

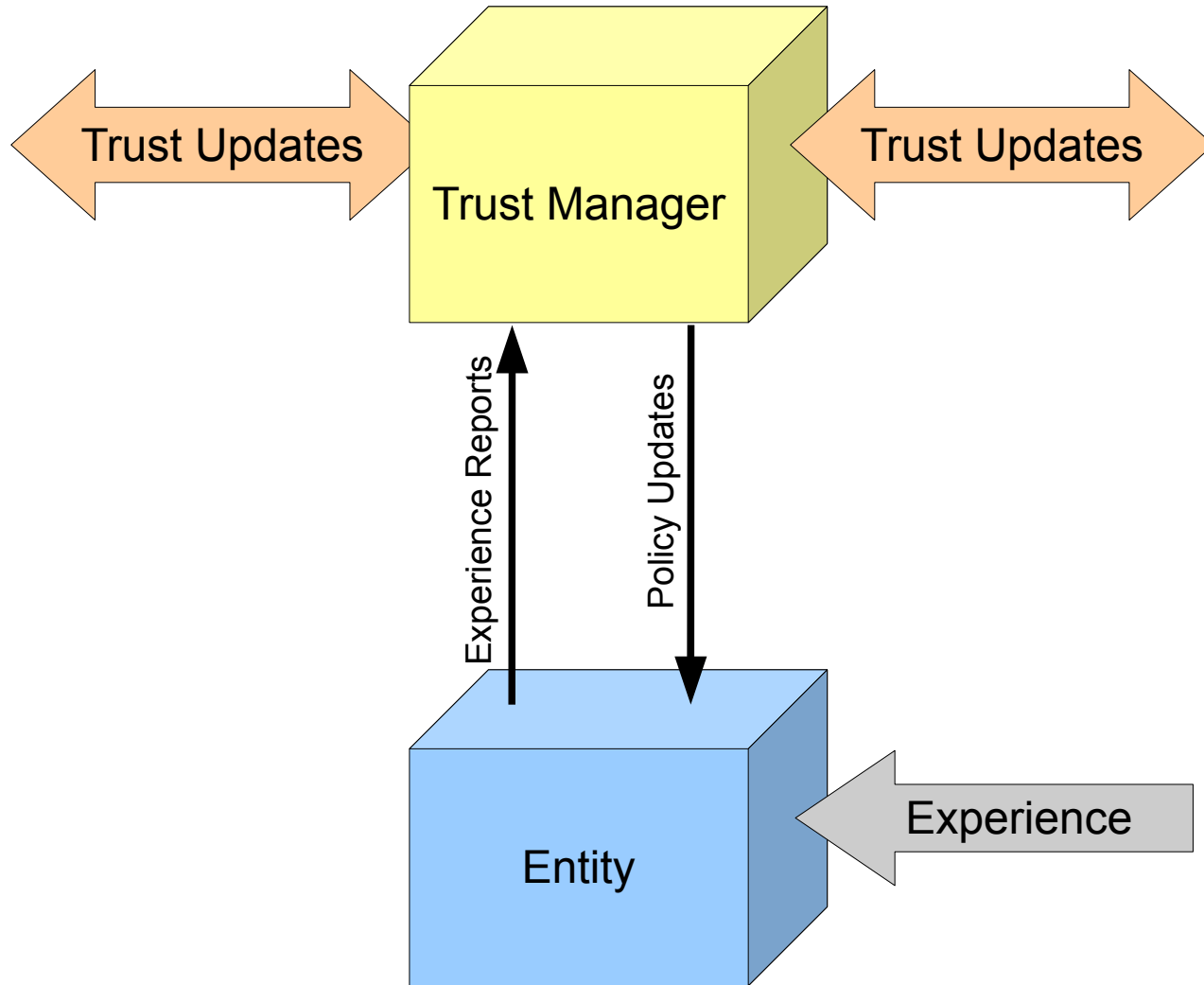
Implementing a Trust Overlay Framework for Digital Ecosystems

Paul Malone, Jimmy McGibney, Dmitri Botvich,
Mark McLaughlin,
Waterford Institute of Technology

Representing Trust

- Trust is an overloaded term
- We model trust mainly along the lines of Gambetta (1988)
 - Trust as a subjective probabilistic measure
 - Used in decision making
- But more than just a single number
- We also consider
 - Confidence – relating to reliability of trust assessment (e.g. How many experiences or measure were made)
 - Recency – when was it last updated

Distributed Trust Model



Our Approach

- Trust Overlay Network
 - Each participant has an associated Trust Manager
 - To report direct experience
 - To query for policy updates
 - Trust Managers form an overlay network
 - To share Trust Updates
- Distributed Trust Management
 - Participants make autonomous trust determinations about other participants based on
 - Direct Experience
 - Information from Third Parties

Our Approach

- Configuring Trust Managers
 - Each participant configures its Trust Manager via an XML configuration document
 - Configured on a contextual and subjective basis
 - Specific trust algorithms are associated with a context
 - One participant's algorithm for a context might not be the same as a different participant's
 - Other Trust Managers cannot determine which algorithm was used to assess the trustworthiness (reducing the ability to game the system)
 - Participants can easily write their own algorithms and provide these to their Trust Manager

Exponential Average Direct Experience Algorithm

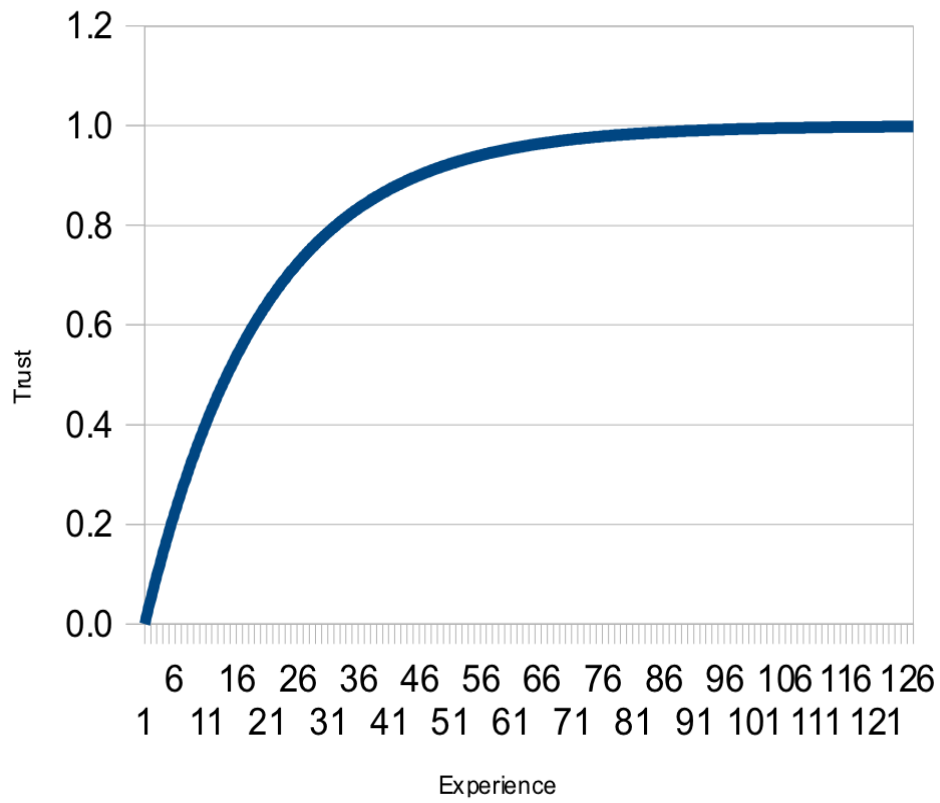
$$T_{i,j(n)} = \alpha E + (1 - \alpha)T_{i,j(n-1)}$$

Where:

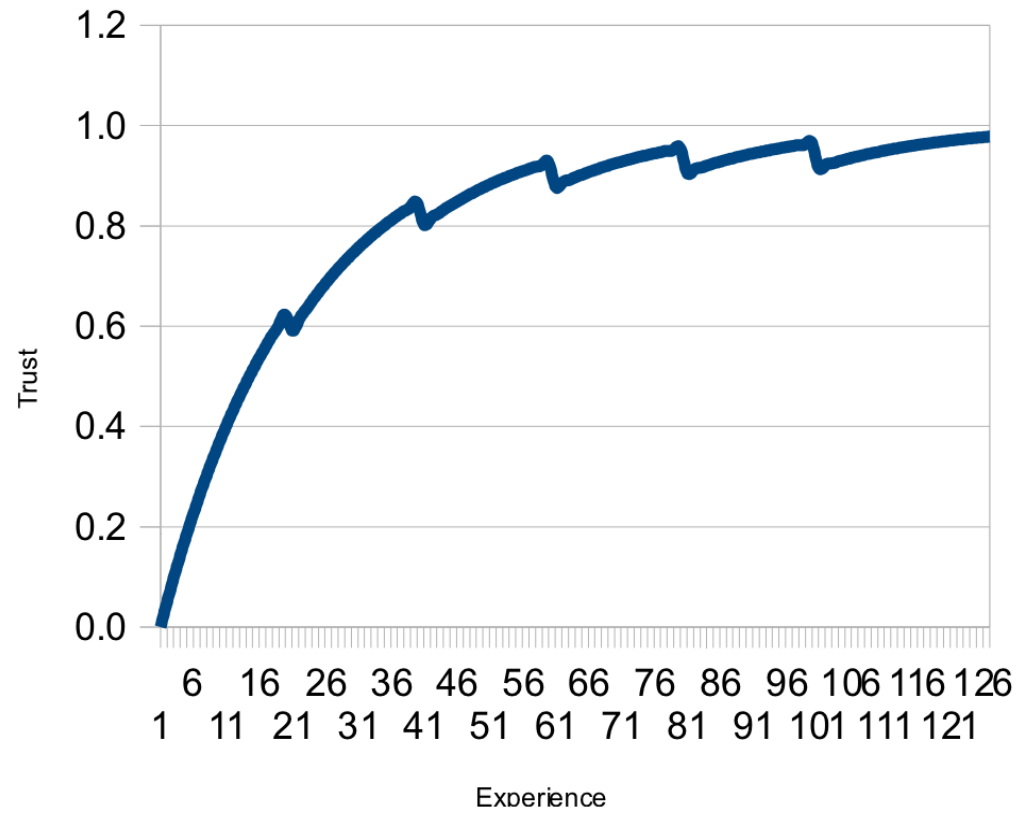
- $T_{i,j(n)}$ is the n th value of trust placed by i in j
- E is the latest direct experience report ($0 \leq E \leq 1$)
- α is the rate of adoption of trust ($0 \leq \alpha \leq 1$)

Exponential Average Direct Experience Algorithm

100% Positive Experience



95% Positive Experience



Exponential Average Referral Based Algorithm

$$T_{i,j(n)} = \beta T_{i,k} T_{k,j} + (1 - \beta T_{i,k}) T_{i,j(n-1)}$$

Where:

$T_{i,j(n)}$ is the n th value of trust placed by i in j

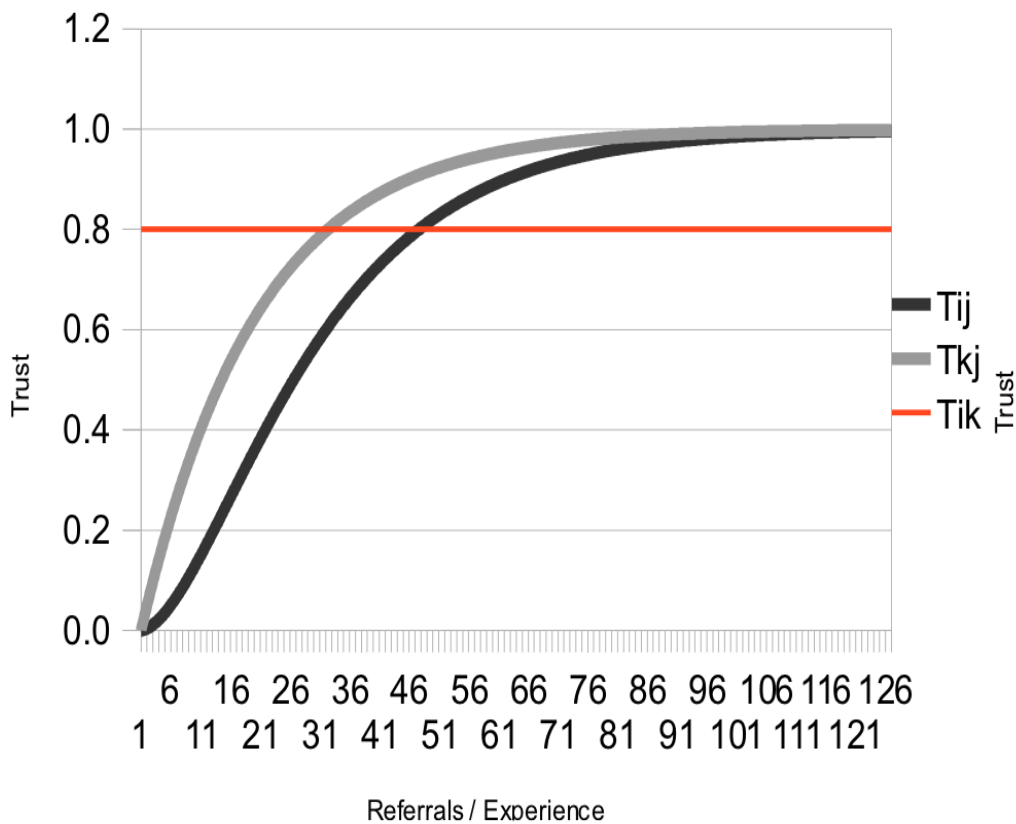
$T_{i,k}$ is the trust i has in k 's referral

$T_{k,j}$ is the trust k has in j

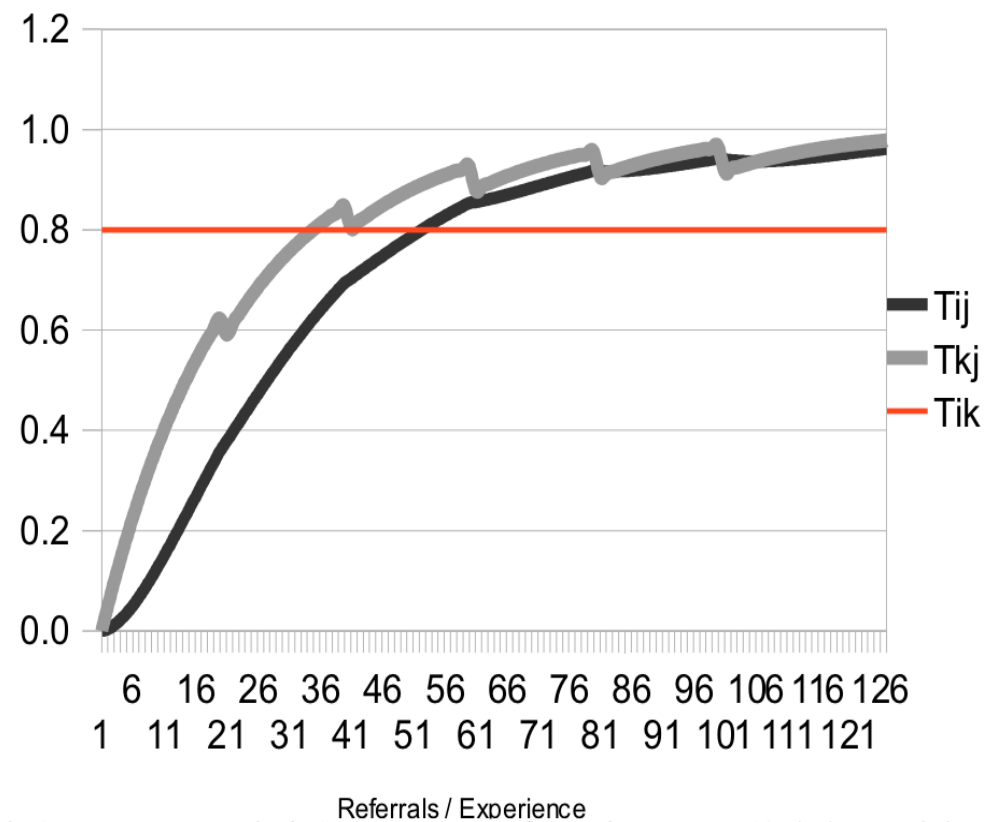
β is the influence that referrals have on local trust ($0 \leq \beta \leq 1$)

Exponential Average Referral Based Algorithm

100% Positive Experience, $\beta = 0.1$



95% Positive Experience, $\beta = 0.1$



Implementation – Trust Manager

- Maintains a set of algorithms for connected Trustor entities for specific contexts. Each context can have distinct algorithms for determining trust based on direct experience, referrals and reputation.
- Receives ExperienceReports from the Trustor entities and generates updated trust values according to the appropriate algorithms.

Implementation - TrustManager

- Provides updated TrustValues to other TrustManagers which can be used as inputs to referral or reputation algorithms.
- Provides PolicyUpdates to the Trustor entities based on current TrustValues derived from direct experience and referrals and reputation.

Implementation – TrustValue

- A TrustValue represents a single instance of a trust value a relying party has in a trusted party in a particular context.
- Each TrustValue instance is the output of a trust algorithm.
- Each TrustValue has a type associated with it.
 - Direct Experience
 - Referral
 - Reputation
 - Hybrid

Implementation - ExperienceReport

- An ExperienceReport represents a unit of experience from an end user (trustor) pertaining to an experience with a trustee in a context.
- The ExperienceReport forms the basis of all trust calculations as it is required to calculate any direct experience TrustValues.

Implementation – TrustAlgorithm

- Takes some known information as input and derives a new trust value according to some predefined logic.
- An interface TrustAlgorithm is defined:
 - `setTrustValues(String, float)`
 - allows the user to input hard values for some trusted parties. For example, a service consumer might decide that it would like to overload the algorithm with some hardcoded values for some of the service providers it interacts with.
 - `setParameters(String, String)`
 - used to load parameters in the algorithm.

Implementation - Configuration

- Each TrustManager manages a set of trust algorithms for various contexts.
- It is useful to devise a means of configuring the TrustManager to achieve this on a run time basis.
- To aid this a TrustAlgorithmConfiguration XML schema is designed.
- Trust Contexts are configured separately and each Context can define TrustAlgorithms for direct experience, referrals and reputation.

Implementation – Open Source

- The overlay network is available via source forge project *TrustFlow*

<http://trustflow.sourceforge.net/>

Conclusion

- Defining Trust
- Trust Model
- Algorithms
- Implementation

Questions

...?