SocEDA

Cloud based platform for large scale social aware EDA

ANR-10-SEGI-013

Document name:	Social-aware Event Modeling and Matching
Document version:	1.0
Task code:	2.1
Deliverable code:	D2.1.2
WP Leader (organisation):	LIRIS / INSA Lyon
Deliverable Leader (organisation):	LIRIS / INSA Lyon
Authors (organisations):	• Omar HASAN (LIRIS / INSA Lyon) • Sonia BEN MOKHTAR (LIRIS / INSA Lyon) • Lionel BRUNIE (LIRIS / INSA Lyon)
Date of first version:	2012-05-16

Change control

Changes	Author / Entity	Code of version
Creation of the document	• **Omar HASAN** (LIRIS / INSA Lyon) • **Sonia BEN MOKHTAR** (LIRIS / INSA Lyon) • **Lionel BRUNIE** (LIRIS / INSA Lyon)	

Table of Contents

1. Introduction

Our goal in this activity is to enable services to subscribe to and match events based on social parameters. The social and trust infrastructure established in the first deliverable allows us to create the social parameters used by services for event subscription and matching.

The Social Filter operates on a social network of services (or nodes) to compute the strength of the relationships between them. The Event Cloud calls the social filter and submits a source node and a list of target nodes. The social filter uses a trust inference algorithm to compute the strength of the relationships between the source node and each target node.

The first possible case is that the source node has a direct social relationship with all target nodes. In this case the social filter computes relationship strength as a weighted sum of normalized characteristics of the relationship between the two nodes. These characteristics include the time of last interaction, interaction count, declared trust, number of mutual nodes, etc.

The second possible case is that no direct social relationship exists between the source node and one or more of the target nodes. In this case the social filter computes relationship strength as the maximum flow of declared trust from the source node to a target node in the social network.

The social filter returns the computed strength of the relationships back to the event cloud. The event cloud may use a relationship strength threshold or a ranking of the destination nodes to select the most trustworthy nodes. Consequently the event cloud can accept to process the events that the source node received from target nodes with which it has high relationship strength whereas it can discard the ones received from target nodes with low relationship strength.

2. Architecture of the Social Filter

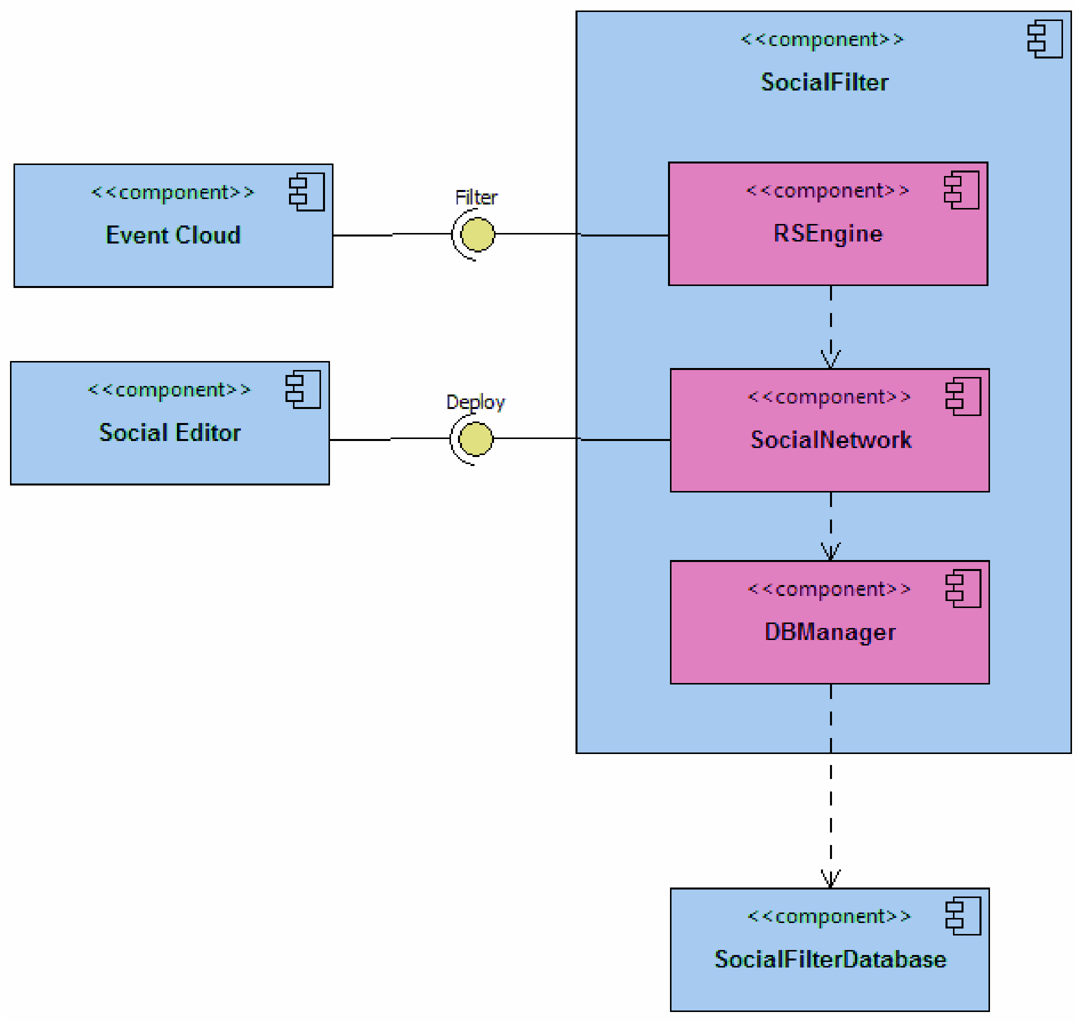


Figure 1: Architecture of the Social Filter

We revisit the architecture of the Social Filter that we described initially in the Deliverable D2.1.1. The updated architecture of the Social Filter eliminates web service interfaces between components in order to reduce the latency of invocation. This updated architecture implies that the Social Filter can be invoked with very high frequency by the Event Cloud without the added latency of a web service interface. The revised architecture provides two clear interfaces to the social filter. The deploy interface allows components to deploy a social network of services. The filter interfaces allows the event cloud to filter events according to the relationships between services.

3. Deploying a Social Network of Services

In this section, we describe a social network of services (Section 3.2) and the options available for deploying such a social network (Sections 3.3 and 3.4). We begin by describing a conventional social network in Section 3.1.

3.1. Social Networks

As established in the deliverable D2.1.1, a social network is a composition of nodes and the relationships between them. The nodes in a social network may be individuals or collectives of individuals. The relationships between nodes are founded on human ties such as friendship, membership in the same family or organization, mutual interests, common beliefs, trade, exchange of knowledge, geographical proximity, etc.

The most commonly discussed characteristics of social relationships include *roles*, *valence*, *provenance*, *history*, and *strength* [Mika2011].

- **Roles.** A social relationship is defined by the roles that are associated with it. For example, the roles of employer and employee define the relationship of employer--employee in a professional setting. The same pair of nodes may take on different roles in a parallel relationship. For example, a employer--employee relationship may be complemented by a neighbor--neighbor relationship.
- **Valence.** A social relationship can have positive, negative, or neutral sentiments associated with it. For example, an individual may like, dislike, or be apathetic towards another individual.
- **Provenance.** Some attributes of a social relationship may be asymmetric, that is, perceived differently by the individual participants of the relationship. For example, a sentiment of *like* from one node may not be reciprocated by the other node in the relationship.
- **Relationship history.** Social relationships have a temporal dimension. A social relationship may evolve with time through interactions or the absence thereof. The history of a social relationship can be considered as an indicator of the current and future status of the relationship. For example, a long positive relationship in the past is likely to be followed by a positive relationship in the present and in the near future.
- **Strength.** Strength of a tie (or social relationship) is a quantifiable property that characterizes the link between two nodes [Petroczi2007]. The notion of tie strength was first introduced by sociologist Mark Granovetter in his influential paper "The strength of weak ties" [Granovetter1973] published in 1973. He defined the strength of a tie as a "combination of the amount of time, the emotional intensity, the intimacy (mutual confiding), and the reciprocal services which characterize the tie" [Granovetter1973].

3.2. A Social Network of Services

The use cases for the SocEDA project (such as the one described in the deliverable D5.3.1) consider an Internet of Services, with the assumption that the services involved can publish their events and create event clouds.

It is evident that social relationships exist between humans. However, we argue that social relationships can also exist between services. The characteristics that we have identified for social relationships between humans also apply to relationships between services. Let's consider each of the following characteristics previously discussed in Section 3.1 in the context of social relationships between humans:

- **Roles.** Roles also exist between services. At a fundamental level, the relationship between two services can be characterized by the role of consumer-provider or vice versa. More complex roles include composition, two services belonging to the same organization, etc.
- **Valence.** A service can have a positive, negative, or neutral opinion towards another service based on the quality of service that it has received.
- **Provenance.** Attributes of a relationship between two services can also be asymmetric. For example, one service can have a positive valence towards the other however the opposite may be true in the other direction.
- **Relationship history.** The relationship between two services also has a temporal dimension. The relationship can evolve with time through varying intensity of interactions.
- **Strength.** The strength of a relationship between two services is characterized by dimensions and indicators similar to those for social relationships between humans. For example: the *intensity* of the relationship as indicated by interaction frequency and the amount of information exchanged, the *intimacy* of the relationship as depicted by the recentness of interaction and the amount of declared trust; the *duration* of the relationship as measured by the time past since the first interaction; and *structural factors* such as the number of mutual nodes.

Our argument that social relationships exist between services is also supported by Qinyi et al. [Qinyi2009] and Maamar et al. [Maamar2011].

Our model for the description of a social relationship between two services is given in Table 1.

Element	Data Type	Comments
source_node	Node ID	
target_node	Node ID	
role	Role ID	
interaction_count	{0,1,2,3,…}	Number of interactions between the two nodes
amount_of_data_exchanged	{0,1,2,3,…}	Amount of data exchanged between the two nodes. The data unit can be considered as kilobytes, megabytes, etc.
time_of_last_interaction	Timestamp	The time of the latest interaction between the two nodes
trust	[0,1]	A value representing the amount of subjective trust that the source node holds in the target node. 0: weak, 1: strong. The context of trust is the general integrity of the target node.
time_of_first_interaction	Timestamp	The time of the earliest interaction between the two nodes
mutual_nodes_count	{0,1,2,3,…}	The count of nodes that have relationships with both the source and the target nodes

Table 1: Model for description of a social relationship between services

3.3. The Social Network Deployment API

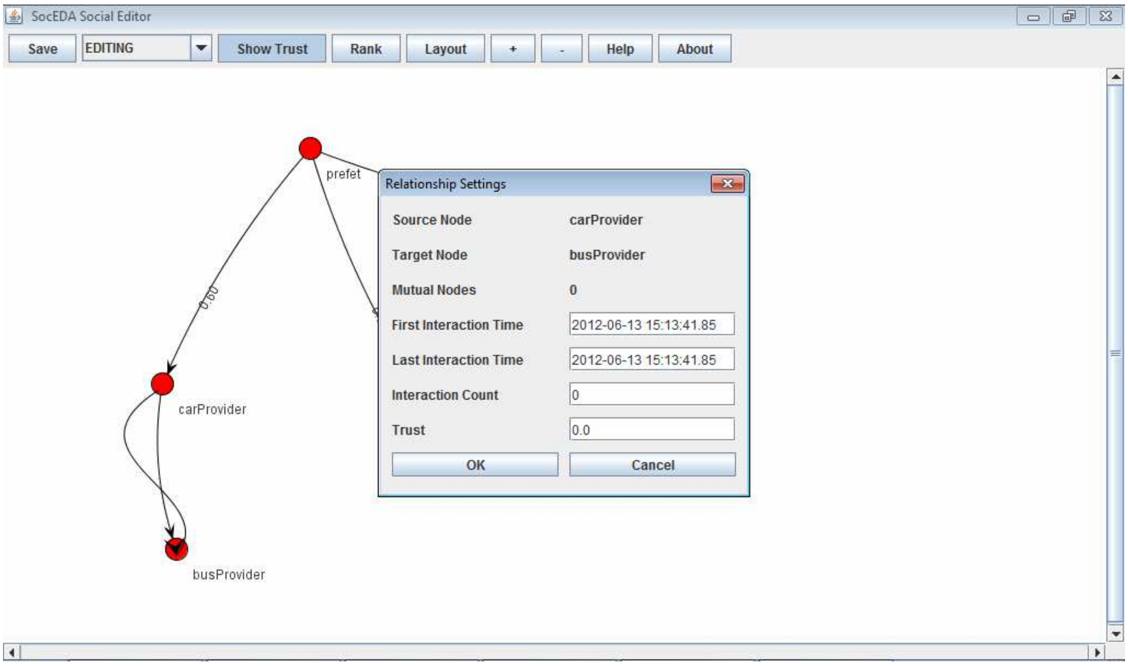
We implemented a Social Network Deployment API that permits components in the SocEDA architecture to deploy and update the social network of services to reflect its evolution. The social network deployment API allows adding and modifying nodes and relationships in the social network. The fields given in Table 1 can be updated for a social relationship using this API.

3.4. The Social Editor

The Social Editor is a graphical tool that allows the user to view and manually edit the social network of services. The social editor uses the social network deployment API to update the social network of services.

3.4.1. Editing the social network

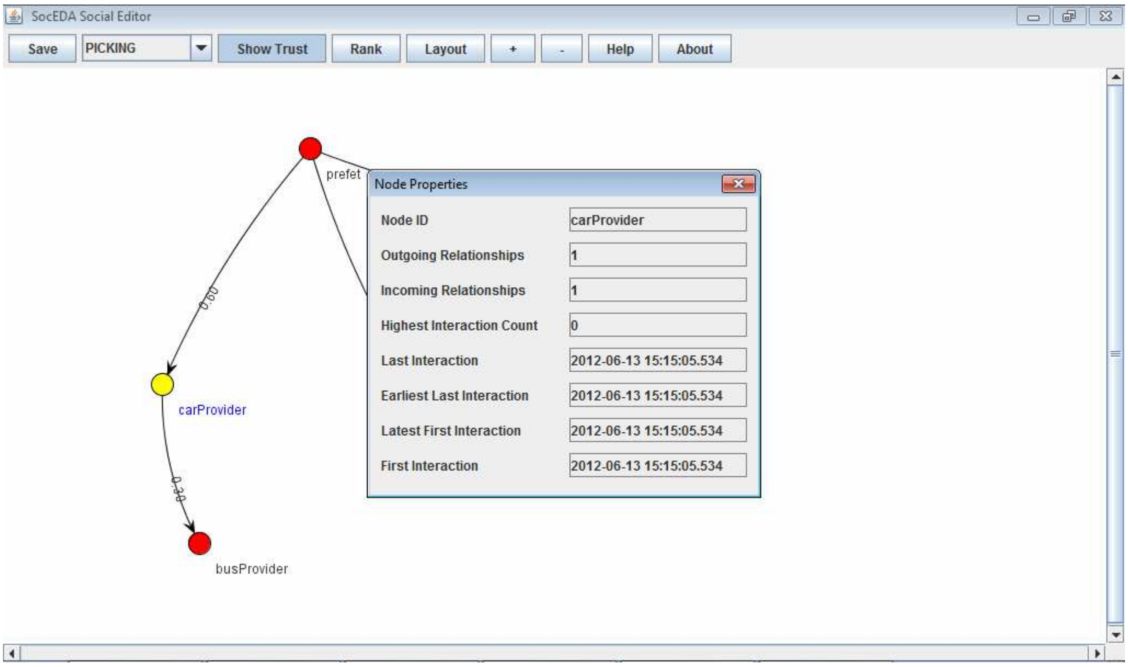
Figure 2 shows a new social relationship being created from the node *carProvider* towards the node *busProvider*. The user can declare relationship properties such as the time of first interaction, the time of last interaction, the interaction count, and the amount of perceived direct trust in the target node.



Source Node carProvider
Target Node busProvider
Mutual Nodes 0
First Interaction Time 2012-06-13 15:13:41.85
Last Interaction Time 2012-06-13 15:13:41.85
Interaction Count 0
Trust 0.0

Figure 2: Add nodes and relationships

Figure 3 shows the properties of a node in the social network. The properties of a node other than the name of the node are auto-deduced by the social network component of the social filter. The properties auto-deduced include the number of outgoing relationships, the number of incoming relationships, highest interaction count among adjacent nodes, the time of last interaction with adjacent nodes, the time of earliest last interaction with adjacent nodes, the time of latest first interaction with adjacent nodes, and the time of first interaction with adjacent nodes. Several properties are similarly also auto-deduced for relationships, such as the number of nodes that both the source and the target node have social relationships towards. The above information is used by the social filter in computing the strength of the relationship between the source and the target node as described in Section 4.2.



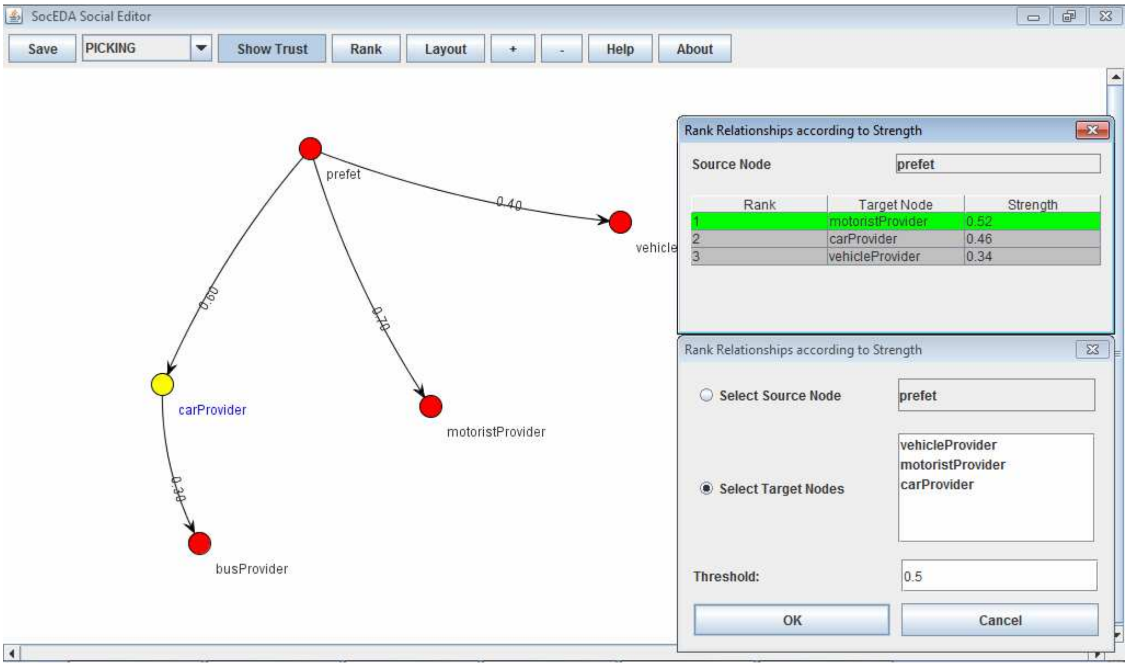
Node Properties

Node ID: carProvider
Outgoing Relationships: 1
Incoming Relationships: 1
Highest Interaction Count: 0
Last Interaction: 2012-06-13 15:15:05.534
Earliest Last Interaction: 2012-06-13 15:15:05.534
Latest First Interaction: 2012-06-13 15:15:05.534
First Interaction: 2012-06-13 15:15:05.534

Figure 3: Social information is declared by the user as well as deduced from node and relationship properties

3.4.2. Visualization of the social filtering process

The social editor also enables visualization of the social filtering process. Figure 4 shows the social relationships of a source node *prefet* towards target nodes *vehicleProvider*, *motoristProvider*, and *carProvider* ranked according to the computed strength of the relationships. The linear model described in Section 4.2 is used in this scenario since node *prefet* has a direct relationship with all three target nodes. The relationship with *motoristProvider* is highlighted in green as it meets the threshold of 0.5. In a similar scenario, the event cloud would use such ranking and threshold to select the events to process and the ones to drop.



Rank Relationships according to Strength

Source Node: prefet

Rank	Target Node	Strength
1	motoristProvider	0.52
2	carProvider	0.46
3	vehicleProvider	0.34

Rank Relationships according to Strength

○ Select Source Node: prefet

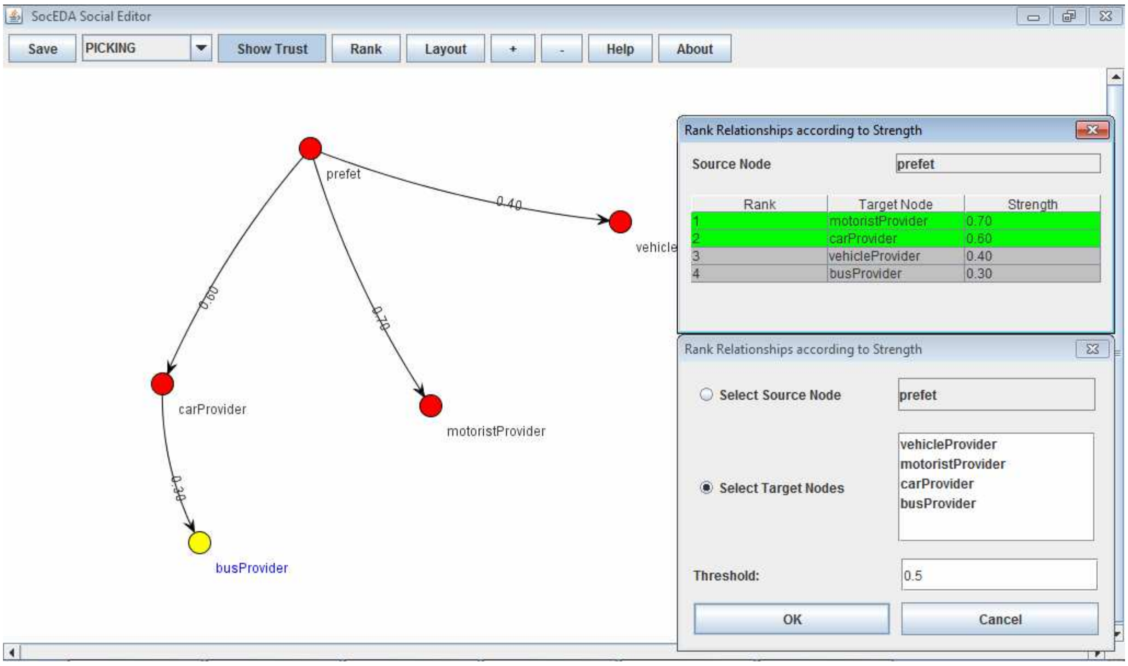
● Select Target Nodes: vehicleProvider, motoristProvider, carProvider

Threshold: 0.5

OK | Cancel

Figure 4: A linear model is used for computing relationship strength of direct relationships

Figure 5 shows the social relationships of the source node prefet towards target nodes *vehicleProvider*, *motoristProvider*, *carProvider*, and *busProvider* ranked according to the computed strength of the relationships. The maximum network flow model for computing indirect trust described in Section 4.3 is used in this scenario since node *prefet* does not have a direct relationship with the target node *busProvider*.



Rank Relationships according to Strength

Source Node: prefet

Rank	Target Node	Strength
1	motoristProvider	0.70
2	carProvider	0.60
3	vehicleProvider	0.40
4	busProvider	0.30

Rank Relationships according to Strength

○ Select Source Node: prefet

● Select Target Nodes: vehicleProvider, motoristProvider, carProvider, busProvider

Threshold: 0.5

Figure 5: Maximum network flow is used for computing relationship strength of indirect relationships

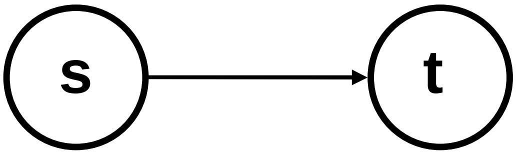
4. Social Event Filtering

4.1. The Social Filter API

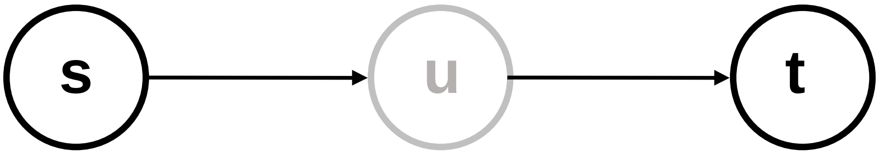
We have developed a Social Filter API that computes the strength of the relationship between a source node and a target node in the social network of services. The social filter API address two scenarios. A direct relationship may exist between the source and the target nodes or no direct relationship may exist between the two nodes. The linear model described in Section 4.2 is employed for computing the strength of the relationships in the first case when a direct relationship exists. The maximum network flow model described in Section 4.3 is used when a direct relationship does not exist.

Two Scenarios:

1. A **direct relationship** exists between the **source** and the **target** nodes.



2. No direct relationship exists between the **source** and the **target** nodes.



4.2. Linear Model

Computing relationship strength between directly connected nodes using the linear model

$$strength(s,t) = \Sigma_{social\_characteristics}\ (normalized\_characteristic * predictive\_power)$$

$$\begin{aligned} strength(nodeA,nodeB) = &\\ & Interaction\_count && * p_1 \\ + \quad & Time\_of\_last\_interaction && * p_2 \\ + \quad & Declared\_trust && * p_3 \\ + \quad & Time\_of\_first\_interaction && * p_4 \\ + \quad & Mutual\_nodes\_count && * p_5 \end{aligned}$$

$$\Sigma p_i = 1$$

4.3. Maximum Network Flow of Declared Trust

4.3.1. Trust recommendation and propagation

Establishing trust in an unknown entity through trust recommendation and propagation takes advantage of the possible transitivity of trust. Let's say that Alice wishes to establish trust in an unknown individual Carol. If another individual Bob trusts Carol then he could give a recommendation to Alice about Carol's trustworthiness. Taking Bob's trust recommendation and her own trust in Bob into account, Alice may establish a trust relationship with Carol. Thus a transitive path of trust that leads from Alice to Bob to Carol, enables Alice to develop trust in Carol. If Alice wishes to establish trust in Carol through Bob's recommendation, we say that Bob's trust in Carol has propagated to Alice.

Trust propagation builds upon multiple trust recommendations to help establish trust in an unknown entity. Let's extend the example presented in the first paragraph: Alice trusts Bob and Bob trusts Carol. We further assume that Carol trusts Dave. Alice may establish trust in Dave as a result of the following two recommendations: 1) the first recommendation builds Bob's direct trust in Dave, and 2) now since Bob trusts Dave, Alice can establish trust in Dave through a second recommendation. This sequence of recommendations is referred to as trust propagation.

In our social filter API, we employ a form of trust propagation that models the problem as computing the maximum network flow in a flow network.

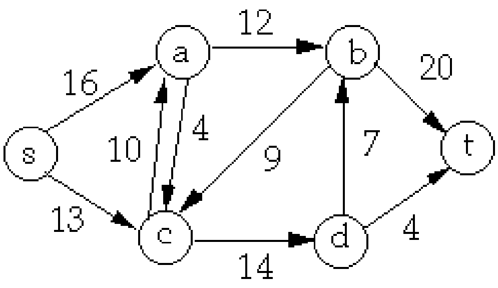
4.3.2. Flow network

A flow network is defined in graph theory as a weighted directed acyclic graph in which the weight of an edge represents the capacity of the edge in terms of the amount of some substance that can flow through the edge. The total weight of incoming edges equals the total weight of outgoing edges of a node. This implies that exactly the same amount of substance that enters a node can exit the node. A node cannot add or subtract substance from the network. There are two special nodes. The source node is a special node that creates the substance that flows through the network. Thus more substance exits the source node than enters into it. The target (or sink) node is a special node that receives the substance that flows through the network. Thus more substance enters the node than exits out of it. A flow network can be used to model the flow of water in a pipe network, the flow of traffic on a road network, etc.

4.3.3. Formal description of a flow network

A formal description of a flow network is presented in this section [SDSU1996].

A flow network G = (V, E) is a directed graph where each edge (u, v) has a capacity c(u, v) >= 0



Source (s) - node with only outflows

Sink (t) - node with only inflows

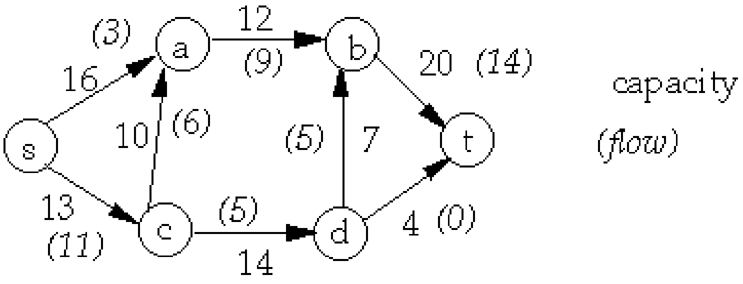
A flow in G is a real-valued function f : V * V -> R such that:

- For all u, v $\in V$, f(u, v) <= c(u, v) [Capacity constraint]
- For all u, v $\in V$, f(u, v) = -f(v, u) [Skew symmetry]
- For all u $\in V$ - {s, t}, $\sum_{v \in V} f(u,v) = 0$ [Flow conservation]

Value of a flow f is $|f| = \sum_{v \in V} f(s,v)$

Positive Net Flow entering vertex v is $\sum_{\substack{u \in V \\ f(u,v)>0}} f(u,v)$

Positive Net Flow leaving vertex v is $\sum_{\substack{u \in V \\ f(v,u)>0}} f(v,u)$



f(a, b) = 9 f(b, a) = -9 c(b, a) = 0 c(a, d) = 0 f(a, d) = 0

4.3.4. Maximum network flow

The maximum network flow problem is to maximize |f|, that is, to route as much flow as possible from s to t. The maximum value of the flow from vertex s to vertex t is equal to the minimum capacity of an s-t cut in the network, as stated in the max-flow min-cut theorem. The max-flow min-cut theorem states that in a flow network, the maximum amount of flow passing from the source to the sink is equal to the minimum capacity which when removed in a specific way from the network causes the situation that no flow can pass from the source to the sink. The Ford–Fulkerson algorithm and the Edmonds–Karp algorithm are two algorithms that can be used to compute the maximum network flow. The complexity of the Ford–Fulkerson algorithm and the Edmonds–Karp algorithm is $O(E\ max|\ f\ |)$ and $O(VE^2)$ respectively.

4.3.5. Trust graph

A trust graph is a weighted directed graph that models trust relationships between nodes. The trust relationship of a node u in a node v can be considered as an edge from node u to node v with the trust value as the weight of the edge. The trust graph is a subgraph of the social network of services.

4.3.6. Modeling a trust graph as a flow network for computing indirect trust

We model the trust graph of the social network of services as a flow network in order to compute the trust of a node *u* in a node *v* when no direct relationship exists from node *u* to node *v*. The declared trust of nodes in other nodes is considered as the capacity of the edges. We consider the node *u* as the source node and the node *v* as the target node of the flow network. We compute the maximum flow of the declared trust from the source node *u* to the target node *v*. This amount of maximum flow is considered as the indirect trust of node *u* in node *v*.

4.3.7. Advantage of computing indirect trust as maximum network flow

This approach of computing indirect trust as maximum network flow has a significant advantage. The amount of indirect trust of node *u* in node *v* cannot exceed the amount of trust that the node *u* has assigned to nodes with direct relationships. Thus it is not possible for a malicious node *v* to mount a sybil attack in which the malicious node *v* creates several fake nodes that assign fake trust to the malicious node *v* in order to raise its perceived trustworthiness. In the maximum network flow model, only the nodes that lie on a path between the source and the target node are considered. Moreover, the malicious and the fake nodes cannot inject any new trust into the flow network since they are not considered as the source nodes when computing *u*'s trust in *v*.

5. Conclusion and Future Work

We propose the following future work on the Social Filter:

1. Extending the social filter to support the peer-to-peer architecture. The current version of the social filter is centralized. The advantage of the centralized approach is that a single replica of the social network is maintained. This eliminates consistency issues that may arise in the deployment and processing of the social network due to decentralization. However, to conform with the vision of distributed and decentralized event clouds that interact in a peer-to-peer fashion, it is necessary to move the social filter to a peer-to-peer architecture. Decentralized versions of trust inference protocols will need to be developed.
2. Adding context to the declared and inferred trust. In the current version of the social filter, a service's trust in another service has a universal context. That is the trusting service trusts the trusted service in general. However, the trust can vary according to different contexts. It is also evident from the two use cases of the SocEDA project that services may trust each other in different contexts. It is thus proposed that the next iteration of the social filter will support multiple contexts of trust that reflect the contexts in the two use cases.

6. References

[Mika2011] P. Mika and A. Gangemi. Descriptions of social relations. Technical report, Department of Business Informatics, Free University Amsterdam, The Netherlands, Retrieved February 17, 2011 2011.

[SDSU1996] CS660 Combinatorial Algorithms – Network Flow. San Diego State University. November 25, 1996. http://www.eli.sdsu.edu/courses/fall96/cs660/notes/NetworkFlow/NetworkFlow.html

[Petroczi2007] A. Petroczi, T. Nepusz, and F. Bazso. Measuring tie-strength in virtual social networks. Connections, 27(2):39 - 52, 2007.

[Granovetter1973] M. Granovetter. The strength of weak ties. American Journal of Sociology, 78:1360-1380, May 1973.

[Maamar2011] Z. Maamar, P. Santos, L. Wives, Y. Badr, N. Faci, J.P.M. de Oliveira. Using Social Networks for Web Services Discovery. IEEE Internet Computing, July-Aug, 2011. Volume: 15, Issue:4, Pages: 48 - 54.

[Qinyi2009] Qinyi Wu, Arun Iyengar, Revathi Subramanian, Isabelle Rouvellou, Ignacio Silva-Lepe, Thomas Mikalsen. Combining Quality of Service and Social Information for Ranking Services. ICSOC 2009.