Fides, Trust Model for Highly Adversarial Global Peer-To-Peer Networks of Intrusion Prevention Systems

Bc. Lukáš Forst

2022

The name of the trust model - Fides. URL - Github. Named after Fides - goddess of trust and good faith.

So far this document is just set of random paragraphs and notes for the future.

1 Possible One Liners

- An algorithm for the peer-to-peer network of IPS agents that evaluates agents' behavior and data to assign them a trust value - essentially saying how much the local agent can trust another remote IPS.
- A trust model for the peer-to-peer network of IPS agents that evaluates agents' behavior.
- A trust model for the global peer-to-peer network of IPS agents in highly adversarial environment.
- Fides, Trust Model for Highly Adversarial Global Peer-To-Peer Networks of Intrusion Prevention Systems.
- Fides, Trust Model for Global Peer-To-Peer Networks of Intrusion Prevention Systems.

2 Problems we are solving

1. How do we determine who and how much to trust in global peer-to-peer network, where peers can join and quickly leave and there is a possibility that all connected peers are adversarial.
2. Given that peers can join and leave any time, we will have problem with cold start. We need to come up with a way how can newcomers safely gain initial trust fast.

3. The main (but not only) purpose of the project is to be able to share threat intelligence over the network. How do we aggregate said threat intelligence into a single aggregated network opinion?

4. As a Slips administrator, I want to be able to share data only with (or trust incoming data only from) specific peers or specific organizations.

2.1 Terminology

The problem: here I want to describe what terminology I'm using

- **Fides** - name of the trust model, performs all trust related computations
- **service trust** - how much a trust model trust a peer to provide us a good service - it is not necessarily how much we trust the data we received from the peer as to the final computation we may include the information about peer's IP address from the Slips (what does Slips think about the IP address)
- **target** - an IP address or domain - point of interest that can be source of network traffic and Slips has a capability to assign threat intelligence data to this object
- **remote peer** - peer that is running somewhere on the internet and it is connected to the global peer-to-peer network
- **local peer** - local instance of Slips that is connected to the global peer-to-peer network and runs instance of Fides

2.2 Interaction Evaluations

In order to determine what remote peers are providing valuable data and what peers are are not, the local peer needs to be able to evaluate each interaction it had with the remote peer. In general, there are two options how to approach this - either by designing evaluation that is protocol aware (meaning, that it understands the protocol and the data that are two peers sharing with each other), or by having an evaluation function that does not need to understand the protocol and can be used for any data.

We choose to implement both approaches and they are described in the following sections. In order to evaluate which strategy is better in what scenarios, we designed and run many simulations - their results are described in section 4.

We will use notation from table (1) when referring to peers and their interactions.

i	local peer
j	remote peer
T	target of network intelligence, domain or IP address
k	evaluation window
$s^k_{i,j}$	i's satisfaction value with interaction with peer j in window k
$S^k_{j,T}$	score computed by the peer j about target T in window k
$C^k_{j,T}$	confidence, how much is the score correct
S^k_T	aggregated score from all threat intelligence reports in window w for target T
C^k_T	aggregated confidence

Table 1: Interactions Symbols

2.2.1 Evaluate all interactions with the same value

The strategy, that does not need to understand underlying data, their semantics nor their structure. It is a naive approach when the trust model uses the same satisfaction value for all data it received. It does not check, if the data make sense (for example when all other peers but one are reporting that the IP address is malicious) and assigns all peers same satisfaction value $s^k_{i,j}$. The idea behind this algorithm is that when the peers are interacting for a longer time or have more interactions, they're more trustworthy.

This approach is for example used by the botnet **Sality** or by the **Dovecot** trust model. Fides implements it as *EvenTIEvaluation* strategy with configurable satisfaction value and administrator can use this strategy if they see it as the most optimal.

The disadvantage of this approach is, that we do not penalize remote peers when they provide wrong data, because the evaluation method does not care nor understand the underlying data. Because of that and in a case when the adversary gains the service trust of the model by following the protocol for longer time, it may significantly influence the aggregated score as the adversary has higher trust then other remote peers. If this happens, there is no way to automatically downgrade adversary's service trust.

2.2.2 Use aggregated network intelligence for evaluation

Because Fides is designed for the sharing and aggregating threat intelligence and understands the protocol that is being used, we can utilize this and penalize peers that are providing local peer with incorrect data. The interaction evaluation is performed at the end of the threat intelligence sharing process, at that point, Fides already aggregated data and decided what is the aggregated network score and confidence. Thus, we can utilize aggregated values and use it as a base line. Then we compare it against every each remote peer's threat intelligence we received. This evaluation strategy is implemented in the Fides a as a $DistanceBasedTIEvaluation$.

Suppose, that remote peer j provided data about target T to local peer i in window k. Provided data consist of score and confidence - $(S^k_{j,T}, C^k_{j,T})$. Where score, $-1 \leq S^k_{j,T} \leq 1$, indicates if the target is malicious (-1) or begin (1). The confidence $0 \leq C^k_{j,T} \leq 1$ on the other hand indicates, how much is the peer sure about its assessment of $S^k_{j,T}$.

In order to evaluate interaction between the local peer i and remote peer j we need to compute satisfaction value $s^k_{i,j}$. It holds that $0 \leq s^k_{i,j} \leq 1$ - where 1 means peer i was satisfied with the interaction.

$$s^k_{i,j} = \left(1 - \frac{|S^k_T - S^k_{j,T}|}{2} \cdot C^k_{j,T}\right) \cdot C^k_T \tag{1}$$

Where S^k_T is final score aggregated across the reports from the peers, C^k_T is aggregated confidence.

The problem in this evaluation algorithm are the situations when the aggregated confidence C^k_T is close to 0. In this case the algorithm will penalize all peers for providing any threat intelligence as the final $s^k_{i,j}$ is close to 0. Another issue with this approach is that when a single honest peer has a unique information about an IP address or domain, which is significantly different then what other peers have, it is automatically penalized for not sharing the same opinion as the other peers. However, if the peer is trusted enough, it has higher impact on the aggregated value and it is not penalized too much.

2.2.3 Use network intelligence only if the confidence is high enough

In order to compensate for the low confidence C^k_T and penalizing all peers in algorithm explained in 2.2.2, this evaluation strategy considers C^k_T value and employs $DistanceBasedTIEvaluation$ only when the C^k_T is *"high enough"*. In this case *"high enough"* means higher then configured value by the Slips administrator - CT. In a case when $C_T^k < CT$, the algorithm fall backs to using $EvenTIEvaluation$, because it is not possible to distinguish between *"good"* and *"bad"* network intelligence due to low confidence of the decision. This strategy is implemented in Fides under the name $ThresholdTIEvaluation$.

Algorithm 1 $ThresholdTIEvaluation$

```
1: $CT \leftarrow configuration$    ▷ configuration provided by the administrator
2: if $C_T^k < CT$ then
3:     $s_{i,j}^k \leftarrow EvenTIEvaluation()$
4: else
5:     $s_{i,j}^k \leftarrow DistanceBasedTIEvaluation()$
6: end if
```

What should be the correct value for CT from configuration is subject to evaluation in the simulations in section 4.

2.2.4 Use local threat intelligence to evaluate network intelligence

This approach uses similar equation for computing satisfaction value outlined in 2.2.2. However, the input is different - instead of comparing remote peer's (j) threat intelligence ($S_{j,T}^k$, $C_{j,T}^k$) to aggregated intelligence (S_T^k, C_T^k), we compare it to the threat intelligence of the local (i) Slips instance - ($S_{i,T}^k$, $C_{i,T}^k$). Thus the evaluation is following:

$$s_{i,j}^k = \left(1 - \frac{|S_{i,T}^k - S_{j,T}^k|}{2} \cdot C_{j,T}^k\right) \cdot C_{i,T}^k \tag{2}$$

This approach is useful when local peer has enough information about the target, but it wants to verify the behavior of the remote peers. To determine whether they are sending data that are somewhat correct. This strategy is implemented in Fides with name $LocalCompareTIEvaluation$.

2.2.5 Weight local opinion with aggregated one

Another implemented strategy combines 2.2.2 and 2.2.4 and mixes them using weight w, provided from the configuration. This is good approach when the the Slips or the network has a lot of data on the target. It evaluates interactions with what local instance thinks and what the network opinion is. What the correct mixture is is subject to simulations and configuration from the administrator.

$$\begin{gathered} s_{i,j}^k = w \cdot \left(1 - \frac{|S_{i,T}^k - S_{j,T}^k|}{2} \cdot C_{j,T}^k\right) \cdot C_{i,T}^k + \\ (1-w) \cdot \left(1 - \frac{|S_T^k - S_{j,T}^k|}{2} \cdot C_{j,T}^k\right) \cdot C_T^k \end{gathered} \tag{3}$$

In Fides implemented as the $WeightedDistanceToLocalTIEvaluation$.

2.2.6 Utilize all available data in evaluation

The goal of this strategy is to evaluate the received data with as much confidence as possible while having fully automatic process without the administrator's configuration. In order to do that, we combine all previous strategies into one, where we utilize all available information into a single $s^k_{i,j}$ value.

We introduce new variables here - p_0, p_1, p_2 - which are essentially weights of the particular strategies. These weights are based on the confidence, that the strategy has in its own decision. Note, that there is a hierarchy and order matters. In our case we decided to prefer decisions coming from strategy 2.2.2, then we add data from the 2.2.4 and if the final decision still does not have confidence of 1, we add static value configured by the administrator (noted as S_C). The last part - S_C - simulates static strategy described in 2.2.1 and is set by the Slips administrator.

$$\begin{aligned} p_0 &= C^k_T \\ p_1 &= \frac{1 - C^k_T + C^k_{i,T} - |1 - C^k_T - C^k_{i,T}|}{2} \\ p_1 &= min(1 - C^k_T, C^k_{i,T}) \\ p_2 &= 1 - p_0 - p_1 \end{aligned} \tag{4}$$

The weights p_0, p_1, p_2 in 4, are designed to gather as much confidence as possible. p_0 is confidence of the aggregated network data, essentially saying how much is the network sure about the given score. p_1 is the confidence coming from the local IPS and the p_2 is the remaining confidence to 1.

When we have the weights, we can compute final $s^k_{i,j}$ where we use strategies - $p_0 \cdot$ 2.2.2, $p_1 \cdot$ 2.2.4 and $p_2 \cdot$ 2.2.1.

$$\begin{aligned} s^k_{i,j} = \\ & p_0 \cdot \left[\left(1 - \frac{|S^k_T - S^k_{j,T}|}{2} \cdot C^k_{j,T} \right) \cdot C^k_T \right] + \\ & p_1 \cdot \left[\left(1 - \frac{|S^k_{i,T} - S^k_{j,T}|}{2} \cdot C^k_{j,T} \right) \cdot C^k_{i,T} \right] + \\ & p_2 \cdot S_C \end{aligned} \tag{5}$$

This strategy is implemented in Fides under a name $MaxConfidenceEvaluation$.

2.3 Network Intelligence Aggregation

Fides is a trust model designed for global peer-to-peer network that is created by instances of Slips. It is designed to support Slips in detecting malicious actors on the network and enables threat intelligence sharing between peers of Slips instances. Because Slips was designed to be as modular as possible, Fides is effectively running as a module which provides aggregated threat intelligence to Slips. In other words, Fides provides a view on what does the network think about some threat intelligence target. This means that Fides needs to be able to aggregate elements of threat intelligence from remote peers into a single value that is then presented to Slips - (S_T^k, C_T^k) - a network opinion on given target T in window k.

We need to be able to say, that some reports are better then others based on the service trust the local peer has in remote peer - $st_{i,j}^k$. Thus we need to weight every report based on this trust. Let $R_{i,T}^k$ denote a set of remote peers that provided threat intelligence ($S_{j,T}^k$, $C_{j,T}^k$) to a local peer i and ws_j^k weight of their report (computed by i as a normalized service trust that it has in the remote peer). Then we can compute aggregated score S_T^k like.

$$\begin{aligned} ws_j^k &= \frac{1}{\sum_{j \in R_{i,T}^k} st_{i,j}^k} \cdot st_{i,j}^k \\ S_T^k &= \sum_{j \in R_{i,T}^k} ws_j^k \cdot S_{j,T}^k \end{aligned} \tag{6}$$

We also need to compute aggregated confidence C_T^k that expresses how confident we are about the S_T^k that was computed in the previous step. Again, we use service trust to compute this for each remote peer and normalize it.

$$C_T^k = \frac{1}{|R_{i,T}^k|} \cdot \sum_{j \in R_T^k} st_{i,j}^k \cdot C_{j,T}^k \tag{7}$$

Aggregated score and confidence (S_T^k, C_T^k) is then sent to Slips. Depending on the selected interaction evaluation strategy (described in 2.2), these values can be further used to evaluate each interaction with the remote peers.

2.4 Cold Start Problem

Dynamic and global environment such as global peer-to-peer network, used by Fides, is open to anyone and any peer can freely join and leave. Because of that, the local peer will encounter many peers that were not seen before. As there was no previous encounter with the peer, the trust model does not have any information about their reliability nor how much can trust them. New peers need to be able to trust from the local peer in order to be useful part of the network. However, the local peer needs to be able to discover malicious actors that are trying to gain its trust to misuse it.

We call this *Cold Start Problem* - how does the new peer gain initial trust from the network. There are multiple ways how to approach this issue, we identified following potential solutions. Fides has reference implementation for all of the following approaches and combines them according to provided configuration with aim to achieve best results for cold starts with malicious peers and behavior in mind.

2.4.1 Static Initial Trust

In this approach, whenever trust model encounters new peer, it assigns static value as an initial trust. What the value is depends on the specific implementation of the trust model.

For example, in **Dovecot** trust model, every peer starts with the trust of 1 (highest possible) and various interactions can lower the trust in the peer to 0. In other words, the trust model considers new peers honest from the beginning and only during the time their reputation can be lowered when they perform incorrect interactions or are discovered as a malicious peers.

On the other hand, **Sality** botnet uses *goodcount* as a counter of interactions with any other peer, higher the *goodcount*, the higher trust the peer has for the local peer. The goodcount for each new peer starts with 0. Meaning, that the botnet does not trust fresh peers at all and they can gain trust only by following the protocol which depends on number of good interaction and time.

Static initial trust is easy to implement, but it somewhat requires assumptions about the network. If the network is considered *mostly being*, it might be safe to use initial trust of 1, however for highly adversarial networks using initial trust of 1 might be dangerous and it is better to use 0. On the other hand, using low initial trust and no mechanism how to gain more trust fast means, that the being peers that joined recently, don't affect the final decisions of the model even though they might have useful information about adversaries.

Static initial trust is supported by Fides as form of fallback, when no other cold start technique is used. Administrator provides configuration which contains initial reputation for each new peer.

2.4.2 Pre-Trusted Peers

In a case, when the peer-to-peer network protocol allows peers to prove their identity, or to prove membership of some group, the trust model can utilize this knowledge and assign higher or lower trust.

The network layer, designed for Slips and Fides, supports this[**nl**] and provides a cryptographically-secure way how to identify single peer in the network and its membership in an organization. This allows administrators to *pre-trust* peers or the whole organization - by assigning them either initial reputation or directly service trust.

The Fides configuration allows administrators to specify static initial reputation for the specific peer or for all peers from the specific organization. This means that whenever new peer joins the network and it is pre-trusted, it gains initial reputation specified in the configuration. During the time, it interacts with the local peer and provides threat intelligence, the data are evaluated and trust model decides how much it trusts them (assigns service trust) based on the reputation and the quality of the data. In detail is this process described in 2.2.

Another option, the administrator can use, is to enforce the service trust for the peer for the time being. This effectively means that trust model will not evaluate any data received from the pre-trusted peer and directly assigns them service trust from the configuration. This configuration for the Fides is called *enforceTrust*.

Option to pre-trust peers solves the cold start problem for specific peers and organizations, as they will start with/keep the high reputation. Which organization or which peer to trust or not is entirely on the administrator of the trust model. The inspiration for whom to trust provides, for example, Tor and their directory authorities[**torauth**]. However as the administrator needs to know the identity of the peers or organization, it does not solve the cold start problem globally for all peers.

Probably do not have it here, but it is great example, this refers to "super trusted we-know-you-and-have-had-many-beers-with-you Tor volunteers"

2.4.3 Recommendations

As the local peer might have multiple remote peers, that it *trusts enough*, it could be able to utilize this relationship and ask remote peers *what do they think about newcomer*. In other words, whenever local peer encounters peer that wasn't seen before, it can ask for recommendation on this peer from the local peer's most trusted remote peers.

The recommendation system introduces new attack vectors, that can be exploited by adversaries either by getting trust for the malicious peer or by lowering trust in honest peer that might have some threat intelligence about the malicious actor. These attacks are called *bad-mouthing* and *unfair praises* and we need to consider them and implement countermeasures.

Because of the possible attacks, the local peer should not solely rely on the network recommendations when computing final service trust for the fresh peer. In case, when the recommending peers are malicious, it might skew the decisions of the local peer for the time being. In order to solve this, when computing the final service trust for the remote peer, the local peer should take in account its own interaction with the peer as well as the received recommendations.

Moreover, the local peer should request recommendations only if it has *enough* trusted remote peers, otherwise it can expose itself to the *bad-mouthing* and *unfair praises* attacks more easily.

Fides employs recommendation systems based on SORT [**sort**] and combines it with the pre-trusted peers (2.4.2) as well as with the static initial trust (2.4.1) as a fallback when no other option is available due to constraints such as having not enough trusted peers. The algorithm used for the recommendation system is explained in detail in section 3.

3 Computational Model of Fides

This section describes how does Fides come up with a single most important value $st_{i,j}$ service trust. Service trust describes how much can local peer i trust remote peer j. Used algorithm, that computes $st_{i,j}$, is based on SORT[**sort**] with modifications to fit our use case - a global peer-to-peer network for sharing threat intelligence. The modifications are based on the interaction evaluation strategies proposed in 2.2 and algorithms to solve cold start problem described in 2.4.

3.1 Intuition

In the following pages, we describe the process top-down starting with the most important parts - service trust - and then breaking it down to bits. There are two main ideas behind the most of the equations.

The first one is that we want to robustly capture average behavior of the peers. In order to do that, we will be computing average behavior and standard deviations from said behavior and normalizing them.

Secondly, we will be comparing and weighting first hand experience with the remote experience. First hand experience is what happened between local and remote peer during time they interacted. This can be, for example, threat intelligence sharing, file sharing or the results of recommendation protocol. Remote experience is what happened between one remote peer and another remote peer. In other words, first hand experience for peer j are actions between j and z. When j shares information about these action with peer i, for i it is a remote experience.

i	local peer, instance of Fides
j	remote peer somewhere on the internet
$st_{i,j}$	service trust - how much i trusts j that it provides good service
$r_{i,j}$	i's reputation value about j
$rt_{i,j}$	i's recommendation trust about j
$sh_{i,j}$	size of i's service history with j
$s^k_{i,j}$	i's satisfaction value with interaction with peer j in window k
$w^k_{i,j}$	weight of i's interaction with j in k
$f^k_{i,j}$	fading effect of i's interaction with j in k

Table 2: Fides Computational Model Notation

The table 2 describes the most important notation we use in the following sections.

3.2 Service Trust

One of the major goals of the algorithm is to compute the service trust $st_{i,j}$. We do that by weighting local experience with peer's j service, with the reputation, j got when it connected to the i. The weight here is size of the service interaction history $sh_{i,j}$ to maximal history size sh_{max}.

$$st_{i,j} = \frac{sh_{i,j}}{sh_{max}} \cdot \left(cb_{i,j} - \frac{1}{2} ib_{i,j} \right) + \left(1 - \frac{sh_{i,j}}{sh_{max}} \right) \cdot r_{i,j} \tag{8}$$

The equation implies that the more interaction there was, between i and j, the bigger impact on $st_{i,j}$ it has. In other words, the more i and j interact the less i rely on the reputation that i computed from the values provided by the network, at the time when j was seen for the first time.

3.3 Local Experience for Service Trust

First part of the equation 8 contains *competence belief*, $cb_{i,j}$, and $ib_{i,j}$ - *integrity belief*. Both of the values are based solely on the interactions history that peer i experienced with the peer j.

3.3.1 Competence Belief

Competence belief represents how much did peer j satisfy local peer i with the past interactions. We measure it as an average of interactions from the past [**sort**].

$$\begin{aligned} cb_{i,j} &= \frac{1}{\beta_{cb}} \sum_{k=1}^{sh_{i,j}} s_{i,j}^k \cdot w_{i,j}^k \cdot f_{i,j}^k \\ \beta_{cb} &= \sum_{k=1}^{sh_{i,j}} s_{i,j}^k \cdot w_{i,j}^k \end{aligned} \tag{9}$$

It holds that $0 \leq cb_{i,j} \leq 1$ and where $s_{i,j}^k$ is the evaluation of the interaction in window k, $w_{i,j}^k$ is the weight of the interaction (how important it was) and $f_{i,j}^k$ is the fading effect of that interaction. We describe $s_{i,j}^k$, $w_{i,j}^k$ and $f_{i,j}^k$ in greater detail in 3.4. β_{cb} is the normalization coefficient that ensures that $cb_{i,j}$ stays within the interval of $0 \leq cb_{i,j} \leq 1$.

3.3.2 Integrity Belief

Integrity belief is a level of confidence in predictability of future interactions. $ib_{i,j}$ is then measured as deviation from the average behavior $cb_{i,j}$. Therefore, $ib_{i,j}$ is calculated as an approximation to the standard deviation of interaction parameters[**sort**].

$$
\begin{aligned}
ib_{i,j} &= \sqrt{\frac{1}{sh_{i,j}} \sum_{k=1}^{sh_{i,j}} \left(s_{i,j}^{k} \cdot w_{i,j}^{\mu} \cdot f_{i,j}^{\mu} - cb_{i,j}\right)^{2}} \\
f_{i,j}^{\mu} &= \frac{1}{sh_{i,j}} \sum_{k=1}^{sh_{i,j}} f_{i,j}^{k} \qquad (10) \\
w_{i,j}^{\mu} &= \frac{1}{sh_{i,j}} \sum_{k=1}^{sh_{i,j}} w_{i,j}^{k}
\end{aligned}
$$

It holds that $0 \leq ib_{i,j} \leq 1$. The more consistent behavior peer j has, the lower the $ib_{i,j}$ is. Consistency is highly desired property as the local peer then have more precise estimates about future behavior of the remote peer.

3.4 Evaluating an Interaction

Should we keep it here or move somewhere else?

In section 2.2, we described multiple strategies, how to compute $s_{i,j}^{k}$ - satisfaction value with kth interaction of the local peer i with peer j. However, not all interactions are same - some of the interactions are more important then the others. Moreover, because peers can change their behavior, most recent interactions should be more important the the interactions that happened long time ago.

3.4.1 Weight of the Interaction

Because each interaction is different and its importance is different, we have $w_{i,j}^{k}$ that measures the importance. Weight belongs to interval $0 \leq w_{i,j}^{k} \leq 1$ and Fides implements it as a discrete function of interaction type. For example, weight of interaction when remote peer shares the threat intelligence is higher, then when the peer sends threat intelligence request.

3.4.2 Fading Effect

Fading effect $f_{i,j}^{k}$ determines *"how much does algorithm forget"* as the algorithm prefers most recent interactions over past interactions and thus $f_{i,j}^{k}$ reduces weight of the past interactions. $f_{i,j}^{k}$ is a *non-increasing function* of interaction and time, or an index of said interaction in the history. This depends on the implementation - Fides implements it as a decreasing linear function $f_{i,j}^{k} = \frac{k}{sh_{i,j}}$. However, it can be implemented as a function of time as well. In that case, the function does not depend on k (as an index of a time window) bur rather on the time, when the interaction happened.

3.5 Reputation and Recommendations

In order to mitigate cold start problem outlined in section 2.4 and in cases when there are no or little interactions between i and j, the algorithm relies on $r_{i,j}$ - *reputation value.* $r_{i,j}$ is the second part of the service trust equation 8 that introduces *remote experience* to the service trust.

Reputation value is computed from the recommendations received from the remote peers. When the local peer i encounters remote peer j for the first time and it does not have any data about its trustworthiness, i can request recommendations on peer j from i's most trusted peers. We denote a set of remote peers, that provided the recommendations as T_i.

A single recommendation response from peer $z \in T_i$ about giving recommendation to peer i about peer j contains following data.

- $cb_{z,j}$, $ib_{z,j}$ - summary of z's interactions with j, competence belief and integrity belief
- $sh_{z,j}$ - service history size, number of the interactions between z and j - the more interactions they had, then the z's recommendation has more credibility
- $r_{z,j}$ - summary of recommendations that z received on j
- $\eta_{z,j}$ - number of peers that provided recommendations for j when j was new to z and their recommendation was used to compute $r_{z,j}$

When the local peer receives all recommendations, it computes reputation value $r_{i,j}$ as a weighted expected local experience ($ecb_{i,j}$, $eib_{i,j}$ - estimates about competence and integrity) from the remote peers with their remote experience ($er_{i,j}$ - estimate about reputation of said peer).

$$r_{i,j} = \frac{\lfloor \mu_{sh} \rfloor}{sh_{max}} \cdot \left(ecb_{i,j} - \frac{1}{2} eib_{i,j} \right) + \left(1 - \frac{\lfloor \mu_{sh} \rfloor}{sh_{max}} \right) \cdot er_{i,j} \tag{11}$$

Do we need to explain why did we added all these properties to a single recommendation? Original paper does that, but I'm not sure if we need to do that or not.

The weight, used in equation 11, is the average of history sizes in all recommendations to sh_{max}, maximal interactions history size. We calculate μ_{sh} as follows.

$$\mu_{sh} = \frac{1}{|T_i|} \sum_{z \in T_i} sh_{z,j} \tag{12}$$

Again, we are weighting local experience to remote experience. However, in this case it is local for remote peers that provided the recommendations.

3.6 Remote Local Experience

This name is weird

Similarly as when we compute service trust in 8, we need to get competence and integrity belief. However, while creating reputation value in 11 where the values are coming from the remote peers, we are trying to estimate those values received from the network. For that reason, we call them *expected competence belief* - $ecb_{i,j}$ and *expected integrity belief* - $eib_{i,j}$.

3.6.1 Expected Competence Belief

$ecb_{i,j}$ is estimation about competence belief made by i about j. This value is computed from the received recommendations in combination with $rt_{i,z}$ - a *recommendation trust* that i has about z. Similarly as for service trust, we have a normalization coefficient β_{ecb} that moves the resulting data to correct interval. It holds that $0 \leq ecb_{i,j} \leq 1$.

$$\begin{aligned} ecb_{i,j} &= \frac{1}{\beta_{ecb}} \sum_{z \in T_i} (rt_{i,z} \cdot sh_{z,j} \cdot cb_{z,j}) \\ \beta_{ecb} &= \sum_{z \in T_i} (rt_{i,z} \cdot sh_{z,j}) \end{aligned} \tag{13}$$

Recommendation trust $rt_{i,z}$ is described in detail in section 3.7.

3.6.2 Expected Integrity Belief

Following the $ecb_{i,j}$, $eib_{i,j}$ is estimation about the integrity belief made by i about j. The equation is almost similar, but we use $ib_{z,j}$ instead of $eb_{z,j}$. This means that normalization coefficient $\beta_{eib} = \beta_{ecb}$.

$$\begin{aligned} eib_{i,j} &= \frac{1}{\beta_{eib}} \sum_{z \in T_i} (rt_{i,z} \cdot sh_{z,j} \cdot ib_{z,j}) \\ \beta_{eib} &= \sum_{z \in T_i} (rt_{i,z} \cdot sh_{z,j}) \end{aligned} \tag{14}$$

3.7 Recommendation Trust Metric

Recommendation trust - $rt_{i,z}$ - is another metric that a peers calculate and store. It expresses how much does i trust z, that it provides *good recommendations*. Even though one could theoretically use service trust $st_{i,z}$ for this, we have another trust metric, because there are peers that can provide very good data (service), but they have very bad taste on other peers, or other way around. This also gives us the ability to have specialized nodes in the network, that serve as a peers registry for organizations - a single node that only provides recommendations on peers.

We calculate recommendation trust in a similar way as the service trust and reputation, but we use recommendation competence belief $rcb_{i,z}$, recommendation integrity belief $rib_{i,z}$ and reputation $r_{i,z}$. This time, used weight is $rh_{i,z}$, which is size of the recommendations history provided by z to i, and rh_{max}, a maximal size of said history.

$$rt_{i,z} = \frac{rh_{i,z}}{rh_{max}} \left(rcb_{i,z} - \frac{1}{2} rib_{i,z} \right) + \left(1 - \frac{rh_{i,z}}{rh_{max}} \right) r_{i,z} \tag{15}$$

3.7.1 Recommendation Competence and Integrity Belief

Similar for interactions, we use three different parameters for calculating the $rcb_{i,z}$ and $rib_{i,z}$. We use satisfaction $rs_{i,z}^x$, weight $rw_{i,z}^x$ and the fading effect $rf_{i,z}^x$. The parameters have the same background as described in section 3.4, but in this case they are connected to recommendations instead of service. We calculate $rcb_{i,z}$ as follows:

$$\begin{aligned} rcb_{i,z} &= \frac{1}{\beta_{rcb}} \sum_{x=1}^{rh_{i,z}} \left(rs_{i,z}^x \cdot rw_{i,z}^x \cdot rf_{i,z}^x \right) \\ \beta_{rcb} &= \sum_{x=1}^{rh_{i,z}} \left(rw_{i,z}^x \cdot rf_{i,z}^x \right) \end{aligned} \tag{16}$$

And for recommendation integrity we compute $rib_{i,z}$ as:

$$rib_{i,z} = \sqrt{\frac{1}{rh_{i,z}} \sum_{x=1}^{rh_{i,z}} \left(rs_{i,z}^x \cdot rw_{i,z}^\mu \cdot rf_{i,z}^\mu - rcb_{i,z} \right)^2} \tag{17}$$

One more time, the computational model is trying to approximate average behavior in recommendations - $rcb_{i,z}$ - and its standard deviation from such behavior - $rib_{i,z}$.

3.7.2 Evaluating Received Recommendation

As outlined in section 3.7.1, in order to evaluate particular recommendation from remote peer z, we have satisfaction, weight and fading effect. We calculate recommendation satisfaction $rs_{i,z}^x$ by comparing values from z's recommendation $r_{z,j}$, $cb_{z,j}$, $ib_{z,j}$, with values that are the results of the the recommendation algorithm. In other words, we compare each recommendation, with the aggregated values - $er_{i,j}$, $ecb_{i,j}$ and $eib_{i,j}$. This gives us estimate how off, was the peer z's recommendation from the final result of the recommendation algorithm.

$$rs_{i,z}^x = \frac{1}{3} \left[\left(1 - \frac{|r_{z,j} - er_{i,j}|}{er_{i,j}} \right) + \left(1 - \frac{|cb_{z,j} - ecb_{i,j}|}{ecb_{i,j}} \right) + \left(1 - \frac{|ib_{z,j} - eib_{i,j}|}{eib_{i,j}} \right) \right] \tag{18}$$

This seems to be too wide, make it smaller

Then we calculate weight of the recommendation as follows.

$$rw_{i,z}^x = \tag{19}$$

4 Simulations and Evaluation

this is section where we describe how we run the simulations and what are the results of the evaluations

These are some equations we will need in order to describe the SORT algorithm.

$$f_{ij}^{\mu} = \frac{1}{sh_{ij}} \sum_{k=1}^{sh_{ij}} f_{ij}^{k} = \frac{sh_{ij}+1}{2sh_{ij}} \approx \frac{1}{2}.$$

$$er_{ij} = \frac{1}{\beta_{er}} \sum_{p_k \in T_i} \left(rt_{ik} \cdot \eta_{kj} \cdot r_{kj} \right).$$

$$r_{ij} = \frac{\lfloor \mu_{sh} \rfloor}{sh_{\max}} \left(ecb_{ij} - eib_{ij}/2 \right) + \left(1 - \frac{\lfloor \mu_{sh} \rfloor}{sh_{\max}} \right) er_{ij}$$

$$rs_{ik}^{z} = \left(\left(1 - \frac{|r_{kj} - er_{ij}|}{er_{ij}} \right) + \left(1 - \frac{|cb_{kj} - ecb_{ij}|}{ecb_{ij}} \right) + \left(1 - \frac{|ib_{kj} - eib_{ij}|}{eib_{ij}} \right) \right) /3$$

$$rw_{ik}^{z} = \frac{\lfloor \mu_{sh} \rfloor}{sh_{\max}} \frac{sh_{kj}}{sh_{\max}} + \left(1 - \frac{\lfloor \mu_{sh} \rfloor}{sh_{\max}} \right) \frac{\eta_{kj}}{\eta_{\max}}.$$

$$rcb_{ik} = \frac{1}{\beta_{rcb}} \sum_{z=1}^{rh_{ik}} \left(rs_{ik}^{z} \cdot rw_{ik}^{z} \cdot rf_{ik}^{z} \right), \tag{20}$$

$$rib_{ik} = \sqrt{\frac{1}{rh_{ik}} \sum_{z=1}^{rh_{ik}} \left(rs_{ik}^{z} \cdot rw_{ik}^{\mu} \cdot rf_{ik}^{\mu} - rcb_{ik} \right)^{2}}, \tag{21}$$

$$rt_{ik} = \frac{rh_{ik}}{rh_{\max}} \left(rcb_{ik} - rib_{ik}/2 \right) + \frac{rh_{\max} - rh_{ik}}{rh_{\max}} r_{ik}$$