Trusting Virtual Tags

Michel Deriaz

Abstract. Spatial messaging, or the fact of publishing virtual tags, is clearly not a new concept. In the GPS world, these tags are materialized through POIs (Point Of Interest) and they consist more or less in geo-referenced information. They are often classified in different categories. A typical use is to show on a map all the neighboring POIs of a certain category. But the only solution to trust the information is to get them from a reliable source. This paper presents a first approach to a generic way of presenting these virtual tags, and how to add trust information to them. Every user is therefore able to create virtual tags and the trust engine is responsible to return to a specific user only the ones that he is interested in.

1 Introduction

It exists different places where you can share your POIs with other people, like POIPlace [1] or the Google Earth Community [2], but you can easily convince yourself that this information can not be trusted. When you search a specific feature, like the "Jet d'eau" of Geneva, you will find it located in several different places. POIs suffer from the following drawbacks:

- They can be trusted only if they come from a known and reliable source. Everybody can, intentionally or not, publish wrong or misleading information. To scope with that, we need to know and trust the provider. There are people paid by companies (like Tele-Atlas) just to drive and record points of interests. This information is usually sold and not always up-to-date.
- They can not be moved. Since they are related to a current geographical position, POIs can not be moved easily. For instance, we can not attach a POI on a person or on a moving object.
- No deadline. The time component is not part of a POI. A reader has therefore no idea if the POI is still current or if it describes a feature that disappeared a long time ago.
- No information about the precision of the position. A position is usually expressed in latitude and longitude, typically with 6 digits after the decimal point. This guaranties a precision of less the 0.12 meters worldwide. However, the devices that are used to record these positions are probably less precise. Typically, a GPS device has a precision between 5 and 50 meters. And this difference can be critical in some situations.
- No possibility to comment them. A POI is static information and there is no way to comment one, or to ask to remove it because it is out of date.
- Search possibilities are limited to position and category.

Spatial messaging goes a step further. A virtual tag can be attached to an object and move with him, can contain any type of information (sounds, video...), can be reviewed and commented, is aware of the time component, and can be trusted. Everybody can publish and make reviews, which favor lots and up-to-date tags, and the system manages the trustworthiness of these tags, which should give us the same quality than provided by commercial solutions.

This paper is divided on two main parts. The first describes what a virtual tag is and defines its generic structure. Several examples illustrate how our contribution covers very different application domains. The second part presents a short state of the art of trust mechanisms related to our work and explains why new models are required. The reader will be convinced that contextual information, which is hardly ever taken into account in traditional trust engines, is capital in the domain of virtual tags.

2 A vTag

We define a vTag as a virtual tag structured in a specific way. Some of the fields are obligatory in order to be treated by a trust engine, and some are not and just given as additional information. Basically a tag is a geo-referenced and dated piece of information with a list of ratings and reviews.

2.1 The ID field

This field, which is an URL, identifies uniquely a tag. Using URLs allows us to see the tags in a web-browser (useful if someone does not have a specific application), let us to bookmark them, to send the URL to someone else, and we are free to use any type of content in the tag. Another advantage is that an URL can be easily coded in different ways. It can for instance simply be written on an object, and people that want to see the tag attached to that object simply browse this URL. Or the URL can be transmitted by a RFID tag, so that the client application gets the tag when the user approaches the tagged object. In a similar way we can also use Bluetooth or wireless hot spots.

Another interesting way is to use visual tags, like described in Mobile Tag [3] (commercial) or in the MUSCLE European project [4] to signal the presence of a virtual tag. A visual tag is a small squared picture encoding an URL. This technology has the advantage of being very simply to use (just point the camera of your mobile phone to the visual tag), intriguing (people want to know what is hidden behind this strange image), and mobile (visual tags can be painted everywhere).

There are and they will be more different possibilities to present an URL. Using them to identify our vTags is a guaranty that future technological improvements will be easily integrated in our model.

It must be clear that we do not need necessarily to know the URL of a tag in order to get it. To get all the tags around us, we simply need to give our current

position. URLs just give us another way to access it, useful when the position is unknown.

2.2 The author field

Each tag is assigned to a given author. This allows a trust engine to compute the reputation of it. An author is represented by a pseudonym so that its real-life identity remains hidden. An application can choose whether it allows anonymous postings as well. Even if this allows more privacy, anonymous posting can not be treated by a trust engine and thus prevents information about the reliability of the tags.

In certain cases, to avoid a Sybil attack [5] an application can require that a single user owns only a single pseudonym (or at least a limited number of them). This can be achieved by different means. For instance we can use a reverse-charge SMS service like the one provided by [6], so that a single user cannot have more pseudonyms than mobile phones. Or, if the user needs to stay anonymous even from the server, he can use blind signatures and a ZKP (Zero Knowledge Proof) algorithm as described in [7] in order to stay completely anonymous and still be able to change, still in an anonymous way, its pseudonym.

2.3 The position field

The position field describes where the tag is. Unlike POIs, a vTag is not limited to a single point. A vTag can also cover an area. The coordinates are expressed in the WGS84 standard (latitudes/longitudes/altitude), so that the system works worldwide. Taking the altitude into account allow us to draw virtual boxes in the air. It is therefore possible to show one tag to the people that are at the top of a tour, and a different tag to the people that are at ground level. An area can be very small: "here you are in my garden" or very big: "high risk of malaria in this region". Like in conventional GIS applications it is possible to draw the tags on a map, for instance to study the progression of a disease. And this with the obvious advantage of being almost real-time thanks to the high interactivity provided by the system.

The position can be given by any technical mean, including GPS or GSM Cell ID. The precision can therefore vary from centimeters (differential GPS) to kilometers (GSM Cell ID). To scope with that, the application is responsible to send also the precision used to make the measure. This precision can also be given manually by the user. For instance, let us consider a user that wants to tag an inaccessible object that is about, according to his estimation, 200 meters in front of him. This precision parameter can also be used in the future by technical means that compute distances. In the former example, we can imagine that the phone contains a built-in radar that computes the distance to the sighted object.

Tags can be mobile. We can imagine users that carry a tag with their profile in order to find neighboring users that share the same interests. Or a taxi company that wants to know in real time where the drivers are in order to send the closest to a client. Theses tags are clearly meant to move and we expect that they send their

new coordinates time to time. But we can also have a tag that was initially designed to be fixed, but that need to be placed somewhere else (for instance the tagged object has moved). In booth cases, the "technical" operation consists in changing the coordinates of the tag. But then a history of the previous positions as well as the corresponding times must be kept. The reason is that the content or one of the reviews can be true only if the tag is at a specific position. It is the application that decides whether a tag can be moved, by whom (author, reviewers...), and what rights are given to the users (rating...) in each situation.

For some tags, it is not possible to give a position related to the Earth, expressed in latitude and longitude. In this case the tag can also be "attached" to a specific object (probably moving), so the position of the tag is simply the position of the object. The attachment is therefore done by a contextual description instead of a latitude/longitude coordinate. For instance, visual tags are paint on the boats participating to a regatta, and the spectators get information about the navigators by scanning the visual tags through the camera of their mobile phones. In this case, the vTag is attached to the boat, wherever it is.

2.4 The creation time field

In order to know the "freshness" of a tag, each new tag contains the date and time of its creation. These are written in the ISO 8601 format (to be understood worldwide) and the time stamp is set by the server (UTC). The choice whether it is the server or the client that put the timestamp is important in cases where the delay between the writing of the tag and the reception of it by the server becomes significant. This can happen in a region where there is no network. A user records tags and they are stored on his device until a network connection is available. According to where the user is, a delay can be expressed in milliseconds as well as in days, weeks or months!

It is not a good idea to let the user to timestamp the tags. First, we can not be sure that he is sending the right date. And second, since there can be various delays, we can have old tags that appear before new ones. This complicates significantly the handling of data by the trust engine, and can also create confusion by the users. For instance, a tag that should exist since a long time but which appears just now. These are the main reasons why we prefer to let the server to timestamp the tags. In this way, the creation time indicates in fact the time the tag has been made available.

However, the time the user makes a given observation can be important (especially if the delays are long). In this case the user sends also the time of the writing (in UTC), which is given as additional information (not taken into account by the system). It is therefore the role of the application to decide how to deal with it. For instance, an application could show a warning if the time difference is above a certain level. This is clearly highly related to the application, and can not be taken into account in a generic system.

2.5 The deadline field

After the deadline, the tag is removed. Some tags report an observation that is true only for a certain amount of time, and sometime this amount of time can be guessed by the author. For instance a mountain guide reporting a high risk of avalanches can clearly put a deadline to its tag. The first advantage is to avoid spamming other users with outdated tags. And the second advantage is to help the trust engine to evaluate how to adapt a trust value after a rating. If you disagree with the content of a tag but you are close to the deadline, you will not decrease to much the trust value of the author since this tag is perhaps simply outdated.

Unlike the creation time field, the deadline field must be completed by the user (ISO 8601 format, UTC time). We can not just say how long a tag must be visible, and let the server to compute the deadline. The reason is that we have no idea about how much time the tag will need to arrive to the server (we saw that it can be milliseconds or months), so the deadline information becomes useless. The tag will be visible only if it arrives on time. For instance if we send a tag today and put the deadline in 10 days, then it will be visible only if the server gets it before 10 days.

2.6 The request to delete time field

To avoid malevolent acts, it is not possible for a user to directly remove a tag. Instead, when certain conditions are met (for instance several users that rated the tag negatively), a "request to delete" is made to the tag. Its value is the time the request is made, and external rules define when the tag should be definitively removed. It is the time of the server that is used.

2.7 The content field

The content of a vTag is coded in XHTML. This language is sufficiently flexible and extendable to add specific fields for a given application, and sufficiently generic to be viewed by any web browser (for people that do not have an application for managing vTags). Doing so, we allow our vTags to contain the same type of content that we find on the Internet (text, pictures, videos...) and give them the possibility to link them though hyperlinks to other tags or to any web page.

2.8 The reviewers field

A user can agree or disagree with the content of a tag. A tag contains a reviewers list that is sorted in an inverse chronological order. Each review contains the current time, the ID of the reviewer, the rating, and possibly some content (same format than the content written by the author). The time of a review is the server time. The reviews must be independent from each other, and always comment the original content. It is not possible to review another review, so this must not be confused with a blog. In the same way, a single user can review only once a given tag. If he reviews a second time, the old review is erased and the new review is put at the top of the list.

2.9 Visibility of a tag

A vTag remains basically information related to a current position. But we saw that this position is not always available, and that it is also possible to access to a given tag by its URL. In fact, it is the application that decides how the tag must be presented to the user. A simple solution consists in showing permanently all the tags around you. But in case where the positions represent areas, we can also decide that a tag is visible while entering the area, while inside the area, or while exiting the area. We can also play with the notion of time, and decide when a tag is visible and when not. Or even leave another external trigger to decide if the tag must be currently visible or not. It is not a good idea to leave the system to decide how to show the tags to the user, since it is very application specific and can not be made generic easily. And anyway, our model allows easily adding such specific requirements (meta-data).

2.10 Interoperability of the tags

To guaranty a quick and efficient deployment and acceptance of the virtual tags among the users, tags must be visible even by people that are not equipped with specific software or material. XHTML is a good candidate. Every request to a server is done by a HTTP request, and every response is returned as an XHTML file. It means that any single web browser on a desktop computer or on a mobile device is sufficient to deal with virtual tags, i.e. reading and rating them. The only difference with a "standard" XHTML file is that vTags are structured in a particular way and contain some additional meta-data, like the position. It is clear that in most cases the web browser is not the best solution to deal with tags. An application can be aware of its position and show automatically all neighboring tags, present them on a map, launch an alarm if certain conditions are met, or propose an intuitive and easy way to rate tags. The fact that we can see tags through a web browser allows potential users to discover easily new services, and then decide if they want to install a specific application for this service.

2.11 Revoking and deleting a tag

To avoid malevolent acts, the user interface does not allow deleting directly a tag. It is the trust engine that decides when it is safe to remove a tag from the system, and this issue is discussed in the trust engine part. The important point here is to see that nobody, not even the author, is able to delete a tag. The user interface allows only creating a new tag and rating it. Of course, the trust engine can be configured so that a single negative rating (even done by the author) deletes the tag, but then it is done under the responsibility of the application. A specific right that is given to the author of the tag is to revoke it. Again, revoking will not delete it. It will just mark this tag as revoked, meaning that the author does not agree anymore with the content of the tag.

The fact that an author cans only revoke (and not delete) a tag is very important. Otherwise a user could easily cheat the trust engine, for instance by posting a tag, increasing its trust value thanks to the reviewers, deleting the tag and recreating the

same tag in order to benefit again from good rating from other users. The details are discussed in the trust engine part of this document.

2.12 Simultaneous updating

This can happen when two users are rating a tag simultaneously. For the system this is not a problem since the time of the rating is the time given by the server when it gets the information. In other words, there is no simultaneous updating for the server. At the client side, it is not a problem as well. Remember that a single user can only have one review (a second rating erases the first one), and that a review is only about the original text (we can not review or rate a former review). Since the reviews are therefore completely independent from each others, the order they arrive is irrelevant.

2.13 Shadow areas

We call shadow area a zone where the user is unable to know its position, like in an urban canyon or a tunnel if he uses a GPS. There is no generic and efficient way to handle this issue. Especially since it is actually only a technical problem. In the future, GPS devices will improve and other positioning systems will appear. And a shadow area is not necessarily one for another user, equipped with a more sophisticated positioning device. It is therefore the application that decides how to handle this problem. For instance, visitors of a national park get vTags thanks to the position given by their GPS while outdoor, and thanks to visual tags while inside grottos.

2.14 Searching for tags

There are several ways to search specific tags. The first consists in using contextual data, like the position or the time the tag has been created. For instance, asking all the tags around my current position that have been created during the last 24 hours. Handling contextual information for searches has been studied in the MobiLife European project [8] and can be reused in this work. The second solution consists in using keywords. The author of a tag can fill a keywords field, and searches will be done using these keywords, like it is done in Flickr [9], a service that allows to geocode and share photos with others. Note that Flickr uses the term "tag" to define what we call "keyword", which can be a bit confusing. A third solution consists in using the title of the tag. For instance, if user Alice drew a virtual path using tags and called them (title) "waypoint 1", "waypoint 2" and so on, an interested user can therefore ask his system to show all the tags authored by Alice where the title starts with "waypoint". These are only basic examples of searches. We can also imagine that searches are done in the all tag through regular expressions [10], which would allow to find, for instance "all the tags around me, authored or reviewed by Alice during the last 12 hours, that does not contain 'waypoint' in the title, with either 'cat' or 'dog' in the keywords".

2.15 Tracking

There are two ways of doing tracking. The first has already been discussed previously and consisted in modifying the position of the tag. This can be used if we are interested to know where a person or an object is, at current time or a few times ago. The second way consists in sending a new tag each given time interval, and so draw a virtual path. For instance the GeoSkating [11] application built over the GeoTracing [12] generic framework allows skaters to draw maps, where the roads are colored according to the quality of the surface.

It is possible to create virtual paths with successions of vTags. We can decide that these tags can only be rated by the author (in order to delete them), and that the content is limited to a number corresponding to a road surface quality. And in some countries where maps are inexistent or forbidden, this will allows to create them in a collaborative way.

A particular way of doing tracking is passive recording. We do not need necessarily a human interaction to send tags. For instance, we can record every second all the information sent by the GPS (like the speed, the altitude...) during a ski trip, and then analyze our journey on Google Earth. But we can also collect data through other sensors (for instance the concentration of pollens, the signal strength of a mobile phone operator, the temperature...) and fill a geo-referenced database. If we have enough data, it becomes possible to interpolate the values for every position. We can therefore draw useful maps, like one indicating the concentration of pollens, or one that show where the network coverage of a phone operator should be improved.

2.16 It is generic

Applications that allow finding neighboring friends exist. Applications that allow tracking people or object exist too. Displaying geo-referenced objects on a map is in a no way something new. However, the architecture described in this document brings a new advantage: it is generic. It means that different new services can be built without writing a single line of code. For instance, the system allows easily in-the-field recordings, like in the GeoSkating project. But we can also set up a dating service just by defining the structure of the vTags. In addition to the web browser, we can easily imagine a generic tag-viewer application that gets the current position through a GPS, displays tags on a map and provides an interface to review them. This application will be able to connect itself to different servers in order to get their tags. Additional tools, like a visual tag scanner can be included as well. To our view, such a generic application would be sufficient for most vTag applications.

3 The trust engine

We saw that POIs can be trusted only if they come from a known and reputable source. In practice it consists in buying them from a company. The main disadvantages are the costs and the fact that information is not necessarily up-to-

date. A collaborative system like the one described in this paper cancels the cost and freshness problems, but then we need a trust mechanism that informs about the reliability of the tags. A trust engine is a software block between the application and the database that is responsible to add trust information when the database is updated, and that filter the tags returned to the user according to his trust relationships.

3.1 Related work

Lots of work has already been done in the trust context, and the obvious question that arises is why not just using well-known trust models and apply them to virtual tags? The answer is simply that it will not work. Indeed, traditional trust models are mainly designed with file sharing or auction applications in mind. In this case, people are rating each other and when user *A* wants to download a file (or buy an item) from user *B*, he questions the system in order to determine how trustworthy user *B* is. Currently, commercial systems (like e-Bay) are using very basic centralized systems, and the academics are suggesting solutions to transform such systems into peer-to-peer architectures. But spatial messaging is noticeably different from file sharing or auctioning. First of all, we want to take care about the context. For example time is important. Imagine that you see during summer time a tag that warns about a high risk of avalanches. Even if there is no snow anymore, it does not mean necessarily that the author was lying; it can also mean that the tag has been written six month ago. Second, we believe that trust cannot only be applied to users. The tags themselves have to maintain information so that a user can compute how reliable it is to him.

We already tackled the time component in a paper that has been published in the PST'06 proceedings [13]. In the survey, we wrote that several authors are aware about the difficulty to take the time into account, but no one proposed a trust model that gracefully solved the problem, or at least it was not directly applicable to virtual tags. Dimmock [14], who realized the risk module in the EU-funded SECURE project [15], concluded in its PhD thesis that "one area that the framework does not currently address in great detail is the notion of time." Guha [16] built a generic trust engine allowing people to rate the content and the former ratings. He recognized however that in case of highly dynamic systems (like in spatial messaging where tags can appear and disappear very quickly), "Understanding the time-dependent properties of such systems and exploiting these properties is another potentially useful line of inquiry." Most existing trust metrics update their trust values only after a specific action, like a direct interaction or the reception of a recommendation. The few trust engines that take the time component into consideration simply suggest that the trust value decreases with the time. Mezzetti's trust metric [17] consists in multiplying the trust value at time t by a constant between 0 and 1. We proposed in the ASG technical report [18] a similar model that also takes into consideration the dispersion of the outcomes. In Bayesian-based trust metrics [19], the trust value converges to its initial value over time. All these models work in situations where the changes occur slowly, but are challenged in short-lived cases.

Our former time-patterned trust metric, called TIPP GC (TIme-Patterned Probabilistic Global Centralized), was used in a collaborative application allowing to signal speed cameras on mobile phones. A full description of the trust engine and the application can be found in our former PST'06 paper. Even if we brought some novelties about the way we updated the trust values, we still used a "traditional" way to store them, i.e. the number of positive outcomes P and the number of negative outcomes N. The trust value equaled $P / (N + P)$. And under a certain trust value, the malevolent users where simply excluded from the system. The problem with this kind of metrics is that it is difficult to decrease the trust value of a user that behaved correctly for a long time. We suggest therefore, to be closer to the human way of handling trust, that any trust value must decrease quickly in case of bad behavior. An honest user that becomes malevolent must not be able to use its long term good reputation to subvert the system.

3.2 The uncertainty of the truth

In traditional computational trust, we usually agree over a set of axioms and hypothesis. For instance, the "truth" is a notion that is common to all. A corrupted file is seen as corrupted by everybody. In spatial messaging however, the truth is context dependant. The truth becomes a subjective and temporal notion. Something that is true for one user is not necessarily true for the others. Something that is true at a certain time is not necessarily true later. We call this new notion the "uncertainty of the truth". If user *A* posts a tag saying "Dangerous path", user *B* only knows that user *A* finds this path dangerous. But *A* is perhaps just a tourist and the path is in no way dangerous for user *B*, which can be a confirmed mountain guide. Or this path was maybe dangerous because of the snow, which melt away by the time.

To our view, trust is not only a tool that can be used to exclude malevolent users from a given system. Trust is also a way of creating relationships between users that behave in a similar way. Like in real life, each user has its own definition of what the truth is. The aim is therefore to create trust relationships between people that share the same definition.

3.3 Contextual trust

The trust given to a user varies according to the context. A first example of context is risk. Trust and risk are closely related topics. There is no use to trust if there is no risk. More precisely, the amount of trust you need before undertaking an action is correlated to the risk and the costs of a negative outcome. You will probably accept more easily to lend 2 € to an acquaintance that you meet sometimes by chance in front of the coffee machine, than 2000 € to a close friend, even if the chance that you will never get back your 2 € is much higher. The way you update a trust value of someone else is also related to the costs involved. If your friend gives you back the 2000 € you lent him, you will probably qualify him as reliable, and perhaps lend him more money next time. However, if it the coffee machine guy that returns you your 2 €, it is not sure that you will now trust him for a bigger amount. The opposite notion of risk is the benefit. Most of the time you undertake

an action only if you get a benefit in return. If you see a tag mentioning an interesting exposition, you will compare the benefit (having fun) with the risk (loosing time) and then compute the probability of a positive outcome according too your trust value in the tag's author.

A second example of context is the domain. You can trust someone for his knowledge in computer science, but have no trust at all in him for his tastes in cooking. When the domains are very different, we can simply decide to have one trust value per domain. In practice however, domains are not always that much different. There are often (more or less strong) links between them. If you know someone that excels in the domain of Java programming, you will trust him more easily to program you something in C++ than in repairing your car.

The third example of context is time and geography. If you know that a tag has been posted in a forest and using a GPS, we will be more tolerant about the precision of the positioning, since trees perturb heavily GPS devices. In the same way, you could decide to be more tolerant with a tag that has been posted by night, or a long time ago.

The fourth example is additional information. Additional information is everything that is not included in the above text. For instance, if you see a tag signed by a trustworthy friend testifying very good food in a restaurant, you will not go there if you learned by another mean (newspaper...) that rats are used in some meals. The tag can however remain true; it is a question of taste.

3.4 Updating trust values

A traditional way to store and update a trust value consists in counting the number of positive outcomes P, the number of negative outcomes N, and to define the current trust value as $P / (P + N)$. It is a simple model that fits very well to file sharing applications where a good file is simply considered as a positive outcome and a corrupted file as a negative one. In spatial messaging however, defining a positive and a negative outcome is more complicated. And since we have to deal with what we called previously the "uncertainty of the truth", we need to define a model that is specific for spatial messaging.

A model that can be used in case people are honest is one that uses data mining techniques in order to determine how reliable a given tag is, in a given situation for a given person. Basically, when you rate a tag, you enforce the trust link with all the people that reviewed it in the same way, and decrease the trust link with all the people that rated it differently. While requesting tags, data mining algorithms are then able to determine how "close" you are with each reviewer according to the situations where you previously interacted with these people, and take this into account to determine how pertinent this tag is to you.

This model is challenged when malevolent users take part in the system. For instance, an attack would consist in rating automatically and positively all new tags so that the next reviewers increases the malevolent user's trust value. And then this user will use its high value to post "reliable" funny tags. A solution to this consists in increasing only the trust value of the author of a tag, since posting randomly interesting tags (if they are not "interesting", nobody will rate them positively) is almost impossible.

In applications where it is possible to scan all the tags, and rate them in an automatic way, it is always possible to cheat the system. It is difficult in some cases to differentiate a normal behavior from a malevolent one. For instance, if you see a tag warning about a specific danger and you do not see this danger, you do not know if the author is a spammer (and you need to decrease its trust value) or if the danger simply disappeared (and then you should not decrease its trust value). To date, the only solution we found consists in specifying how a trust value changes according to a specific behavior, and a simulator helps us to find the values of the parameters. For instance, we will find how much a trust value must be decreased when we rate negatively a tag, so that an honest user is not too much penalized, but so that a spammer can be excluded from the system in a reasonable delay. It means that even if the system is generic, it needs a high comprehension of the application domain in order to determine what are the right rules and parameters. The main challenge today is to simplify all this, and ideally to end up with a single generic trust engine that is specialized for a given application only via a few parameters.

Like in the human world, trust varies not in the same way when it increases than when it decreases. It is well known that trust takes time be built, but can be destroyed very fast. And this non-linear way of handling trust is certainly necessary to protect ourselves. If you lent 10 times 2 € to someone that always paid you back, you will probably stop to trust him before 10 times when he stops refunding you. The reason is even more accentuated in a digital world where people can act in an automatic way, thus very fast. If we use our former $P / (P + N)$ example, it is easy for a user to behave correctly (most probably in an automatic way) for a certain time, and then use its high trust value to subvert the system. A first idea consists in representing a trust value as a single value. A good behavior increases it, a bad behavior decreases it. But the maximal value is limited. It means that even if someone behaves very well for years, its trust value is not that high, and can quickly become negative in case of a big bad behavior, or a succession of a few bad behaviors. Another important point is that trust increases in a linear way but decreases exponentially. We know that an exponential function varies very slowly at the beginning and then increases endlessly. Like in the human model, we forgive seldom and small misbehaviors, but we break our trust relationships if you we face a big misbehavior or a succession a small misbehaviors.

3.5 Deleting a tag

We saw previously that the system does not allow deleting directly a tag. The reason is obvious. A malevolent user could delete tags and since they do not exist anymore nobody will be able to decrease the trust value of this malevolent user. Instead, we can make a request to delete. For instance, if a few people rate the tag negatively, then a request to delete order is given to the tag. The latter remains visible for a certain amount of time, and if nobody rates positively the tags in the meantime, the tag is definitively deleted. But the fact that a user can rate a tag while a request to delete is made lets him the possibility to decrease the trust value of the malevolent user wiling to delete all the tags. The amount of time the tag remains visible can be determined according to the application. An example could

be to keep the tag for the same amount of time than elapsed between the creation of the tag and the request to delete order, so that an "old" tag needs more time to be deleted. In addition to that we can also define a minimum delay (to avoid that people scan the system in order to delete any new tag as soon as they appear) as well as a maximum delay (so that a "very old" tag still can be deleted in a reasonable time).

3.6 Reverse rating

Reverse rating is an automatic process that consist to rate the people that are rating you. For instance, an author can decide to rate positively all the users that rates its tag positively as well, since they seem to share the same opinion. There are two main advantages in doing reverse rating. The first is to speed up the creation of trust relationships. And the second, the most important, is to motivate people in publishing virtual tags. Otherwise the only motivation for an author is to update its reputation. But since this value is limited (to avoid future malevolent acts), there is no interest for an author to post lots of tags. However, if this allows him to build bidirectional trust relationships, so that the quality of the information he will get in the future improves, then the author has a clear motivation in publishing new tags.

However reverse rating can be easily exploited by malevolent users. The easiest way consists in scanning the system for new tags and to rate them automatically and positively, so that the authors increase the trust values of these malevolent users. To scope with that, we found mainly two solutions. The first is to hide the information and not allow system scans. For instance, a user that rate tags in different remote places within a short amount of time is suspicious. But again, defining what is suspicious and what is not is directly related to the application. This is especially true if we allow also using the system without being in the field. The second consists in posting funny tags in inaccessible places, or tags that are not true. If someone rates all the tags in an automatic way, he will soon or late fall in such a pitfall. And then the system can simply exclude this user from the system. Again, posting pitfalls is an operation that is application specific and that probably needs human interaction.

In short, reverse rating brings some advantages (motivation to post new tags, and speeding up the process of creating trust relationships), but need to be used with greatest care since malevolent users can use this opportunity to increase easily their own trust value in order to subvert the system.

3.7 Opinions versus facts

The fact that two users do not share the same opinion does not necessarily mean that one of them is malevolent. For instance, if you see a tag about a restaurant that pretends that the food is very good, it will not necessarily suits you, even if the author has a good reputation. In this case you will rate negatively this tag in order to mark the fact that you do not agree with the content, but however you will not ask the remove this tag since it can be useful for someone else.

But if you see a tag warning about a danger that clearly does not exist (anymore), then you will do a request to delete for this tag. By decreasing the trust

value of the author and the former reviewers that agreed with this tag, you will help the system to exclude potential malevolent users. The trust engine has therefore two roles: the exclusion of malevolent users and the creating of virtual communities grouping people sharing the same ideas and opinions.

3.8 Querying for tags

We saw previously that querying for tags consists in applying a filter to all the existing tags and return only the ones that cross the filter. We can, for example, ask all the tags around the current position that are authored by Alice and that contain "cat" or "dog" in the keywords. The trust engine offers an additional filter that accepts only "pertinent" tags. A pertinent tag is one that has been authored by a trustworthy friend sharing the same opinions than you. So that you are not disturbed by spammers or by tags that are irrelevant for you, like the ones that advertise about art expositions if you do not like art.

3.9 Web of trust

The notion of Web of trust has been defined by Zimmerman [20]. It says that if *A* trusts *B* and *B* trusts *C*, then *A* trusts *C*. The weight of a trust relation decreases with the number of levels. For instance, if *A*'s trust in *B* is 0.9 (out of 1), and *B*'s trust in *C* is 0.5, then *A*'s trust in *C* could be 0.9 * 0.5 * *k*, *k* being a constant in the interval]0..1[. Roughly speaking, a web of trust allows you to trust people to trust other people.

This notion is very useful in spatial messaging for big communities. Indeed, when you join a community, you do know the others and you are not able to trust them either. You need actually to interact with every of the users in order to know if you will be able to trust them next time. In a web of trust system, you just need to make a few friends. When you later face a tag signed by an unknown author, you will be able to ask your friends if this author is reputable. And your friends will ask their own friends, and so on up to a certain level.

To judge the trustworthiness of a tag, we combine the local trust value (our own opinion of the author, computed during past interactions) with the global trust value (the opinion of my friends, who also asked their own friends...). Each of these two values is weighted according to the application domain. For instance we could decide to take 20% of the local trust value and 80% of the global trust value. It is important to note that we always ask our friends for advice, even if we have a local trust value for the author. This allows us to still trust someone even if we gave him previously a bad rating, for instance if we saw a tag warning about a danger that disappeared. But this avoids us also to trust "old friends" that became malevolent since the last interactions we had with them.

3.10 Passive behavior

A passive behavior is when a user observes something that deserves to be tagged (like a rock on the road), but he does not report it. It can have its importance for the trust engine. Imagine you know that a given user drove on a specific road a few

minutes ago. You could think that he will tag any danger, so if there is not tag, it means that there is probably no dangers as well. But perhaps there is a danger and the driver could not tag this object, simply because he is alone in is car and it is thus not save to handle its mobile device while driving. Or he just wanted not to tag this object for any other reason.

We prefer a different approach. We prefer to consider that someone that does not tag is like someone that does not exist. Doing so solves the previous problem of knowing why a given user did not reported a given situation. Of course, doing so make us also loose some information. But remember that a tag can occupy any area. It can also be represented by a line. So a driver can also post a tag that covers exactly his path and that informs that there is no danger. Any following driver will therefore know that while he is inside the tag area, the road is safe.

4 Conclusion

We defined different requirements for virtual tags in order to have a single generic solution that can be applied in very different application domains. We saw that POIs (Points Of Interest), the most well known way to present geo-referenced information on a map, is actually only a simplified and limited instance of our virtual tags. In the second part of this paper, we showed that adding a specific trust mechanism to these tags allow us to cumulate the advantages of a collaborative system (where everybody can tag new object) and a reliable system (where the information is provided by a reliable source).

References

[1] http://poiplace.oabsoftware.nl/
[2] http://bbs.keyhole.com/
[3] http://www.mobiletag.com/
[4] http://www.muscle-noe.org/content/view/39/64/
[5] John R. Douceur, “The Sybil attack”, in Proceedings of the IPTPS02 Workshop, 2002
[6] http://www.truesenses.com/
[7] Michel Deriaz, "ASG technical report 06", 2006.
[8] http://www.ist-mobilife.org/
[9] http://www.flickr.com/
[10] http://www.michelderiaz.com/Softs/RegexSR/
[11] http://www.geoskating.com/
[12] http://www.geotracing.com/
[13] M. Deriaz and J.-M. Seigneur, "Trust and Security in Spatial Messaging: FoxyTag, the Speed Camera Case Study", in Proceedings of the 3rd International Conference on Privacy, Security and Trust, ACM, 2006.
[14] N. Dimmock, "Using trust and risk for access control in Global Computing", PhD thesis, University of Cambridge, 2005.
[15] http://secure.dsg.cs.tcd.ie/
[16] R. Guha, "Open Rating Systems", 2004.
[17] N. Mezzetti, "A Socially Inspired Reputation Model", in Proceedings of EuroPKI, 2004.
[18] M. Deriaz, "What is Trust? My Own Point of View", ASG technical report, 2006.

[19] S. Buchegger and J.-Y. Le Boudec, "A Robust Reputation System for P2P and Mobile Ad-hoc Networks", in Proceedings of the Second Workshop on the Economics of Peer-to-Peer Systems, 2004. Or D. Quercia, S. Hailes, and L. Capra, "B-trust: Bayesian Trust Framework for Pervasive Computing", in Proceedings of the 4th International Conference on Trust Management, LNCS, Springer, 2006.
[20] P. Zimmerman, "PGP Users Guide", MIT, 1994.