

Literature Review
**Resource Management in Suburban Ad-Hoc
Networks with Decentralized Capabilities**

Stanley Gunawan
School of Computer Science and Software Engineering,
Monash University

August 11, 2003

Contents

1	Introduction	2
2	Outline	2
3	Availability techniques	3
3.1	Detection approaches	3
3.1.1	Distributed and Cooperative Intrusion Detection . . .	3
3.1.2	Watchdog and Pathrater	4
3.1.3	CONFIDANT (Cooperation Of Nodes, Fairness In Dy- namic Ad-hoc NeTworks)	5
3.1.4	CORE (Collaborative Reputation)	6
3.2	Incentive based approaches	6
3.2.1	Terminodes	8
3.2.2	Sprite	9
3.2.3	Secure Charging Protocol in Ad Hoc Stub Networks .	10
3.2.4	Priority Forwarding	10
4	Conclusion	11
	Bibliography	11

1 Introduction

A Suburban Ad Hoc Network (SAHN) is a self-organizing, quasi-static ad hoc network. It provides high speed connectivity in a suburban community of cooperating nodes. Typically a gateway is attached to it, to provide access to the Internet, a company or university intranet, or another suburban network.

A key property of SAHN is its self-organizing nature. The nodes in the network rely on each other to provide connectivity and resources. The absence of central infrastructures and authorities means that the network can emerge out of only its participants, removing the need of an authority figure.

But unfortunately, robust self-organization is very hard to achieve. It's very hard to maintain availability since the network relies on its participants being unselfish, and there are incentives not to cooperate: saving own resources. As according to [1], servicing forwarding requests can greatly effect a node's own throughput.

2 Outline

This literature review focuses on existing techniques that foster availability in ad hoc networks generally, and not specific to SAHNs. In the higher level, SAHN's architecture is quite general and very similar to other existing ad hoc network architectures. So most availability techniques can be applied to SAHN. In fact, most of the techniques can work more effectively in SAHNs due to the quasi-static nature.

Having said that, due to SAHN's generality, techniques that take advantage of a specific characteristic of the network that they're designed for are probably not suitable for SAHN. An example is the availability approach for ad hoc stub networks, which assumes and exploits the existence of access points and gateways in the networks.

Note that the majority of existing availability techniques for ad hoc networks concentrates on the issue of cooperation for the purpose of routing and packet forwarding. Although they are not the only resources that nodes can, or need to share, they are the most important ones. Ad hoc network is about the lowest level of networking, hence the major activity is the sending of a packet from one point to another.

Generally, existing techniques that address availability can be divided

into techniques that detect nodes that display behaviours that can threaten availability, and the ones that provide incentives for nodes to cooperate and not misbehave. In the following chapters, I'll discuss some of them.

3 Availability techniques

3.1 Detection approaches

Detection based approaches mostly relies on having some of the network's participants to detect other nodes' misbehaviours. This is usually done by listening on network activities in each node's local coverage. What differentiates the methods below is the way they use the information gained from detection.

All of the detection methods described below, and most other similar existing methods suffer from the problem that the method used for detection relies on the broadcast nature of wireless networks. Unfortunately not all wireless network technologies support this. For example, monitoring won't work well when directional antennas are used.

3.1.1 Distributed and Cooperative Intrusion Detection

An Intrusion Detection System (IDS) is a system commonly deployed in networks in order to detect and react to intrusion attempts. Intrusion is defined as "any set of actions that attempt to compromise the integrity, confidentiality, or availability of a resource" [2]. In traditional networks, IDS components are installed in the fixed infrastructures like routers, switches and gateways, where they can monitor the activity of the network. The absence of such infrastructure in ad hoc networks means that traditional IDS are unusable in such setting.

In [3] Zhang and Lee describes an IDS designed for mobile ad hoc networks, which is distributed and cooperative. Every node in the network shares the responsibility of detecting and responding to intrusion attempts in their local coverage. Each node continuously collects network activity data in its local coverage, and if it detects a definite intrusion attempt, an appropriate response will be initiated. On the other hand, if the evidence is not conclusive, it can propagate the information to other nodes, asking for further information. If the combined evidence is definite, the effected nodes can initiate a response. Such response could be re-authenticating a node, or maybe isolating it if the evidence shows that it has been compromised.

The nature of any detection system, especially IDS, means that such system must compile a large database of intrusion patterns. This needs to be updated from time to time, and the updates need to be distributed to all nodes. This is not a simple task. IDS is effective for detecting active misbehaviours, but the nature of ad hoc networks is that it's more vulnerable to passive misbehaviours, like not forwarding packets.

3.1.2 Watchdog and Pathrater

Marti, Giuli, Lai and Baker [7] propose a modification to routing protocols in ad hoc networks so that they include a *Watchdog* and a *Pathrater*. The watchdog component detects misbehaving nodes, and the pathrater rates paths according to the knowledge it gains from the watchdog.

Every node is associated with a watchdog, and whenever it forwards a packet, its watchdog makes sure that the next node in the path forwards too. The watchdog does this by eavesdropping on the node's outgoing packets. The watchdog maintains a buffer of packets recently sent from its owner node, and compares them to packets it overhears from the next node in the path. If there's a match that means that node has forwarded the packet, so it's not misbehaving. The packet is then removed from the buffer. If there's no match after a certain time, the watchdog mark the node as misbehaving. Note that the watchdog must be within its owner's transmission range, for it to be able to eavesdrop on the neighbour nodes transmissions.

A node that wishes to globally transmit a packet can then calculate the *path metric* of every available path. It takes into account past successful transmission, the link reliability, and the existence of known misbehaving nodes in the path. The pathrater effectively avoids any path with misbehaving nodes.

Although this system is quite effective in choosing good paths, it doesn't punish the misbehaving nodes. Those nodes still benefit from the network, while saving their own resources. This effectively encourages selfish behaviours. It's also hard to make a perfect watchdog. There are many known methods to circumvent the watchdog under certain cases, although mostly minimal, but it's impossible to detect every possible kind of misconduct.

3.1.3 CONFIDANT (Cooperation Of Nodes, Fairness In Dynamic Ad-hoc NeTworks)

In the papers [4, 5, 6], the authors propose a system similar to the Watchdog and Pathrater system, in that nodes learn of other nodes behaviours by observation. The paper extends the idea further to not only take into account observed misconduct, but also misconduct reported by other trusted nodes. Furthermore, unlike the watchdog/pathrater system, rogue nodes are punished. Nodes with bad reputation are isolated, so thus their activity in the network is limited.

To achieve this, the paper proposes that each node in the network have these components: *monitor*, *reputation system*, *trust manager*, *path manager*. The monitor observes any routing and forwarding misconduct of its neighbouring nodes. It then passes this information to the reputation system, which will then update its perception of the node.

Once the reputation system considers the node to be intolerable, it tells the path manager to remove all paths that contain the considered rogue node. In addition to being responsible for route discovery, the path manager is also responsible for handling route requests; and it takes into consideration whether the request has the reported rogue node is in the source route.

The reputation system also asks the trust manager to issue a message to the node's *friends*, which are nodes that it trusts. The message contains the type of misconduct and the number of observed occurrences. Upon receiving this message, the friend nodes' trust manager asses the trustworthiness of the reporter node. It also considers whether it has received similar report from other nodes before. If it consider the report trustworthy, it passes this information to the reputation system, which then updates its perception of the allegedly rogue node.

This system addresses availability better than the aforementioned watchdog/pathrater system, but it's also much more complex and has too many variables. There's simply too much potential misbehaviour that need to be detected for the punishment to be effective. For example, the monitor might falsely, intentional or not, report that a node is misbehaving. The system tries to lessen the effect of false reporting by the ability to update perception due to new information, and by assigning time limit to reputation information, but they don't solve the problem. Another example is collusion, agreement between detectors and rogue nodes to trick everybody else.

A more inherent problem is that a reputation system is subjective. A

node that act badly only to others that it doesn't expect to benefit from won't be effected by isolation. Reputation system being subjective also means that everybody wants to choose paths with the best reputed nodes, which means they are then one that will get the most job requests, regardless of the nature of the job. So there's a disincentive to be too well reputed.

3.1.4 CORE (Collaborative Reputation)

The CORE mechanism is a reputation based system similar to the ones before. Each node has a watchdog component which monitors the behaviour of neighbour nodes, and a reputation component which calculates reputation of known nodes based on information given by the watchdog.

Like CONFIDANT, the CORE mechanism supports subjective, first-hand reputation; and indirect, reported reputation. But unlike CONFIDANT, CORE doesn't allow negative rating. This prevents malicious false accusations. Since nodes can only commend other nodes' good behaviour, nodes that never got any positive rating eventually have very low reputation, and like in CONFIDANT, they get isolated. CORE further improves the reputation system by not just calculating a single for a node, but also for each *function*. For example, a node can be well known for forwarding packets, but not for participating in routing information gathering. Each function is then given a weight, and a formula is used to calculate the total reputation.

CORE improves on the previous reputation system by not allowing negative reporting, but it still relies on nodes behaving properly, commending other nodes that behave well. There's an incentive not to do so: since reputation values can only go up, it benefits a node to prevent other nodes from getting better reputation.

CORE suffers the same problem as CONFIDANT in that job requests concentrates on well reputed nodes. Although the consideration of *functional reputation* helps balance the load of well reputed nodes to jobs that it does well.

3.2 Incentive based approaches

The goal of incentive based approaches is to foster cooperation, without relying on participants to report and punish misbehaving nodes. In other words, instead of nodes monitoring and punishing other nodes, this type of approach promotes positive behaviours.

All the projects reviewed below use a virtual money concept, which is used to charge for resource usage. They mainly concentrate on charging for packet forwarding, such that the forwarder gets some amount of virtual money, usually very small, from the packet originator. In this scheme, nodes have the incentive to forward packets. They need to gain virtual money, in order to be able to request other nodes to forward its packet too. Generally, the aim of money-based system is to limit resource usage by shared resource. This goes both ways in that besides limiting free-riding, resource charging also makes resource abuse like flooding-style denial-of-service attack (DoS) “expensive”.

Limiting resource usage by resource shared makes more sense than the aim of detection based systems, which is to monitor all misbehaviours, and optionally punish the misbehaving nodes. When faced between forwarding a packet, and sending its own packet, a node should be able to prioritise its need. Forcing to do otherwise will create an environment that’s too controlled. Nodes often have periods when they are very active, and when they aren’t. They should have the choice to donate resources only when it doesn’t cost them greatly. Note that during ‘peak-time’, when the majority of the nodes are quite active, their own money also gets drained quicker. So in such periods, they are forced to share their resources, even if they’re busy themselves; or they can choose to be active at another quieter period.

Incentive based approaches, as mentioned by [12], relies on the fact that people normally shy away from actively attacking other people maliciously. They’re more inclined to be passively dishonest. Moreover, in ad hoc networks where the protocols are implemented in a specialised embedded system, reverse engineering is a necessity for nodes to misbehave. This is not easy to do, and so people won’t be motivated to do it, unless there’s a personal gain. Maliciously attacking other nodes just to annoy them, although quite common in the Internet, does not give the attacker any benefit.

Having said that, there are cases where such malicious attack is conceivable. But they are usually performed to gain benefit outside the network protocol. Examples are impersonation, stealing sensitive data, and so on. They are the type of attacks that’s already ubiquitous on the Internet, and are outside the scope of this document, which reviews techniques that foster availability, or more specifically, cooperation. Intrusion Detection System, on the other hand, can handle this problem well.

Note that availability can also be threatened not only by selfish participants, but also by non-optimal path selection during network congestion. This is a point that is not addressed by all existing money-based incentive approaches. By adding a little bit more complexity in the pricing protocol,

money-based systems can have flexible pricing, where unpopular nodes can offer cheaper price, and handle non-critical network activities. This effectively balances the network load.

Any money-based systems are faced with the difficulty of storing and generating the money. What prevents a node from generating an unlimited amount of money? The *Terminode* project handles this by storing money in a tamper-resistant hardware module, while Lamparter, Paul and Westhoff [12]; and also the Sprite [11] project proposes a central, commonly trusted storage. Low protocol and processing overhead is also very critical, since money exchange needs to be performed for every hop of every packet transmission.

Money based system also needs to face problems similar to the real world currency. It needs to design unforgeable money, handle inflation, and prevent cheating (not fulfilling promise of performing the service, or giving the money).

3.2.1 Terminodes

Buttyán and Hubaux [9] are, to my knowledge, the first to propose the concept of using virtual money to stimulate cooperation. In their approach, each terminode (a hardware node in the Terminode [8] network), comes with an initial amount of *nuglet* (terminode's virtual currency). This nuglet is stored in a minimal, tamper-resistant security module in the hardware. The protocol in terminodes communicates with the *nuglet counter* in each node, such that every successful packet-forward decrements the originator's nuglet counter, and increments the forwarder's.

They offer two payment schemes, the *Packet Purse Model* (PPM), and the *Packet Trade Model* (PTM).

In the PPM, the originator of a packet compensates all forwarders. The originator injects the packet it wants to send with some amount of nuglets. Each intermediate node that helps forward the packet withdraws some amount of nuglet out from the packet, and increases its own nuglet counter. This goes on until the packet reaches the destination. If there's no more nuglet in the packet, and it hasn't reach the destination, it's dropped. On the other hand, if there's still nuglet left when the packet has reached the destination, those nuglets are discarded. It's apparent that this model requires good estimation of the number of hops necessary in each global transmission. Overestimation and underestimation leads to lost nuglet.

In the PTM, the cost of packet forwarding is covered by the destination node. The forwarder at each hop *buys* some amount of nuglet from the previous forwarder, which it then sells to the next forwarder. This goes on until the packet has reached the destination. At this point, all forwarders have increased their nuglet counter, and the destination has lost the same total amount of nuglets. This model improves on the previous one in that the originator doesn't need to calculate a good estimation on the number of nuglets required for a transmission. This model is suitable when the destination node is the one that requests the packet to be sent (e.g. a web server sending a page to a browser). Note that having the destination node pay for the packet, means that the previously mentioned flooding prevention advantage of money systems no longer works as mentioned. A workaround is to have the forwarder a little bit more intelligent. Since a forwarder node has to buy the packet, and sell it later at a higher price, it wants to be reasonably certain that its investment be returned. So flooders can be made unpopular among the forwarders. Although this prevents DoS, it's not as effective, and much more complicated (remember, reputation complicated and difficult to do right).

In their second paper [10], they model the optimal behaviour of nodes, given that they each have a nuglet counter, and limited battery. Each node needs to have enough nuglets to support its network activity, but it also needs to preserve its battery life. The model determines when a node should forward a packet, and when it should send its own, in order to optimise its own benefit in the network.

Terminode's reliance on a tamper resistant hardware module makes it unsuitable for deployment in community based networks, because it limits the manufacturing of the node devices. Relying on tamper resistant module for a critical module is also not very reliable, since tamper-proofness is very difficult to achieve, or maybe even impossible.

3.2.2 Sprite

Like the Terminode project, the nodes in the *Sprite* [11] system charges for the service of forwarding other nodes' packets. But unlike Terminode, Sprite doesn't rely on a trusted hardware. Instead, it relies on a commonly trusted *Credit Clearance Service* (CCS) node to keep every node's credit/money account. When a node forwards a packet, it creates a *receipt* from the packet. Although the receipt is derived from the message in the packet, it doesn't reveal the actual content of the message. The receipt can then be sent to the CCS to claim credit for the service performed, or stored locally for later claiming.

An interesting aspect of Sprite is that since there's a commonly trusted entity that keeps everybody's credit account, the Sprite system lets people increase their credit by paying real world money to the CCS admin.

The requirement of a commonly trusted node greatly limits the applicability of Sprite. Most ad hoc network architectures assume that every node are of the same level. In fact, one of the main advantage of ad hoc networking is the independence from infrastructures controlled by a an authority figure. Even in an architecture where commonly trusted node is expected, credit clearance can incur a large overhead given that the CCS node is centralised, and could be placed far away from the node who owns the receipt. Sprite tries to minimise the overhead by allowing the receipts to be stored and accumulated. Even so, the receipts need to be liquidated quite often, considering that the nodes do need them. Sending the receipts only when it's absolutely necessary will cause a lot of problems (e.g. unavailability of CCS, stressing further an already congested network, etc.).

3.2.3 Secure Charging Protocol in Ad Hoc Stub Networks

The virtual-money scheme described in [12] limits its usage to *Ad Hoc Stub Networks*. An ad hoc stub network is an ad hoc network that is connected to a gateway, commonly an Internet gateway. So in this architecture, there is a commonly trusted third party (the ISP) that's willing to provide gateways and access points to the network. As a result, a scheme similar to *Sprite* can be employed effectively here, where each node's money account is stored and controlled in the access points. Obviously, the charging system described in the paper is only relevant to a very specific architecture.

3.2.4 Priority Forwarding

In [13] Raghavan and Snoeren criticise existing currency-based incentive approaches. They observe that many nodes might not be able, or willing to participate in the currency system. They give an example where nodes on the edge on the network have very little request for forwarding packets, hence potentially poor. So they propose that standard best-effort forwarding should still be employed, along with an incentive system. Nodes that are able, and willing to forward packets receive money, and with it they can receive prioritised treatment; while nodes that are poor, or not willing to pay, can still receive best-effort treatment.

One problem the paper doesn't address is when an intermediate node drops a packet that's supposed to be forwarded, when the originator doesn't

offer any money. It also doesn't address any of the weaknesses of the current currency-based system.

4 Conclusion

Presently, there are no availability techniques that work effectively in generic ad hoc networks. Detection based approaches suffer from the fact that the most common technique used for detection is fallible not portable across all types of network. Detection also needs to be accompanied by punishment to discourage further misbehaviour, but existing punishment methods are also not accurate enough, and inherently complicated. Incentive based approaches handle the problem from a different point of view. Instead of detecting selfish behaviours, they make it so that it's more beneficial for nodes to cooperate. So there's no need to actively detect and punish selfish nodes. Unfortunately, existing incentive based approaches are not generic enough. Currently, all of them either rely on trusted hardware, or commonly trusted centralised node. Consequently, they are not desirable in a network like SAHN, where such reliance isn't feasible.

References

- [1] J. Li, C. Blake, D. S. De Couto, H. I. Lee, R. Morris. Capacity of ad hoc wireless networks. In *Proceedings of ACM MOBICOM*, pages 61-69, July 2001.
- [2] R. Heady, G. Luger, A. Maccabe, and M. Servilla. The architecture of a network level intrusion detection system. Technical report, Computer Science Department, University of New Mexico, August 1990.
- [3] Y. Zhang, W. Lee. Intrusion Detection in Wireless Ad-Hoc Networks. In *Proceedings of MOBICOM 2000*, pages 275-283, 2000.
- [4] S. Buchegger, J-Y. Le Boudec. The Selfish Node: Increasing Routing Security in Mobile Ad Hoc Networks. IBM Research Report, 3354, 2001.
- [5] S. Buchegger, J-Y. Le Boudec. Nodes Bearing Grudges: Towards Routing Security, Fairness, and Robustness in Mobile Ad Hoc Networks. In *Proceedings of the Tenth Euromicro Workshop on Parallel, Distributed and Network-based Processing*, pages 403-410.
- [6] S. Buchegger, J-Y. Le Boudec. Performance Analysis of the CONFIDANT Protocol. In *Proceedings of the 3rd ACM International Symposium on MobiHoc*, pages 226-236, June 2002.

- [7] S. Marti, T.J. Giuli, K. Lai, M. Baker. Mitigating Routing Misbehaviour in Mobile Ad Hoc Networks. In *Proceedings of MOBICOM 2000*, pages 255-265, 2000.
- [8] J-P. Hubaux, J-Y. Le Boudec, S. Giordano, M. Hamdi. The Terminode Project: toward mobile ad-hoc WANs. In *Proceedings of the Mobile Multimedia Conference, MOMUC*, San Diego, 1999.
- [9] L. Buttyán, J. P. Hubaux. Enforcing service availability in mobile ad-hoc WANs. In *IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing (MobiHOC)*, Boston, MA, August 2000.
- [10] L. Buttyán, J-P. Hubaux. Stimulating Cooperation in Self-Organizing Mobile Ad Hoc Networks. *ACM/Kluwer Mobile Networks and Applications (MONET)*, October 2003, Vol. 8 No. 5
- [11] S. Zhong, Y. R. Yang, J. Chen. Sprite: A Simple, Cheat-proof, Credit-based System for Mobile Ad Hoc Networks. Technical Report Yale/DCS/TR1235, Department of Computer Science, Yale University, July 2002.
- [12] B. Lamparter, K. Paul, D. Westhoff, Charging Support for Ad Hoc Stub Networks, *Journal of Computer Communication, Special Issue on Internet Pricing and Charging: Algorithms, Technology and Applications*, Elsevier Science, Summer 2003.
- [13] B. Raghavan, A. C. Snoeren. Priority Forwarding in Ad Hoc Networks with Self-Interested Parties. Workshop on *Economics of Peer-to-Peer Systems (P2PEcon '03)*, Berkeley, CA, June 2003.
- [14] P. Michiardi, R. Molva. CORE: A Collaborative Reputation Mechanism to enforce node cooperation in Mobile Ad Hoc Networks. Sixth *IFIP conference on security communications, and multimedia (CMS 2002)*, Portoroz, Slovenia, 2002.