STRUDEL: Supporting Trust in the Dynamic Establishment of peering coaLitions

Daniele Quercia, Manish Lad, Stephen Hailes, Licia Capra, and Saleem Bhatti
Department of Computer Science, University College London, London, WC1E 6BT, UK.
{D.Quercia, M.Lad, S.Hailes, L.Capra,S.Bhatti}@cs.ucl.ac.uk

ABSTRACT

The Coalition Peering Domain (CPD) is a recent innovation within the field of mesh networking. It facilitates the management of community-area networks in a distributed and scalable form, allowing devices to pool their network resources (particularly egress links) to the common good. However, as in P2P systems, this form of cooperative sharing architecture raises significant concerns about the effect of free-riders: nodes that utilise the bandwidth of others without providing an adequate return to the community. To address this problem, we propose STRUDEL, a distributed framework that tackles the problem of free-riders and consists of: (i) a mechanism for the detection of malicious peers; (ii) a formal Bayesian trust model, to assess peers' trustworthiness; (iii) a forwarding mechanism based on the maximisation of trust-informed utility.

Categories and Subject Descriptors

D.2.0 [**Software Engineering**]: General—*protection mechanisms*; C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*distributed applications*

General Terms

Algorithms, Security

Keywords

Distributed Reputation Systems, Mesh Networks, Distributed Trust Models

1. INTRODUCTION

Sophisticated network applications demand considerable bandwidth, but the current Internet connectivity available to many (especially mobile) devices does not satisfy such demand. Increasingly sophisticated network applications are being offered to a diverse spectrum of users. As a result, the bandwidth demand imposed on networked devices has increased dramatically over the last few years. Although the data rates offered by local-area connections are usually sufficient to meet this demand, the opposite is true with respect to Internet connectivity for many mobile devices.

A new entity in the routing landscape, the Coalition Peering Domain (CPD), has been proposed to better cope with increasing bandwidth demands. We may observe that communities of users may between them have considerable Internet bandwidth in the form of the summed set of relatively low bandwidth GPRS connections. As a consequence, the CPD concept has been put forward to facilitate the exploitation of under-utilised local-area bandwidth in such a way as to enhance both Internet connectivity and connection survivability [9].

Intrinsically, CPDs rely on cooperative bandwidth sharing, which poses a "tragedy of the commons" dilemma [8]: if too many people exploit others' connections, the excess of free-riders drives away the people that make the coalition viable.

To address the "tragedy of the commons" in CPDs, we propose STRUDEL, a distributed adaptive framework that combines trust-informed selection of the forwarding path for packets with a mechanism for identifying and isolating misbehaving peers. It makes use of reputation evidence (i.e., direct experience evaluations and recommendations) to support trust and, consequently, the formation and maintenance of Coalition Peering Domains. It consists of:

- an approach for detecting malicious nodes based on the 2-ACK scheme, whose application to bandwidth sharing scenarios is innovative;
- a Bayesian formalization for trust formation and trust evolution that possesses a range of desirable properties (i.e., (i) support for fine-grained discrete trust metrics, as opposed to the binary metrics currently used by current Bayesian trust models; (ii) use of recommendations that are weighted according to recommenders' trustworthiness and recommenders' subjective opinion - to distinguish honest and dishonest recommenders and to resolve the different ontological views of the world honestly held by different peers; (iii) incorporation of the time dimension to prevent nodes from capitalizing excessively on past behavior);
- a forwarding mechanism that integrates the Bayesian trust model and locally maximizes each peer's utility.

This is achieved by minimizing the use of heavyweight mechanisms, and by removing the assumption of having a public

SAC'06, April, 23-27, 2006, Dijon, France.

key infrastructure (PKI) in place. In fact, the most heavyweight mechanisms (e.g., per packet signatures) are only activated when there is evidence to suppose that misbehavior is occurring. In addition, to support the 2-ACK scheme, STRUDEL does not require a trusted binding between a real identity and corresponding public key, but rather only between a peer address and its public key. This can be achieved by means of Cryptographically Generated Addresses [3], without the need for a PKI.

The remainder of the paper is structured as follows. Section 2 discusses prior work about trust management frameworks and their application to bandwidth sharing scenarios. Section 3 introduces the idea of the Coalition Peering Domain. Section 4 describes a peer's state machine consisting of a Bayesian trust model and a forwarding mechanism. Section 5 draws the conclusions.

2. RELATED WORK

A broader concept of trust in the computer science arena began with work in the early 1990s by Marsh [11]. Abdul-Rahman and Hailes first proposed the use of recommendations for managing trust [1] and then introduced a comprehensive distributed trust model for online environments [2], based on Marsh's model. From direct experiences and recommendations, each entity forms its trust, and it is also able to deal with false recommendations. Although foundational, the previous approach suffered from being rather too architectural in style: they lacked processes for trust evolution, for example. To fill the gap, Mui *et al.* [12] proposed a Bayesian formalization for a distributed rating process. However, two issues remained unsolved: they considered only binary ratings and did not discount them over time. Buchegger and Le Boudec [4] tackled the latter issue, but not the former: they proposed a Bayesian reputation mechanism in which each node isolates malicious nodes, ages its reputation data (i.e., weights past reputation less), but can only evaluate encounters with a binary value (i.e., encounters are either good or bad). Using a generic n-level discrete trust metric, our Bayesian model addresses the issue. Furthermore, each peer discounts its trust beliefs over time (i.e., it decreases the confidence level it has in its trust beliefs).

Since that time, numerous papers have described either ad-hoc or more formal distributed trust models, but their integration with decision-making mechanisms, though fundamental, is less well developed. Within the SECURE project, Carbone *et al.* [5] proposed a formal model for trust formation, evolution, and propagation based on a policy language whose output (trust values) feeds a decision-making mechanism [6]. More recently, Quercia and Hailes [13] proposed a decision model for reputation-based interactions that, on input of reputation assessments, estimates the probability of potential risks associated with an action based on which it decides whether to carry out the action.

Given the novelty of the coalition peering domain architecture, its integration with distributed trust and decision-making frameworks is unexplored territory. Zhu and Mutka [15] proposed a credit-based trust system for connection sharing among ad-hoc network devices. Although similar, their proposal differs from the concept of coalition peering domain. As each device takes turn to act as a temporary gateway, the aggregated bandwidth in their proposal is less than that in a coalition-based approach, in which multiple devices simultaneously act as gateways. Also, the credit-based trust system uses ad-hoc trust evolution rules and does not discount past reputation data over time.

3. COALITION PEERING DOMAIN

The formation of a Coalition Peering Domain (CPD) emerges from a new class of community-area networks (e.g., Consume.net and FreeNetworks.org) [7]: one in which individuals connect together their home and/or personal-area networks, on an ad hoc basis, forming a local *mesh* or *community network*. Although existing mechanisms are employed by users to administer these existing community networking initiatives, such mechanisms tend to be manual, slowly changing and relatively limited in their abilities. In particular, they tend to concentrate on sharing a single connection between multiple machines and do not take into account the possibility of utilising multiple connections simultaneously. Within a CPD, however, devices share multiple connections simultaneously. The under-utilised local-area connectivity between Coalition Members (CMs) is used to distribute traffic across all CMs that are willing to share their wide-area connectivity. Therefore, CMs are able to better utilise the full aggregate wide-area bandwidth that is available to the CPD. For example, consider that the Internet connectivity of 4 PDAs increases by a factor of 5.5 when joining a CPD [10].

The resulting community-area networks do not represent a single Administrative Domain (AD) that is under the control of a single organisation or entity, but rather a *collaborative* group of such entities. Existing mechanisms are designed to operate in network environments where administrative responsibility is not distributed. However, such an approach cannot be applied to the community network environment because it is most unlikely that members would be willing to trust all others unconditionally.

The "tragedy of commons" that arises from too many free-riding coalition members exploiting resources made available by others, without actually sharing their own resources within the CPD, undermines the value of the CPD.

By using reputation evidence to evaluate the behaviour of peers, STRUDEL provides a distributed framework to overcome this hurdle.

4. A PEER'S STATE MACHINE

Each peer in a CPD can be described by the state machine depicted in Figure 1.

To join a CPD, an IDLE peer p_X broadcasts a join request (JR) message, containing a list of its minimum requirements for peering to be feasible (ranging from minimum bandwidth, maximum loss rate, and so forth through to constraints on the credentials that are acceptable). The peer p_X enters the AGREEMENT state in two possible situations: (i) p_X receives a request accepted (RA) message from p_Y in response to its JR. The peer p_X evaluates peering agreement terms contained in the RA (e.g., amount of bandwidth p_Y is willing to offer), p_Y's credentials (i.e., other peers' ratings about p_Y, some of which may be forwarded by p_Y), and p_X's locally historical experiences with p_Y. Based on this information, p_X decides whether or not to initiate a peering agreement with p_Y; (ii) p_X receives a JR from a peer p_Y, asking to form a CPD with p_X; as JR also contains p_Y's credentials, p_X evaluates them and decides whether to accept (thus sending a RA message back), or to reject (thus

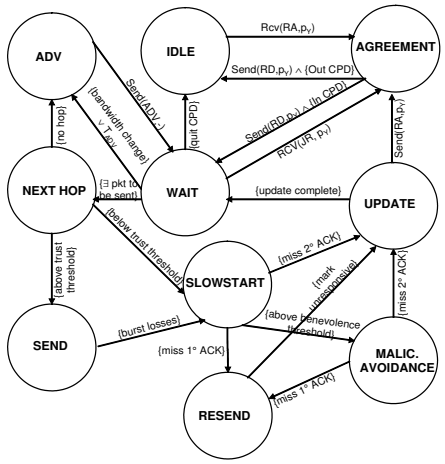

Figure 1: A peer's state machine.

sending a request denied RD message) the peering agreement.

If p_X accepts at least one peering agreement, it enters the WAIT state, meaning that it belongs to at least one CPD and it is ready for action. It may then enter the ADV state to send advertisements to its peering nodes either to refresh its peering agreement terms periodically or to change them (e.g., when p_X's available bandwidth changes). An advertisement $ADV_{X,Y}$ from p_X to p_Y contains: (i) the total (Internet plus local) bandwidth that p_X is willing to offer to p_Y; (ii) p_X's current trust in p_Y. Therefore, from the advertisement, p_Y determines the amount of traffic that p_X is willing to receive and obtains a trust tuple that it may later use as recommendation letter.

Whenever p_X has to send packets, it must first select the forwarding peer p_Y, based on stored reputation and routing information, from p_X's ISP and the set of peers with which p_X has an agreement. The selection process is performed in the NEXT-HOP state and is based on utility maximisation, where utility is a measure of delivery probability. Based on the next-hop's reliability, p_X selects one of the two sending modes: either normal operation mode or suspicious mode. If p_X deems p_Y reliable, it enters normal operation mode represented by the SEND state. Here, p_X sends packets to p_Y without any mechanism for maliciousness detection, other than an assessment of end-to-end loss rate. At this point unless there is some mechanism for assessing whether packets have been delivered, there is no incentive for a forwarder to act honestly. If an end-to-end mechanism is available - for example, TCP acknowledgements - then it may be used to assess whether the loss rate on the entire link between sender and destination is an acceptable one. We know that if the loss rate is acceptable, then peers within the community are well (enough) behaved and we need not track their activities more closely. However, the converse is not true - if the end-to-end loss rate is unacceptable, it is still entirely possible for all peers to behave honestly and well, and for the loss to be occurring within the wider Internet. In this case, we need to identify the cause of the problem and, as a conequence, if the loss rate is unacceptable, or if there is no end-to-end mechanism for acknowledging p_X switches into *suspicious mode.*

In suspicious mode, all forwarding peers are required to use Pau and Mitchell's 2-ACK scheme [14]. At the price of greater packet overhead, the scheme allows the identification of suspicious peers along the path and also allows us to distinguish them from unresponsive peers. Suspicious mode comprises two substates: SLOW START and MALICIOUSNESS AVOIDANCE. In the former state, p_X sends packets to p_Y according to a sending window which grows exponentially. After exceeding a threshold (benevolence threshold), p_X enters the latter state, in which the sending window grows linearly. Section 4.2 discusses suspicious mode in detail.

If the next-hop is unresponsive, p_X enters the RESEND state and retransmits the same packet up to a retransmission threshold. When p_X exceeds the threshold, it marks p_Y as unresponsive and enters the UPDATE state to update its routing information. Reputation information is updated when either (i) a next-hop node is cleared of suspicion; (ii) or it initiates a peering agreement.

In the following subsections, we consider three of the key states in more detail.

4.1 NEXT-HOP State

In this state, the peer p_X uses a utility-based decision mechanism to judge whether to send the next packet in its queue and, if so, to choose the next-hop p_Y that will act as packet forwarder. For this purpose, p_X uses a modified version of the risk-aware decision model by Quercia and Hailes [13]. This decision model includes the following elements:

1. A set of *actions* that p_X can carry out, such as *'send packet to p_Y'* or *'drop the packet'*.

2. A set of *states* that completely defines the set of occurrences of interest, such as s_1 =*'acceptable packet delivery through p_Y'*, s_2 =*'just-in-time packet delivery through p_Y'*, and s_3 =*'out-of-time packet delivery through p_Y'*. The peer p_X associates with each state an acceptable latency (which corresponds to the residence time in that state) that is dependent both on the next hop and on the *packet type.* For example, there will be different acceptable latencies for the state s_2 =*'packet sent just in time through p_Y'* depending on whether the packet belongs to real-time traffic or to FTP traffic.

3. A set of *state probabilities.* To compute the probability of a state, p_X first needs to compute: (i) the state residence time (i.e., the maximum packet delay from p_X to the destination under which each state will continue to obtain); as described previously, p_X maps each j^{th} state with a state residence time d_j that is a function of the packet type; (ii) the expected packet delay ed^Y from p_X to the destination when p_X selects p_Y as the next-hop. p_X computes the expected delay ed^Y as a function of the RTT $d^Y_{\text{net}} = \text{RTT}/2$, and of the delay d^Y_{trust} caused by p_Y (which depends on p_X's trust in p_Y): $ed^Y = d^Y_{\text{net}} + d^Y_{\text{trust}}$. Observing that the smaller the difference between the midpoint of the interval $[d_{j-1}, d_j]$ and the expected delay ed^Y, the more likely the state, we compute each j^{th} state probability as $P_j = \frac{\frac{1}{|ed^Y - \frac{d_{j-1}+d_j}{2}|}}{\sum_j \frac{1}{|ed^Y - \frac{d_{j-1}+d_j}{2}|}}$. The probability of the

j^{th} state is the reciprocal of the difference between the midpoint and the expected delay divided by a normalization factor. If such difference is zero, then the state probability is maximum.

4. A set of *outcomes* such as o_1 ='*packet sent without significant delay*', o_2 ='*packet sent just-in-time*', and o_3 ='*packet dropped*', and o_4 ='*packet sent too late*'. For a given state and a given action a unique outcome exists. For instance, for the state s_2 ='*just-in-time packet delivery through* p_Y' and p_X's action '*send the packet to* p_Y', the outcome o_2 ='*packet sent just-in-time*' takes place.

5. An elementary *utility function* u which, given an outcome, returns p_X's utility (ranging in $[0, 1]$) for the outcome. Considering the notation above, p_X could have the following utility ordering: $u(o_1) > u(o_2) > u(o_3) > u(o_4)$.

6. A *decision rule*. The peer p_X computes its utility for each next-hop p_Y and action a_i: $U_{a_i}^{p_Y} = \sum_j P_j \cdot u(s_j, a_i)$. It then carries out the action a_m with the peer p_q so that it maximizes its utility: $U_{a_m}^{p_q} = max_{\forall p_Y, a_i}\{U_{a_i}^{p_Y}\}$.

4.2 Suspicious Mode: SLOW START and MALICIOUSNESS AVOIDANCE States

Suspicious mode is entered whenever we suspect that a node or set of nodes on the path to the destination is misbehaving. The purpose of the mode is rapidly to identify whether there is misbehaviour and, if so, who is responsible for it, so that they may be excluded from the path, and in such a way that they are less likely to be included in future paths. However, such identification mechanisms are costly, and, where nodes appear to be well-behaved, we wish to minimise the impact on them.

Suspicious mode is analogous to TCP congestion control: we wish effectively to exploit transmission capacity. Therefore, we shape the suspicious mode in TCP congestion control's likeness. The suspicious mode comprises two states: slow start and maliciousness avoidance. As TCP congestion control has a congestion window constraining the sending rate, the suspicious mode has a benevolence window that constrains the sending rate as well and grows exponentially during the slow start, and linearly during the maliciousness avoidance.

Before describing how the suspicious mode unfolds, we briefly describe the 2-ACK scheme [14], which the suspicious mode uses to detect either unresponsive or suspicious peers. Consider three peers: p_X, its next-hop p_Y, and p_Z (p_Y's next-hop). With the 2-ACK scheme, after sending a packet to p_Y, p_X has to receive two acknowledgments from p_Y: one cryptographically signed p_Y (called *one-hop away acknowledgment*) and the other by p_Z (called *two-hops away acknowledgment*). The scheme runs as follows: p_X sends a packet to p_Y; p_Y receives the packet, signs an acknowledgement with its private key, sends the acknowledgment to p_X, and sends the packet to p_Z. Upon receiving the packet, p_Z signs an acknowledgment with its private key and sends it to p_Y; p_Y forwards the acknowledgment to p_X. If p_X does not receive the one-hop away acknowledgment, it retransmits the same packet up to a retransmission threshold. If it still does not receive any acknowledgment back from p_Y, p_X marks p_Y as *unresponsive*. If p_X does not receive the two-hop away acknowledgement instead, p_X marks p_Y as *suspicious* because p_Y either did not send the packet to p_Z or relied on an untrustworthy peer p_Z.

When entering in the suspicious mode, p_X sets its benevolence window (i.e., the number of packets p_X sends before requiring a 2-ACK) to $q = 1$ and enters the SLOW START state. At each stage it sends q packets to p_Y and, if it receives the two-hop away acknowledgment for this set, increases its benevolence window by q. As a consequence, the benevolence window grows exponentially.

The peer p_X's benevolence window, grows exponentially until one of the following events occurs: (i) the benevolence window value exceeds a threshold, called the benevolence threshold, in which case, p_X enters the MALICIOUSNESS AVOIDANCE state and increases its benevolence window linearly; (ii) p_X misses at least one two-hop away acknowledgment, in which case, p_X sets the threshold to half the current benevolence window size; it then enters the UPDATE state and refreshes its trust in p_Y based on the amount of bandwidth p_Y claimed to offer (in a similar way to the way that lack of TCP acknowledgments signal congestion, lack of two-hop away acknowledgments signal suspicious behavior); (iii) p_Y offers the transmission capacity that it promised, in which case the window remains fixed and a timer is set; if the timer expires, the node returns to normal operation, and the bounding packet loss parameters for the end-to-end link are updated to avoid state flapping.

4.3 UPDATE State

The main purpose of this state it to allow p_X to update its trust information. It uses a Bayesian trust framework for pervasive computing. p_X's *overall* trust in p_Y combines p_X's direct trust in p_Y (i.e., trust p_X builds after direct experiences with p_Y) and p_X's recommended trust in p_Y (i.e., trust p_X builds from others' recommendations about p_Y).

Since the random variables describing direct trust, recommended trust, and overall trust are discrete (i.e., they assume one of n discrete values $\{l_1, \ldots, l_n\}$), *B-trust* has numerous advantages: (i) the random variable distributions emerge as a consequence of updates and are not fixed *a priori*, as existing models impose; (ii) a generic n-level metric is more fine-grained than a binary metric (for which an entity is either completely trustworthy or completely untrustworthy), as existing models impose; (iii) discrete metrics are more computationally tractable than continuous metrics (e.g., they do not involve the computation of integrals).

We now discuss when and how p_X updates its direct, recommended, and overall trust beliefs in a peer p_Y. The peer p_X updates its *direct trust* after completing a suspicious mode session with any peer p_Y. Let $DT_{X,Y}$ be a random variable expressing p_X's direct trust in p_Y and $DE_{X,Y}$ be a random variable expressing p_X's evaluations of direct experiences with p_Y. The peer p_X carries out the Bayesian revision process as follows:

1. for $\alpha = (1, \ldots, n)$, p_X has prior probability of the event $DT_{X,Y} = l_\alpha$ (i.e., p_X deems p_Y deserves a level l_α of direct trust) and has prior conditional probability of the event $DE_{X,Y} = l_\beta$ (i.e., p_X evaluates the direct experience with p_Y with a l_β satisfaction level) given the event $DT_{X,Y} = l_\alpha$ took place.

2. p_X computes its satisfaction for the just completed suspicious mode with p_Y:

$$s = \begin{cases} \frac{\delta T}{T}\frac{AVG}{OFF} & \text{if } \delta T < T \text{ and } AVG < OFF \\ \frac{AVG}{OFF} & \text{if } \delta T \geq T \text{ and } AVG < OFF \\ 1 & \text{if } AVG \geq OFF \end{cases}$$

where AVG is p_X's average throughput during the immediately previous suspicious mode, OFF is the throughput that p_Y was supposed to offer, δT is the duration of the suspicious mode, and T is the typical suspicious mode's duration. As p_X's throughput comes closer to p_Y's promised throughput, the satisfaction s rises. If $s \in [(\beta-1)\cdot\frac{1}{n}, \beta\cdot\frac{1}{n}]$, for any $\beta \in [1,n]$, p_X evaluates the direct experience with p_Y with a l_β satisfaction level (i.e., the event $DE_{X,Y} = l_\beta$ takes place).

3. Given this level of satisfaction, p_X applies Bayes' theorem to update its direct trust. It thus computes $p(DT_{X,Y} = l_\alpha | DE_{X,Y} = l_\beta)$ for $\alpha = (1,\ldots,n)$ as $\frac{p(DT_{X,Y}=l_\alpha)\cdot p(DE_{X,Y}=l_\beta | DT_{X,Y}=l_\alpha)}{\sum_{\gamma=1}^{n} p(DT_{X,Y}=l_\gamma)\cdot p(DE_{X,Y}=l_\beta | DT_{X,Y}=l_\gamma)}$

Similarly, the peer p_X updates its *recommended trust* ($RT_{X,Y}$) upon receiving a set of credentials from any peer p_Y.

Finally, to determine its *overall trust* in p_Y, p_X computes, $\forall j \in [1,n]$, $p(T_{X,Y} = l_j) = \sigma \cdot p(DT_{X,Y} = l_j) + (1-\sigma)\cdot p(RT_{X,Y} = l_j)$, where $T_{X,Y} = l_j$ is the event 'p_X *deems* p_Y *deserves an* l_j *overall trust*' and σ holds the importance p_X places in direct experiences over recommendations.

Two more aspects are worthy of discussion: how p_X *ages* its trust beliefs and how it *bootstraps* them. Direct trust and recommended trust confidence levels decrease with time. The aging process is carried out by decreasing the trust probabilities that are greater or equal to $\frac{1}{n}$ and increasing those that are below $\frac{1}{n}$, so that the probabilities sum to 1. As for bootstrapping, when the peer p_X meets p_Y for the first time, it has no information about p_Y; its beliefs are thus uniformly distributed, i.e., $\forall j \in [1,n]$: $p(DT_{X,Y} = l_j) = p(RT_{X,Y} = l_j) = \frac{1}{n}$.

5. CONCLUSION

We have presented an adaptive framework, named STRUDEL, that allows CPDs to isolate malicious peers and, thus, minimize connectivity disruptions. Using STRUDEL, peers are able to identify malicious behaviors by means of the 2-ACK scheme. They keep track of other peers' reputation and team up with only trustworthy ones. They do this by means of a fully distributed Bayesian trust model that updates reputation data structures based on direct experiences and recommendations, and that produces trust assessments. Based on such assessments, a decision-making process maximizes each peer's expected utility, with the result that well behaved peers will preferentially be selected, and with the overall effect that malicious ones will be isolated. As such, most traffic will flow through cooperating peers, whilst free-riders will be excluded from the Coalition Peering Domain.

Two general models stand out from STRUDEL: the Bayesian trust model (described in subsection 4.3) and the decision-making model (described in subsection 4.1 and widely discussed in [13]). They are general in the sense that many applications, in addition to bandwidth sharing, benefit from them. Any decentrilized collaborative application benefits from both the trust model, which manages collaborating parties' reputation information, and the decision-making model, which on input of reputation information determines the best possible action.

6. ACKNOWLEDGEMENTS

The authors gratefully acknowledge the support of the European Commission through the SEINIT and RUNES projects.

7. REFERENCES

[1] A. Abdul-Rahman and S. Hailes. Using Recommendations for Managing Trust in Distributed Systems. In *Proc. of IEEE Malaysia International Conference on Communication*, Kuala Lumpur, Malaysia, November 1997.

[2] A. Abdul-Rahman and S. Hailes. Supporting Trust in Virtual Communities. In *Proc. of the* 33^{rd} *IEEE Hawaii International Conference on System Sciences*, volume 6, page 6007, Washington DC, USA, 2000.

[3] T. Aura. Cryptographically Generated Addresses (CGA). In *Proc. of the* 6^{th} *LNCS International Conference on Information Security*, pages 29 – 43, Bristol, UK, 2003.

[4] S. Buchegger and J.-Y. L. Boudec. A robust reputation system for p2p and mobile ad-hoc networks. In *Proc. of the* 2^{nd} *Workshop on the Economics of Peer-to-Peer Systems*, Cambridge, MA, USA, June 2004.

[5] M. Carbone, M. Nielsen, and V. Sassone. A Formal Model for Trust in Dynamic Networks. In *Proc. of the* 1^{st} *IEEE International Conference on Software Engineering and Formal Methods*, pages 54–63, Brisbane, Australia, September 2003.

[6] N. Dimmock. How much is 'enough'? Risk in Trust-based Access Control. In *Proc. of the* 12^{th} *IEEE International Workshop on Enabling Technologies*, page 281, Washington, DC, USA, June 2003.

[7] N. Easen. Welcome to the Wi-Fi revolution. In *http://edition.cnn.com/*, September 2003.

[8] G. Hardin. The tragedy of the commons. *Science*, 162:1243–1248, December 1968.

[9] M. Lad, S. Bhatti, S. Hailes, and P. Kirstein. Enabling Coalition-Based Community Networking. In *Proc. of the London Communications Symposium*, London, UK, September 2005.

[10] M. Lad, S. Bhatti, P. Kirstein, and S. Hailes. Challenges, Opportunities and Incentives for Coalition-Based Community Networking. In *Technical Research Note Number RN/05/08. University College London*, London, UK, June 2005.

[11] S. Marsh. Formalising Trust as a Computational Concept. Ph.D. Thesis. Department of Mathematics and Computer Science, University of Stirling, 1994.

[12] L. Mui, M. Mohtsahemi, C. Ang, P. Szolovits, and A. Halberstadt. Ratings in Distributed Systems: A Bayesian Approach. In *Proc. of the* 11^{th} *Workshop on Information Technologies and Systems*, New Orleans, Louisiana, USA, December 2001.

[13] D. Quercia and S. Hailes. Risk Aware Decision Framework for Trusted Mobile Interactions. In *Proc. of the* 1^{st} *IEEE/CreateNet International Workshop on The Value of Security through Collaboration*, Athens, Greece, September 2005.

[14] P.-W. Yau and C. J. Mitchell. 2HARP: A secure routing protocol to detect failed and selfish nodes in mobile ad hoc networks. In *Proc. of the* 5^{th} *World Wireless Congress*, pages 1–6, San Franciso, USA, 2004.

[15] D. Zhu and M. W. Mutka. Promoting Cooperation Among Strangers to Access Internet Services from an Ad Hoc Network. In *Proc. of the* 12^{nd} *IEEE International Conference on Pervasive Computing and Communications*, pages 229–240, Orlando, FL, USA, March 2004.