CSR: Constrained Selfish Routing
in Ad-hoc Networks

Christine Bassem and Azer Bestavros

Computer Science Department, Boston University, Boston MA 02215, USA
{cbassem,best}@cs.bu.edu

Abstract. Routing protocols for ad-hoc networks assume that the nodes forming the network are either under a single authority, or else that they would be altruistically forwarding data for other nodes with no expectation of a return. These assumptions are unrealistic since in ad-hoc networks, nodes are likely to be autonomous and rational (selfish), and thus unwilling to help unless they have an incentive to do so. Providing such incentives is an important aspect that should be considered when designing ad-hoc routing protocols. In this paper, we propose a dynamic, decentralized routing protocol for ad-hoc networks that provides incentives in the form of payments to intermediate nodes used to forward data for others. In our Constrained Selfish Routing (CSR) protocol, game-theoretic approaches are used to calculate payments (incentives) that ensure both the truthfulness of participating nodes and the fairness of the CSR protocol. We show through simulations that CSR is an energy efficient protocol and that it provides lower communication overhead in the best and average cases compared to existing approaches.

1   Introduction

Motivation: The design and implementation of practical routing protocols for ad-hoc networks is still an open and challenging problem, whose solution is critical for the widespread deployment of the many distributed applications envisioned for ad-hoc networks.

Most of the ad-hoc routing protocols proposed in the current literature presume that nodes are cooperative and are always willing to contribute their own resources (*e.g.*, power, bandwidth, storage) in support of routing processes. In such settings, nodes are assumed to be truthful in the sense that intermediate nodes do not alter the content of forwarded packets and do not mischaracterize routing parameters so as to gain an advantage with respect to routing. However, in real settings, nodes of an ad-hoc network are under the control of individuals, who may not necessarily be cooperative. Indeed, such individuals are likely to be rational, selfish, or even malicious. Malicious nodes are those bent on disrupting the network functionality, whereas rational, selfish nodes are those that do not aim to disrupt the network, but are simply interested in maximizing the utility they beget from the network, even if doing so requires them to be untruthful.

Besides cooperative routing protocols, there have been several proposed routing protocols that provide incentives for nodes to help out in carrying the network load. Such incentives can be in the form of payment for cooperation [1–5] or in the form of punishment for non-cooperation [6, 7]. In this paper, we focus on

payment-based models, where nodes are rewarded for their help. In these models as well as in ours, a node would be willing to inform other nodes of its private (secret) costs for providing help in order to get paid, and a node would be willing to lie about its (or others') costs if this might lead to a higher payment.

Related Work: Routing protocols for ad-hoc networks vary greatly in design and in the assumptions they make about the network. Several protocols assume that the nodes in the network are selfless (altruistic) and are willing to help other nodes when such help is needed [8–11].

The Ad-hoc On-demand Distance Vector (AODV) protocol [8] and the Dynamic Source Routing protocol [9] are both reactive protocols, wherein routes are established on demand, thus reducing the communication overhead. In DSR, all the route information is kept in the control packets. This means that control packets grow larger as the route grows longer. This is in contrast to AODV where all route information is stored locally at the nodes and the control packet sizes remain constant. Therefore, AODV is largely considered a faster and less power-consuming protocol than DSR [10, 11].

As we alluded earlier, in realistic settings, nodes are expected to be selfish, and thus would not help unless incentivised to do so. Example mechanisms that punish non-cooperative nodes through the use of reputation-based protocols include the works described in [6, 7]. Example mechanisms that reward cooperative nodes include the works described in [1–5]. In [2] the intermediate nodes are paid with a virtual currency called NUGLETS. In [5], Sprite is proposed which provides incentive – in a game theoretic sense – for mobile nodes to cooperate and report actions honestly.

In Ad-hoc VCG [1], the route discovery process is based on the DSR [9] protocol where the source node floods a request packet to all of its neighbors looking for a path to the destination. Each intermediate node appends to the request packet its own costs to send data. This information is forwarded to its neighbors and so on until the request reaches the destination node. The destination node collects all the requests flooded through the network and integrates them to build a complete logical view of the network. Using this information, the destination chooses the most cost-efficient path and uses a variation of the VCG model (named after Vickrey [12], Clarke [13] and Groves [14]) to calculate the payments to be given to each intermediate node in that path. Using the VCG payment model, the destination node calculates for each intermediate node (i) in the most cost efficient path (SP) the cost of the second most-cost-efficient path without that node (SP^{-i}). As the intermediate node cooperates in delivering data from the source to the destination, it receives a payment to cover its costs plus a premium to ensure that the nodes would not lie about their secret costs. This premium is the difference between the costs of SP and SP^{-i}.

Paper Contributions and Organization: As mentioned above, the main disadvantage of DSR when compared to AODV is the increased communication overhead. In contrast, the Constrained Selfish Routing (CSR) protocol that we propose in this paper uses the Ad-hoc On demand Distance Vector (AODV) protocol [8] to decrease the communication overhead thus decreasing the power

consumption at the nodes. In that respect, we design a mechanism that provides the most cost efficient path between a pair nodes in the network. It is a power efficient protocol that ensures the truthfulness of the nodes participating in it. We describe the system model in section 2, followed by a detailed description of the proposed Constrained Selfish Routing protocol in section 3. In section 4, we analyze the protocol's truthfulness properties and its overhead. In section 5, we evaluate the protocol's performance through simulations and provide results that support in an empirical setting the analytical results presented in section 4. Finally in section 6, we conclude the paper with a summary of our contributions.

2   System Model and Assumptions

We augment the model used by Anderegg and Eidenberz in [1] where the network is represented as a graph $G = (V, E, w)$ with the set of vertices V that represent the mobile nodes in the network and the set of directed edges E which represents the unidirectional links between the nodes in the network. The weight function $w : E- > R$ for each edge (i, j) represents the weight of the link between the node i and the node j, which is the cost of transmitting a packet from i to j.

Each node has a unique identifier i and has an individual cost of energy parameter c_i. The cost of energy c_i of a node i is its private type, *i.e.*, only i knows its true value, which is a measure of the level of inconvenience the node faces when asked to forward a packet. One of the factors affecting this measure is the level of difficulty the node faces in order to recharge its power. The payment that a node receives for helping others is proportional to its cost of energy c_i; therefore, a node might lie about its true value of c_i if such a lie would increase its payment.

All nodes use omni-directional antennas for communication, *i.e.*, when a node sends a signal carrying a packet, all neighboring nodes in its transmission range receive that packet. We assume that nodes can control their signal emission power; as the node increases its emission power, its transmission range increases, and as a result more neighboring nodes receive the packets sent and vice versa. When a node i uses an emission power P_i^{emit} to send a packet, the node j at distance d from i receives the signal with power given by

$$P_{i,j}^{rec} = K \times \frac{P_i^{emit}}{d^\alpha} (1)$$

where K is a constant and α is the distance power gradient, another constant that ranges between 2 and 6 depending on the network conditions. A node successfully receives a signal if the power of the received signal is above a certain acceptance threshold. While the acceptance threshold might differ from one node to another in the network, for simplicity, we assume that all nodes in the network agree on the same value for the acceptance threshold P_{min}^{rec}. If the power of the signal received exceeds P_{min}^{rec}, then j successfully receives the packet carried by the signal. However, the original emission power used by node i (P_i^{emit}) might be *overvalued* and less emission power could have been used to successfully send a packet to j. Node j can calculate the minimum emission power that the node i

would need to use to send a packet successfully to j using the following formula,

$$P_{i,j}^{min} = \frac{P_i^{emit}.P_{min}^{rec}}{P_{i,j}^{rec}}(2)$$

Once calculated, this value is sent back to i and later, i uses this value as its default emission power – if it needs to transmit a packet to j in the future.

The total weight of a link (i, j) is the product of the cost of energy c_i and the minimum emission power that i uses to send a packet to j.

$$w(i, j) = c_i \times P_{i,j}^{min}(3)$$

As previously mentioned, in the proposed protocol nodes will be paid to forward data for others. In [1], the authors propose two payment models; either the source node is responsible for paying all intermediate nodes or some central authority – a "bank" – holds accounts for all nodes in the network and is responsible for all transactions performed on them. Any node can communicate with the bank if it is in its communication range. If the bank is inaccessible, a node is allowed to store the transaction information locally. This information is relayed to the bank as soon as the bank becomes accessible again in the future. In Ad-Hoc VCG, the authors assume that the bank will deduct the payments given to cooperating nodes from the accounts of all the nodes in the network. In this paper, we assume that the destination node is responsible for calculating the payments for each of the helping nodes and it sends the payment information to the bank. The bank credits the accounts of all the intermediate nodes and debits the accounts of the source and/or destination nodes.

3   Constrained Selfish Routing Protocol

The Constrained Selfish Routing protocol is an on-demand routing protocol for ad-hoc networks. The VCG payment model is used to provide incentives for nodes to help out other nodes in the network as used in [1].

3.1   Overview

CSR consists of three components: route discovery, data transmission and route recovery. When a node S needs to sends data to D but D is not in the transmission range of S, route discovery begins. In route discovery, S floods the network with a request to find a path to D and payments for the intermediate nodes are calculated. The route discovery phase will be explained in more details later in this section.

During data transmission, after the path from S to D is known, the data is sent between them and intermediate nodes get paid for forwarding the data. Whenever the destination node successfully receives a data packet, it keeps track of how much it should pay each intermediate node and when possible it notifies the bank of the payments.

Route recovery is activated when links between nodes are broken during data transmission. If the next hop node is not available for any reason (such as node failures, link failures, or node mobility), the node that detects the failure sends an ERROR packet back to the source node. Upon receipt of such an error report, the source node starts the route discovery all over again.

3.2 Route Discovery

The route discovery in CSR is adapted from the AODV routing protocol, which is an on demand ad-hoc routing protocol proposed by Perkins and Royer in [8]. AODV provides several advantages in ad-hoc networks such as low communication overhead and less power consumption. In CSR, the route discovery phase is divided into two separate phases: the first phase is the actual discovery phase when the most cost-efficient path between the source and destination is found. The second phase is when the payment calculation occurs. Nodes do not have to wait for the termination of this phase to start data transmission as it can start on the completion of the first phase. The second phase can be performed offline and at any time after the first phase is completed or during data transmission.

Phase 1: Finding a route. The first phase is similar to the route discovery phase in AODV [8] where a source node S floods the network with its request to a destination node D. Then S has to wait for D to send out a reply with the most cost efficient path between them to start data transmission.

Packets Used: Two types of packets are used in this phase; the REQUEST and the REPLY packets.

The REQUEST packet is the packet that floods the network. In addition to the typical ID and addressing fields that constitute the *Packet header*, this packet contains a *Cost of Energy* field and an *Emission Power* field which indicates the cost of energy and emission power of the inner source when sending the current instance of the packet and a *Total Weight* field which represents the total weight of the path from the source node to the inner source node.

The REPLY packet is sent out by D and it contains information about the most cost-efficient path between S and D. It is only forwarded by the nodes in the best path between S and D. It contains the Packet Header, the original Request ID and the List of Costs, a list that includes the ID, cost of energy and the minimum power of each intermediate node along the chosen path.

Data Structures Used: During route discovery, each node maintains two data structures to help out in the route discovery phase. The *Best Route Cache* is used to store the best REQUEST packet received for a specific request between a pair of nodes and the *Neighbor Cache* is used to keep track of which of the nodes neighbors can provide a path to the source node.

Details of Phase 1: When a source node S wishes to find a path to a destination D, it prepares the initial REQUEST packet with its cost of energy and emission power and floods it to its neighbors.

When a node j receives a REQUEST packet from neighbor i it follows Algorithm 1 to decide whether to drop the packet or forward it. The weight in the packet is updated and forwarded if the packet carries a better weight than the best weight stored in the Best Route Cache. When the destination node receives the first REQUEST from the source node, it follows the same algorithm but without forwarding the packet with the best weight, and keeps listening for other route REQUEST packets in case a better REQUEST is received from the same source node. If a better REQUEST is received, the corresponding entry in the best route cache is updated. After a timeout period expires, the destination

extracts the best REQUEST it received from neighbor i from the Best Route Cache and sends out a REPLY packet with $\{c(i), Pmin(i, D)\}$ to the source node through i.

Algorithm 1 Actions performed by a node j upon receiving a REQUEST packet from node i.

```
if REQUEST has no new information then
    drop REQUEST
else
    Add neighbor i to the Neighbor Cache
    Calculate Pmin(i,j) = Pemit(i) * Prec(min) / Prec(i,j)
    Calculate w = total weight of path from S to j through i
    best = total weight of the best REQUEST stored in the Best Route Cache
    if best is null then
        Replace Pemit(i) with Pmin(i,j) in REQUEST
        Add REQUEST to the Best Route Cache
        Forward REQUEST packet with c(j) and Pemit(j)
    else
        if w < best then
            Replace Pemit(i) with Pmin(i,j) in REQUEST
            Replace entry in the Best Route Cache with the current REQUEST
            Forward REQUEST packet with c(j) and Pemit(j)
        end
    end
end
```

Each intermediate node i receiving the REPLY packet from node j will extract the corresponding best REQUEST packet from the the Best Route Cache and extract the inner source node k from that packet. Then j will add $\{c(k), Pmin(k, i)\}$ to the list of costs in the original REPLY packet and forward it to k.

Once the source node receives the REPLY packet, it can enter the data transmission phase and does not have to wait for the second phase to end.

Phase 2: Payments Calculation. While route discovery allows the destination D to identify the best (most cost-effective) path, D is unable to calculate the payments to the intermediate nodes along that path. As we indicated earlier, in CSR these payments are based on a VCG model, requiring the calculation of the second-best path when each node on the best path is excluded. This process is carried out in Phase 2 of the CSR protocol. This phase is similar to the route discovery performed in phase 1, except that it is done in the opposite direction (from D to S) and the request packets have constraints to not include certain nodes in the discovery.

To start phase 2, S sends to D information about the intermediate nodes in the best path between them. This is done by piggy-backing a CONFIRM packet to the first DATA packet send to D.[1]

Packets Used: We introduce 3 types of packets in this phase, namely, the CONFIRM, FIND and FOUND packets. The CONFIRM packet is piggy-backed with the first DATA packet sent by the source node S. It is used to inform the destination node D about the details of the intermediate nodes in the chosen shortest path (SP) between S and D. There is only one field in the CONFIRM packet; the list of costs that were obtained from the REPLY packet in phase 1. When sending a DATA packet, the source node S signs and appends this list to the packet.

The FIND packet is used to discover whether a path between two nodes without a certain intermediate node. Its fields are the *Packet Header* with multiple inner destinations, the *Excluded Node*, the original *Request ID* and the *Hop Count* which is used to calculate the time out period the node will spend before sending the FOUND packet.

The reply to the FIND packet is the FOUND packet which is used by nodes in the second phase to notify the originator of the FIND packet that a path is indeed found excluding a specific node and it also carries the total weight of that found path. Its fields are the *Packet Header* with multiple inner destinations, the *Excluded Node*, original *Request ID*, *Total Weight* of the path and the *Cost of Energy* and *Emission Power* of the node sending the FOUND packet.

Data Structures Used: During phase 2, each node maintains similar data structures as those kept in phase 1 but for the FIND and FOUND packets.

The first data structure is the *Find Neighbor Cache* which is used to track which neighbors sent similar FIND packets to the node. It is used to keep track of which neighbors have previously sent a FIND packet, to guarantee that only one FIND and one FOUND packets are sent for each request. The second data structure is the *Best Found Cache* which is used to store the best FOUND packet received so far for each corresponding FIND packet.

Details of Phase: Once the destination node D receives the CONFIRM packet, it extracts the list of intermediate nodes in the shortest path between the source and the destination (SP). For each intermediate node i, D will search for a best path from S to D without that intermediate node (SP^{-i}).

The destination extracts from its Neighbor Cache the set of neighbors N that have paths to the source node and for each intermediate node i obtained from the CONFIRM packet, D will send a FIND packet with Hop Count of 1 excluding i to be sent to all the nodes in the set $N - i$. Then D starts a timer to indicate the end of phase 2.

When an intermediate node i receives a FIND packet from j, it drops the packet if it had been received before. Otherwise, it forwards the packet with an incremented Hop Count to all the nodes in the set of neighbors obtained from

---

[1] As mentioned above, this phase can be performed offline and data transmission does not depend on it.

the Neighbor Cache except the excluded node. Then,node i starts a timer with a value inversely proportional to the Hop Count in the FIND packet.

When the source node receives a FIND packet with itself in the Destination field, it will send a FOUND packet to the Inner source of the FIND packet.

During the timeout period, when a node j (including the destination node D) receives a FOUND packet from node i, it updates the total weight of the packet and then decides to store the packet in the Best Found Cache if it's the first FOUND packet received for a corresponding FIND or it's the FOUND packet with the better total weight received so far.

When the timer at node i stops, i will extract the best FOUND packet received from the Best Found Cache and extract the set of neighbors from the Find Neighbor Cache. Then the node updates the fields in the FOUND packet and sends it to all the neighbors in list extracted from the Find Neighbor Cache and clears its caches.

When the D finishes its timeout period, it takes the information in the Best Found Cache and calculates the payments that should be made to each intermediate node using the following formula:
$$M(i) = |SP^{-i}| - |SP| + c_i * P_{i,j}^{min} (4)$$

The value $|SP^{-i}|$ is obtained from the FOUND packets in the Best Found Cache and the values $|SP|$, c_i and $P_{i,j}^{min}$ are obtained from the CONFIRM packet received at the beginning of the phase.

4 Analysis

In CSR, since nodes only forward requests with better total weights, we guarantee that the most cost efficient route is always chosen in phase 1 in the route discovery. As for the incentives provided to the nodes in the network, our VCG-based payment model guarantees the truthfulness of all the nodes in the network.

Theorem 1. *CSR guarantees the truthfulness of all the nodes in the network.*
Proof. The payment of each node i is $M(i) = |SP^{-i}| - |SP| + c_i * P_{i,j}^{min}$ which is its utility. As mentioned before, node i may be untruthful (*i.e.*, lie about its type) in order to increase its utility. Let us consider the various possible ways that an intermediate node may be untruthful.

1. The node may lie about the true value of c_i or P_i^{emit} in phases 1 or 2.
2. The node may lie about the computed value $P_{i,j}^{min}$ in phases 1 or 2.
3. The node may change any entry in any packet when forwarding it in phases 1 or 2.

We prove that in CSR, intermediate nodes will be truthful in all of the previously mentioned cases. The complete proof can be found in [15].

A very important aspect of an ad-hoc routing protocol is its communication overhead because it affects the performance of the network and the power consumption at the nodes. We prove that CSR has a linear lower bound on the number packets sent and that the (worst-case) upper bound on the number of overhead packets is equal to that of the Ad-Hoc VCG protocol [1]. The detailed proofs of the theorems below can be found in [15].

Theorem 2. *The lower bound on the overhead of CSR in terms of the number of packets sent is $O(n)$.*

Theorem 3. *An upper bound on the overhead of CSR in terms of the number of packets sent is $O(n^4)$.*

5   Simulation Results

An event-based simulator was designed to simulate ad-hoc networks and to test the performance of various routing protocols. Using this simulator, the performance of CSR is compared to Ad-Hoc VCG [1]. The simulations were run in a closed environment in which any number of simulated nodes move freely.

The mobility model used in the simulator is adopted from the Random Walk model [16]. Initially, each node picks a random direction (Θ) taken from a uniform distribution on the interval $[0, 2\Pi]$. The node moves in the chosen direction for a fixed period (default is 4 seconds) and then pauses for another fixed period of time (a control variable). Next, the node picks a new random direction (Θ') taken from a uniform distribution on the interval $[\Theta - 20, \Theta + 20]$ and repeats the same process. The reason for choosing the interval $[\Theta - 20, \Theta + 20]$ when deciding Θ' is to simulate a realistic motion.

Applications on the simulated wireless nodes are designed to send data to random destinations (picked uniformly at random) at random times, with inter-messaging time picked from an exponential distribution with a mean of 10. Once a destination is chosen, the route discovery of the simulated protocol starts.

Four sets of simulations were performed to measure the efficiency of CSR in various conditions. In all simulations, the environment is $500m \times 500m$ in size, the nodes' transmission range is $100m$ and the total simulation time is $100sec$. The measured performance parameters are the *Communication overhead* which is represented in two forms: the total number of packets sent out during the route discovery phase, the total number of bytes sent out during the route discovery phase, and the *Power consumption rate* which measures the average rate of power consumed by the nodes during the whole simulation.

In the first set of simulations, the number of nodes in the network is varied from 8 to 12 devices, the pause time of each node is 6 sec and the total number of requests made in each simulations is 5 requests. The performance evaluation of the protocols is shown in Figure 1, Figure 2 . Each value in the graph represent an average of 5 simulation runs with different seeds within a 90th-percentile confidence interval.

The results show that as the network size grows larger, the communication overhead in CSR exhibits a nearly linear growth while Ad-Hoc VCG shows a super-linear growth. This is anticipated from the analytical results (see Theorem 2): CSR has a nearly linear communication overhead in the average case. Moreover, the results prove that CSR is more power efficient than Ad-Hoc VCG because of the less communication overhead of CSR.

The results of the other sets of simulations could be found in [15].
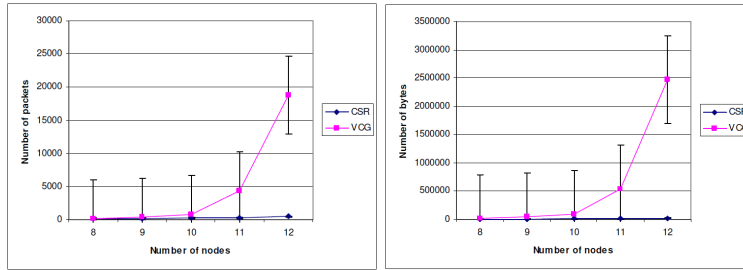
((a)) Packets overhead   ((b)) Bytes overhead

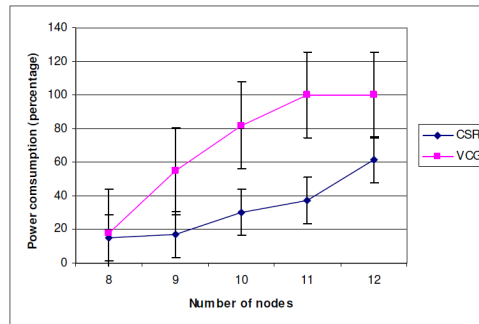Fig. 1: Communication overhead as the network size increases. CSR shows an improved performance over Ad-Hoc VCG.

Fig. 2: Power consumption rate as the network size grows larger. CSR is more power efficient that Ad-Hoc VCG.

6   Conclusion

The CSR protocol is an incentive-based routing protocol which provides incentives for selfish nodes in the network in a game theoretic setting. In this paper, we have shown CSR to induce thruthful node behavior through the use of a VCG-based model for calculation of payments to intermediate nodes. With truthfulness guaranteed, CSR provides the most cost efficient path between any pair of nodes in an ad-hoc network with a linear lower bound of $O(n)$ on the communication overhead, where n is the number of nodes in the network. The lower communication overhead also guarantees lower power consumption.

References

1. Anderegg, L., Eidenbenz, S.: Ad hoc-vcg: a truthful and cost-efficient routing protocol for mobile ad hoc networks with selfish agents. In: MobiCom '03. (2003) 245–259

2. Buttyan, L., Hubaux, J.: Nuglets: a virtual currency to stimulate cooperation in self-organized ad hoc networks. Technical Report DSC/2001 (2001)
3. Feigenbaum, J., Papadimitriou, C., Sami, R., Shenker, S.: A bgp-based mechanism for lowest-cost routing. Distrib. Comput. **18**(1) (2005) 61–72
4. Zhong, S., Li, L.E., Liu, Y.G., Yang, Y.R.: On designing incentive-compatible routing and forwarding protocols in wireless ad-hoc networks: an integrated approach using game theoretic and cryptographic techniques. Wirel. Netw. **13**(6) (2007) 799–816
5. Zhong, S.: Sprite: A simple, cheat-proof, credit-based system for mobile ad-hoc networks. In: In Proceedings of IEEE INFOCOM. (2003) 1987–1997
6. Buchegger, S., Le Boudec, J.Y.: Performance analysis of the confidant protocol. In: MobiHoc '02. (2002) 226–236
7. Buchegger, S., Boudec, J.Y.L.: Nodes bearing grudges: towards routing security, fairness, and robustness in mobile ad hoc networks. In: Proceedings. 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing. (2002)
8. Perkins, C.E., Royer, E.M.: Ad-hoc on-demand distance vector routing. In: In Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications. (1999) 90–100
9. Johnson, D.B., Maltz, D.A.: Dynamic source routing in ad hoc wireless networks. In: Mobile Computing, Kluwer Academic Publishers (1996) 153–181
10. Misra, P.: Routing protocols for ad hoc mobile wireless networks. Technical report, Ohio State University (1999)
11. Hong, X., Xu, K., Gerla, M.: Scalable routing protocols for mobile ad hoc networks. IEEE Network **16**(4) (July/August 2002) 11–21
12. Vickrey, W. The Journal of Finance (1) 8–37
13. Clarke, E.H.: Multipart pricing of public goods. Public Choice **11**(1) (September 1971) 17–33
14. Groves, T.: Incentives in teams. Econometrica **41**(4) (July 1973) 617–31
15. Bassem, C., Bestavros, A.: Csr: Constrained selfish routing in ad-hoc networks. Technical Report BUCS-TR-2009-018, Computer Science Department, Boston University (2009)
16. Camp, T., Boleng, J., Davies, V.: A survey of mobility models for ad hoc network research. Wireless Communications Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications **2** (2002) 483–502