] B. Qureshi, G. Min and D. Kouvatsos, "A Framework for Building Trust Based Communities in P2P Mobile Social Networks," 2010 10th IEEE International Conference on Computer and Information Technology, Bradford, 2010, pp. 567-574.

[20] Netrvalova, Arnostka, and Jiri Safarik. "Trust model for social network." Department of Computer Science and Engineering. University of West Bohemia, Czech Republic (2011).

[21] Nepal, Surya & Sherchan, Wanita & Paris, Cécile. (2011). STrust: A Trust Model for Social Networks. 10.1109/TrustCom.2011.112.

[22] Yu, Bin, and Munindar P. Singh. "A social mechanism of reputation management in electronic communities." International Workshop on Cooperative Information Agents. Springer, Berlin, Heidelberg, 2000

[23] Liu, Guanfeng, Yan Wang, and Mehmet Orgun. "Trust inference in complex trust-oriented social networks." 2009 International Conference on Computational Science and Engineering. Vol. 4. IEEE, 2009.

[24] Kuter, Ugur, and Jennifer Golbeck. "Sunny: A new algorithm for trust inference in social networks using probabilistic confidence models." AAAI. Vol. 7. 2007.

[25] M. Maheswaran, H. C. Tang and A. Ghunaim, "Towards a Gravity-Based Trust Model for Social Networking Systems," 27th International Conference on Distributed Computing Systems Workshops (ICDCSW'07), Toronto, Ont., 2007, pp. 24-24, doi: 10.1109/ICDCSW.2007.82