Detecting Selfish Behavior in a Cooperative Commons

Hyun Jin Kim and Jon M. Peha
Carnegie Mellon University
{hyunjin, peha}@cmu.edu

Abstract

A cooperative commons is a type of ad hoc network in which all devices are required to communicate and carry each other's traffic, even when devices are associated with different administrative domains. Thus, infrastructure is constructed at little cost to each owner. One unusual feature of the cooperative commons is that as more devices join, total communication capacity increases. These advantages are possible when devices are willing to cooperate and use their own resources to carry traffic of others, but are undermined by *selfish* behavior, where a device's actions increases benefit for that device while decreasing the average benefit for all devices. This paper demonstrates that selfish behavior cannot be detected with the prominent routing protocols currently used in ad hoc networks, and proposes a novel approach that includes use of routing protocols in which selfish behavior cannot be concealed, and watchdog algorithms that observe behavior of neighbors for signs of selfishness. We prove that our approach reliably detects all acts of selfishness by individual devices in a network where devices are fixed and there are no packet collisions. We demonstrate that our watchdog algorithms work with a general class of routing protocols, and show how existing routing protocols can be extended to fit in that class.

1. Introduction

If it proves to be viable, the *cooperative commons* could provide a valuable and radically different approach to the deployment of wireless infrastructure and the management of spectrum [1-3]. A block of spectrum could be allocated for shared use by any and all wireless devices. Unlike today's unlicensed spectrum bands, all devices in the cooperative commons would be able to communicate with each other using an established protocol, and would be required to cooperate with each other.

A cooperative commons could be established in a number of ways. First, a regulator or license-holder may allocate a spectrum band in which a device can only be deployed if it cooperates as part of a cooperative commons [1, 2]. Second, a cooperative commons may form in today's unlicensed bands, if enough people choose to participate, although with this approach, the cooperative commons must contend with some unlicensed devices that are not part of the commons. A number of commercial companies, such as Meraki in San Francisco [10] and non-profit organizations, such as NetEquality Equal Access Community Internet [5], and the Champaign-Urbana Community Wireless Network [6], have pursued this approach [7]. Third, devices in a cooperative commons may share spectrum with the existing licensed primary spectrum user, and operate as secondary devices [1, 2]; secondary devices may cooperate both to avoid harmful interference to the primary, and to carry each other's traffic. Many researchers have suggested this approach using cognitive radio system (e.g. [8, 9]).

In any one of the above scenarios, the end devices form their own shared infrastructure, fundamentally changing the cost of large-scale deployment. For example, this may make a wireless metropolitan-area network financially sustainable where it would not be sustainable today [10]. Moreover, this can lead to far greater spectral efficiency [3, 11], perhaps alleviating the shortage of available spectrum [3]. However, there are serious security challenges associated with the cooperative commons [1, 2], some of which are addressed in this paper.

A cooperative commons is one type of ad hoc network without pre-existing infrastructure, but with additional technical challenges that are not present in typical ad hoc networks. First, because any device is allowed to join the cooperative commons, devices must cooperate with and carry traffic for other devices, even though these devices do not serve the same administrative domain [1, 2]. This leads to many challenges. For example, no protocol can be adopted for which it is assumed that cooperating devices have a trust relationship. Second, there is no limit in the number of devices that may join a cooperative commons. Thus, protocols must remain efficient even if the network grows large. The potential for large size also has benefits. One unusual advantage of the cooperative commons is that as more devices are added, total system capacity increases. This occurs because increasing the number of devices may decrease the mean distance between devices, so that devices can reduce transmit power and increase frequency reuse [3, 11]. Consequently, communications capacity increases, which is referred to as *cooperative gain* [1-3].

Because devices in a cooperative commons cooperate and carry each other's traffic, these devices must be willing to use their own battery power and delay their own packets in order to forward packets for unaffiliated devices outside their administrative domain. However, there may be devices that exhibit *selfish* behavior, which is behavior that increases the benefit for this device, but decreases the average benefit for all devices, by some reasonable measure of benefit. For example, consider a network in which a device establishes a path with routing packets before sending data packets. An effective selfish behavior would be to drop these routing packets or forward with a time-to-live (TTL) of 0 so that no paths can be established. A device could thereby avoid forwarding many subsequent data packets. Another selfish behavior would be to make paths that include the selfishly behaving device seem longer than they really are, perhaps by artificially increasing hop counts so the sources are more likely to choose another routes that appear to be shorter. The selfish acts above are even more problematic when combined with selfish manipulation of transmit power. Often, part of detecting selfish behavior is requiring devices to watch the transmissions of their neighbors [16-23, 26]. When devices know that their behavior is observed by neighbors, they may still suppress routing packets selfishly and evade detection by transmitting at a power large enough to be seen by the watchdogs, but too small to be received by the nominal recipient.

In a cooperative commons, there may be devices that behave maliciously to disrupt and damage normal network operation, a topic that is also being studied [12]. This paper addresses selfish behavior which has received less attention.

Our approach detects selfish behavior that involves routing protocol packets but not data packets because devices can benefit greatly from selfish behavior with routing packets, but not with data packets. When Device A behaves selfishly and drops a data packet, Device A may benefit at that moment by avoiding a transmission. However, the dropped data packet is likely to be retransmitted to Device A, possibly causing a collision that prevents Device A from receiving a packet it wants. Moreover, if Device A selfishly drops data packets but handles routing packets in accordance with the protocol, then the source may repeatedly attempt to reestablish a route through Device A, which will create additional packets that Device A must carry. Therefore, although selfish behavior in dealing with data packets may benefit a device in the short term, it ends up wasting resources in the long term. On the other hand, manipulating even a small number of routing protocol packets can provide a big gain because a device may be able to avoid subsequent transmission of a potentially large number of data packets for others.

This paper presents a novel approach that makes it possible for all devices behaving selfishly to be detected by one or more neighbors. We require each Device A to observe the behavior of its neighbors even when those neighbors are transmitting to devices *other than Device A*, and to watch for signs of selfishness. *No approach* based on this can succeed when steerable antennae can change transmission direction from one packet to the next, because if a device can send entirely different content to each of its neighbors, then a wathdog device can only observe packets sent to it deliberately. However, as long as there are no steerable antennae such that a device that increases the power it radiates in one direction by *x*% also increases the power in all directions by *x*%, it is possible to detect selfish behavior.

Similarly, when neighboring devices violate the protocol in a coordinated or complementary way, *no approach* can succeed in detecting all forms of selfish behavior. Consider the case where Device A and all of its neighbors collude to avoid expending resources on devices outside the group. Device A can drop all routing packets that attempt to establish a path unless the source or destination of that path is among the device's partners in collusion. The only devices that can observe Device A's violations of the protocol also benefit from those violations. Vulnerability to collusion is an inherent and potentially problematic property of a cooperative commons. However, any device can join a cooperative commons, so no device can be assured of having neighbors willing to collude. In this paper, we consider the case where few devices are willing to collude selfishly. We will show that if a device behaves selfishly and its neighbors follow the prescribed protocol and watchdog algorithms, then one or more of those neighbors will detect the selfish behavior. Moreover, while we will assume that multiple devices do not collude, we will allow for the possibility that a single device may create multiple identities through a Sybil attack [29]; selfish behavior can be detected even if those collocated identities collude.

Section 2 presents reviews previous work on detecting selfish behavior in ad hoc networks, and explains why no existing approach can detect all selfish behaviors described above in the context of a cooperative commons. Our approach requires use of routing protocols with characteristics that are generally not present today. Section 3 states our requirements for routing protocols, and how some prominent routing protocols can be extended to meet these requirements. Section 4 describes how selfish behavior is detected and proves that all selfish acts are detected in an environment in which devices are fixed, and packets are not lost. Although it is beyond the scope of this paper, these mechanisms can be extended to support a more realistic environment that includes mobility and packet loss. We conclude this paper in Section 5.

2. Related Work

Many approaches [16-27] have been suggested to detect selfish behavior in different kinds of ad hoc networks, although not necessarily for a cooperative commons. In some approaches, devices determine whether a neighbor is selfish based entirely on what they observe (watchdog-based schemes). In other approaches, devices make decisions in part on what they are told by others (which would include acknowledgement-based schemes). While many previously proposed approaches may be effective in the networks for which they were intended, none detects selfish behavior in the unique context of a cooperative commons.

In a number of approaches [16-24], a device that selfishly drops packets is detected as follows. After transmitting a packet that must be forwarded, a device observes its neighbors. Failure to forward the packet is identified as selfish behavior, but all these watchdog-based schemes mis-detect devices that are cooperative as selfish in several common situations.

One cause for these problems is duplicate suppression, which is generally needed in ad hoc networks. When a packet is flooded, many devices receive multiple duplicates (although duplicates can differ in fields that change en route like hop count). With duplicate suppression, each node forwards only one. Duplicate suppression vastly decreases the overhead of flooding, which is an important feature of routing protocols typically used in ad hoc networks such as Dynamic Source Routing (DSR) [13] and Ad-hoc On-demand Distance Vector (AODV) [14]. However, when a device does not forward a routing packet because that device has already forwarded a duplicate, the watchdog schemes that have previously been proposed [16-24] would misidentify this as selfish behavior.

The occasional false positive might be tolerable if it were rare, as devices might then reasonably assume that neighbors that are only rarely seen acting selfishly are really cooperating. However, the false positives described above may be a common occurrence for some unfortunate devices. Consider the topology where a source floods a routing packet, and both Devices A and B re-flood the packet. Device C, that is a neighbor of both Devices A and B, forwards the packet from Device A, but suppresses the duplicate from Device B. Thus, Device B concludes that Device C is acting selfishly. Every time the source floods a new packet, Device C will be seen as behaving selfishly by either Devices A or B, depending on whose packet Device C receives first. Moreover, the same problem will occur in any topology with a loop.

When duplicate suppression is used, watchdog-based detection approaches are also vulnerable to a timing attack. Typically, routing protocols that suppress duplicate packets, such as DSR [13] and AODV [14], allow a device to suppress any later legitimate packets as long as one is forwarded, even if the duplicates arriving later provide better paths. Thus, a device can wait to receive several duplicates, and only forward the packet with the longest path, reducing the chance that a route through this device will be selected. Current watchdog approaches would not detect this.

As explained in Section 1, a device can also use power control to evade detection by watchdog schemes, and the previously proposed watchdog schemes [16-24] have no way to detect selfish behavior when this technique is used.

In some approaches, devices rely in part on information they do not observe themselves. This category includes the approach based on two-hop acknowledgements [26-28]. In this approach, devices observe their neighbors for possible selfish behavior similar to the watchdog-based approaches. When Device A becomes suspicious of its neighbor Device B, Device A transmits a packet through Device B to Device C with a request for an explicit 2-hop acknowledgement from Device C. When Device C successfully receives the packet with the request, Device C sends an acknowledgement to Device A after cryptographically hashing [26-27] or signing [28] the packet. Device A concludes that Device B has forwarded the packet when Device A receives the acknowledgement from Device C.

The above detection approach requires a security association between any pair of devices, and this is fine for the context for which it was developed. However, trusting other devices does not work without identity verification of devices due to the possibility of a Sybil Attack [23]; to lengthen a route, a device may use multiple identities. Unless it is guaranteed that a device is only assigned a unique identity, i.e. with tamper-proof hardware, approaches like this will not work in a cooperative commons.

In addition to the authentication challenges in a cooperative commons, approaches that are based on trusting other devices suffer from another problem; devices may intentionally lie. For example, devices using the two-hop acknowledgement scheme may choose not to send acknowledgements so their neighbors appear to be acting selfishly.

3. Routing Protocol Features

Although known approaches cannot reliably detect selfishness in a cooperative commons due to complications like duplicate suppression and power control, we can make selfish behavior detectable by requiring that the underlying routing protocol has features that are not found in current protocols. In this section, we propose a set of features. Section 4 will show how selfish behavior involving routing protocols with these features can always be detected.

Path lengths are measured in hops. Where information on path length is equally recent, we define the best path as the one with the minimum hop count. We allow but do not require the routing protocol to favor paths with more recent information about length. For example, the path with a higher "sequence number" takes higher precedence in AODV [14], regardless of hop counts. Duplicate suppression is allowed, but a duplicate cannot be suppressed if it announces a better path than was previously known.

Every device maintains the addresses of the two-hop neighbors that are on the best paths to reach destinations in their routing tables. With this field, devices can check if packets travel on correct paths. If the routing table stores full source routed paths, this field is not needed.

Power control has been suggested for energy-efficient routing in ad hoc networks [30]. In protocols such as DSR [13], AODV [14], and Destination Sequenced Distance Vector (DSDV) [15], a device uses maximum power to transmit packets without specified recipients, but on packets with specified recipients, power is reduced to the minimum level that is sufficient to reach the intended recipient(s). In order to prevent the selfish attack with transmission power control as described in Section 1, we require devices to transmit all routing packets at maximum power.

Our mechanism detects selfishness for packet types that fall in any of three categories. Definitions of these three categories and the algorithm that indicates how devices handle packets of each category are described in Sections 3.1 to 3.3. Section 3.4 describes how minor changes to a number of existing protocols can make all of their packet types fall within these three categories.

3.1 GENERAL ANNOUNCEMENT

Packets in the first category, which we call GENERAL ANNOUNCEMENTs, are flooded through the network to announce the distance to the device that initiated the flood. In some protocols, GENERAL ANNOUNCEMENTs are targeted to a specific destination to find the shortest path. . With minor changes, Routes Request packets (RREQ) in DSR [13] and AODV [14], and Update messages in DSDV [15] are prominent examples of this category.

GENERAL ANNOUNCEMENTs must contain three fields that are common to routing protocols: the addresses of the *source* and the *destination*, and the distance to the source. Distance can be represented by hop count, or any field from which hop count can be derived (i.e. full source-routed path in DSR [13]). The source that floods the GENERAL ANNOUNCEMENT may set a TTL, but it is not required. These packets also contain atypical fields. One is the address of the device that is the next hop to the source, which we call *next_to_source*. The 1-bit *duplicate_flag* indicates whether the packet is a duplicate. Sending a duplicate packet with the flag set allows a watchdog to verify that the packet was not selfishly dropped. A device that receives a GENERAL ANNOUNCEMENT with this bit set is required to forward the packet only if it announces a path to the source that is better than the best path that the receiving device knew previously. Since this is often not the case, the packet can be dropped, without risk that the behavior will be identified as selfish. Other than fields cited above (*duplicate_flag, next_to_source, path traveled, hop_ count, and TTL)*, we assume that fields in a GENERAL ANNOUNCEMENT remain unchanged as the packet is forwarded through the network.

Every device promiscuously listens to packet transmissions, and here is how a device acts after receiving a GENERAL ANNOUNCEMENT. If the packet has a TTL that has expired, no action is required and the device may drop the packet. If there is no TTL that has expired, and the duplicate_flag is set, then the device is required to respond if and only if the packet announces a path to the source that is better than the best path that the device knew previously. If the duplicate_flag is not set, then the device must respond in one of the following two ways. If the GENERAL ANNOUNCEMENT is targeted to a specific destination and the device knows of a path to that destination, then the device may respond by sending a TARGETED ANNOUNCEMENT to the sender with information about the best known path to the destination. The TTL field is set such that the sum of the TTL and the Hop_count is the same as the received packet, and the duplicate_flag is not set. If it does not send such a TARGETED ANNOUNCEMENT, the device must forward the GENERAL ANNOUNCEMENT to all of the device's neighbors as follows. The *Hop_count* of the outgoing packet is one plus the *Hop_count* of the incoming packet, or the length of the shortest path this device knows back to the source, whichever is smaller. If there is a TTL field, the device sets the TTL such that the sum of the TTL and the *Hop_count* is the same as for the received packet. The device sets the *duplicate_flag* of the packet if and only if the device already flooded a duplicate packet with a path back to the source of the flood that was as good as, or better than the path announced in this packet. In cases where the device updates its routing table because of this GENERAL ANNOUNCEMENT, the two-hop neighbor in the table is the device indicated in the *next_to_source* of the this packet.

3.2 TARGETED ANNOUNCEDMENT

Packets in the second class, which we call TARGETED ANNOUNCEMENTs, are typically used to confirm working paths. These packets are sent along paths that have been previously identified as valid with the minimum number of hops, perhaps using GENERAL ANNOUNCEMENTs or some other means. TARGETED ANNOUNCEMENTs are unicast along working paths, and announce the distance to devices that initiated these packets. With modifications, Route Reply packets (RREP) in DSR [13] and AODV [14] are examples.

TARGETED ANNOUNCEMENTs contain fields that are common to today's routing protocols: the addresses of the source and destination, distance to the source, and *recipient*, which is the neighbor to whom this packet is addressed. TARGETED ANNOUNCEMENTs may contain the TTL field, but it is not required. TARGETED ANNOUNCEMENTs also contain three atypical fields: the addresses of the device that is the next hop to the source and the device that is the next hop to the destination beyond the *recipient*, which we call *next_to_source* and n*ext_to_destination* respectively. Specifying the *next_to_destination* tells the *recipient* to whom it must forward the packet so as not to be identified as behaving selfishly*,* and specifying *next_to_source* tells the device so indicated to watch for selfish behavior. There is also a 1-bit *duplicate_flag* field, as described above for GENERAL ANNOUNCEMENTs. Other than fields cited above (*duplicate_flag, next_to_source, next_to_destination, path traveled, hop count, recipient, and TTL)*, we assume that fields in a TARGETED ANNOUNCEMENT remain unchanged as the packet is forwarded through the network.

Every device promiscuously listens to packet transmissions, regardless of the recipient specified in the packet header. Here is how a device handles a TARGETED ANNOUNCEMENT. If the packet announces a path to the source that is better than what is in the device's routing table, then it updates the hop count field and the two-hop neighbor field, as indicated in the *next_to_source* of the TARGETED ANNOUNCEMENT, in the routing table. If the packet contains a TTL that has expired, no action is required. If there is no TTL that has expired and the device is the intended *recipient*, then the device forwards the TARGETED ANNOUNCEMENT if and only if the *duplicate_flag* is not set or the packet announces a path back to the source that is shorter than the previously known best path. When forwarding, the device sets the *Hop_count* of the outgoing packet to either one plus the *Hop_count* of the incoming packet, or the length of the shortest path this device knows back to the source, whichever is smaller. If the packet contains a TTL, the device sets the TTL such that the sum of the TTL and the *Hop_count* is the same as for the received packet. The device sets the next_to_source as the 1-hop neighbor from which the device received the TARGETED ANNOUNCEMENT, and the next_to_destination as indicated in the two-hop neighbor field for the intended destination in the routing table. The device sets the *duplicate_flag* only if the device has already sent a packet to the same neighbor announcing a path back to the source that was as good or better than the one in this TARGETED ANNOUNCEMENT.

3.3 UPDATE

A device broadcasts a packet of the third class, which we call UPDATE packets to report that a route that went through the device is broken, or has increased in length. Update packets contain the sender's address, and for every path that has changed, the destination and the new path length. (If a protocol only uses UPDATEs where the length has become infinity, then this path length is implicit.)

A device responds to an UPDATE as follows. If the device knows of a better path to the destination that does not go through the sender of the UPDATE, then the device sends a TARGETED ANNOUNCEMENT to the sender with information on this path. Otherwise, the device updates its table as appropriate. If paths get worse, the device may transmit its own UPDATE packet, but we do not require this. With modifications, Route Error packets (RERR) in DSR [13] and AODV [14] , and Update messages that are generated when path lengths increase in DSDV [15], are prominent examples.

3.4 Extending Known Routing Protocols

Known routing protocols can be extended such that all routing packets fit in the categories described above. In DSR [13], a RREQ can be easily extended to a GENERAL ANNOUNCEMENT by adding the *duplicate_flag*. The n*ext_to_Source* field is already in the RREQ since DSR is source routed, and the *hop_count* field can be easily derived. A RREP can be extended to be a TARGETED ANNOUNCEMENT by adding the *duplicate_flag* and the *hop_count.* The *next_to_source* and the n*ext_to_destination* are implicit in the source route. A RERR is an UPDATE packet without any modification.

Similarly, all of the routing protocol packets for AODV [14] can be extended. A RREQ becomes a GENERAL ANNOUNCEMENT by adding the *next_to_source* and the *duplicate_flag*. The *next_to_source, next_to_destination,* and *duplicate_flag* fields can be added to a RREP packet to be a TARGETED ANNOUNCEMENT. By looking in the routing table, every device can easily obtain the values for the n*ext_to_source* and the n*ext_to_destination.* A RERR packet and a HELLO message are UPDATEs without modification.

Unlike current DSR and AODV protocols, which only require the RREQ and the RERR to be transmitted at maximum power, devices must transmit all routing packets at maximum power. Moreover, DSR and AODV allow devices to suppress duplicate packets, but we require devices to propagate duplicate packets if they announce shorter paths.

It is also possible to devise proactive routing protocols whose routing protocol packets all fit within the above categories. For example, each device could periodically flood a GENERAL ANNOUNCEMENT, which would allow all other devices to find the current shortest path back to the originator of the packet.

Note that there are protocols containing packet types that do not fall within any of the above categories, and which therefore cannot easily be extended such that selfishness is detectable using the algorithms described in this paper. For example, there are packets that are flooded with a field that specifies forwarders, as in [31]. If a device is allowed to choose which of its neighbors forward a packet and which do not, then a device can behave selfishly by specifying only those neighbors that are not on the shortest path. Thus, only the longer paths that go through this device will be visible and longer paths are less likely to be selected.

4. Detecting Instances of Selfishness

This section presents our detection mechanism. It proves that our mechanism detects every instance of selfishness and never identifies a device as behaving selfishly when that device follows the protocol under simplifying about the environment. These assumptions are described in Section 4.1. A follow-up paper will extend the mechanism to a more complex environment.

Selfish behavior can take one of three forms: failing to correctly transmit a packet that should be transmitted, transmitting a packet that should not be transmitted, and transmitting a packet at the incorrect power. (Forwarding a packet incorrectly is both failing to forward correctly, and transmitting a packet that should not be transmitted.) With the categories of packets defined in Section 3, the only time a device is required to transmit a packet is after that device has received a GENERAL or TARGETED ANNOUNCEMENT. Thus, the sender of such packet can watch to ensure that the proper action is taken by all neighbors. When a device fails to respond as required in a way that might be selfish, a watchdog mechanism identifies this behavior as selfish. This mechanism is described in Section 4.2. As described in Section 1, a device can also behave selfishly through power manipulation alone. To detect this, all devices observe the power levels of packets received from all neighbors. This mechanism is described in Section 4.3. Where devices generate packets at their own initiate in violation of the protocol and in a way that might be selfish, this must be detected by a different mechanism, which is described in Section 4.4. For this, devices observe their neighbors promiscuously to detect packets containing incorrect information.

4.1 Assumptions

It is possible to accurately detect every instance of selfish behavior in an environment where packets are not lost due to interference, links between neighbors are bidirectional, devices are stationary, and queuing delays are bounded. More precisely:

1. No Packet Loss: Let P_{AB} be the power threshold from Device A to Device B where P_{AB} is a constant. If Device A transmits a packet to Device B at power ≥ P_{AB}, then Device B always receives the packet. On the other hand, if Device A transmits at power lower than $P_{AB,}$ then Device B never receives the packet.
2. Bidirectionality: Devices send packets to and accept packets from their neighbors. A Device A considers another Device B to be its neighbor if and only if Device A has successfully received a packet from Device B and Device B has successfully received a packet from Device A.
3. Power Relation: For packets traveling from any Device A to Device B, increasing Device A's transmit power increases the received power at Device B.
4. Bounded Delay: Any packet that a device intends to transmit will be transmitted and will be received by all neighbors within a period of duration *x* after arriving, and all devices in the network know the duration *x*.

Theorem 1 If Devices A and B are neighbors and Device A transmits a packet at maximum power, then Device B receives it.

Proof:

By assumption, Devices A and B are neighbors. According to Assumption 2 above, if Devices A and B are neighbors, then Device B has successfully received a packet from Device A. By Assumption 1, Device B receives a packet only if Device A transmits the packet at a power ≥ P_{AB}, which implies that Device A transmitted some Packet Q at a power ≥ P_{AB}. By definition, Device A's maximum power is greater than or equal to the power at which Packet Q was transmitted, so this maximum power is ≥ P_{AB}. By assumption 2, whenever Device A transmits a packet at any power above P_{AB}, which would include Device A's maximum power, Device B must receive the packet. QED

4.2 Selfishness in Packet Forwarding

As discussed in section 1, after receiving a TARGETED or GENERAL ANNOUNCEMENT, a device that does not want to carry traffic for others can alter fields to make a path look worse than it is, or it can stop spreading news of good routes that include the device itself. Thus, after transmitting a TARGETED or GENERAL ANNOUNCEMENT, a device determines which of its neighbors are expected to respond, perhaps by forwarding the packet. It then watches those neighbors until the expected packets are observed, and makes sure that the neighbor did not alter any field in a selfish manner.

Each device uses a Verification Table to keep track of anticipated responses from neighbors. Each entry in the Table represents a packet for which that device is waiting for a response. Each entry contains the packet, a *neighbor_list* and a *timer.* The *neighbor_list* is the list of neighboring devices that are expected to respond. The *timer* indicates the expiration time of the entry; whenever an entry is added, the *timer* is set to expire after a period of duration x, as defined in Assumption 4.

Sections 4.2.1 and 4.2.2 present and prove the effectiveness of watchdog mechanisms that can determine whether all neighbors who are required to respond to a packet have indeed responded. Section 4.2.1 addresses TARGETED ANNOUNCEMENTs and Section 4.2.2 addresses GENERAL ANNOUNCEMENTs. It is assumed in these sections that devices can tell when a neighbor forwards a packet *correctly*, which in this paper means a device is transmitting the packet without any alterations that could possibly constitute selfish behavior. Section 4.2.3 describes how this is achieved. Section 4.2.4 explains why a separate watchdog mechanism of this kind is not needed for UPDATEs.

4.2.1Detection with a TARGETED ANNOUNCEMENT

This section describes the algorithm which detects devices that behave selfishly when receiving a TARGETED ANNOUNCEMENT. After transmitting a TARGETED ANNOUNCEMENT, a device must monitor if it expects its neighbor to forward the packet, which occurs when the *duplicate_flag* is not set and the TTL (if specified) is not expiring. In this case the device creates an entry for that packet in the Verification Table in which the *neighbor_list* consists of this neighbor.

If the device overhears its neighbor forward the TARGETED ANNOUNCEMENT, the device checks that its neighbor has not increased the hop count by more than one. It also verifies that the packet was forwarded correctly, the details of which will be presented in Section 4.2.3. If all conditions are met, the device removes the entry for this neighbor from the Verification Table. On the other hand, if the *timer* for an entry expires, then the device concludes that the device in the n*eighbor_list* of that expiring entry has behaved selfishly by failing to forward the TARGETED ANNOUNCEMENT correctly.

We now prove the correctness of our algorithm; selfish acts of this type will be identified by the algorithm, and a device's behavior will only be identified as selfish if that device is violating the protocol in a potentially selfish manner. Because there is often no easy way to prevent the Sybil attack [29] in a cooperative commons, we allow the possibility that any device may have multiple identities. We first prove that if Identity B behaves selfishly, then a neighboring Identity A can detect it. We then prove that even if Device B has multiple identities, it will be identified as selfish by at least one other device if and only if Device B behaves selfishly.

Theorem 2 Let Identity A (ID_A) follow the protocol and transmit a TARGETED ANNOUNCEMENT to its neighbor Identity B (ID_B). Let the protocol require ID_B to forward the TARGETED ANNOUNCEMENT, but ID_B fails to forward the TARGETED ANNOUNCEMENT correctly. Then, ID_A detects ID_B's behavior as selfish using the mechanism in Section 4.2.1.

Proof (by contradiction):

Assume that ID_A does not detect ID_B's selfish behavior. Thus, no entries in ID_A's Verification Table expire when ID_B is in the *neighbor_list*. This implies that either the entry was never added to the Verification Table, or the entry was added and removed before the *timer* of the entry expires.

As described in Section 3.2, a device is required to forward a TARGETED ANNOUNCEMENT if and only if the *duplicate_flag* is not set and the TTL counter is not expired (if specified), or the duplicate_flag is set, the TTL counter is not expired (if specified) and the packet announces a better path than what ID_A has announced before. If ID_A announces a better path than what ID_A previously announced with the *duplicate_flag* set, this implies that ID_A has violated the protocol as specified in Section 3.2, contradicting the assumption that ID_A follows the protocol. Thus, the assumption that ID_B was required to forward the TARGETED ANNOUNCEMENT implies that ID_B received the packet with the *duplicate_flag* not set and the TTL counter not expiring (if specified). As the algorithm in Section 5.2.1 specifies, a device creates an entry in the table when the device forwards a packet with the *duplicate_flag* not set and the TTL counter not expiring (if one is specified). This implies that ID_A did create an entry in its table. Thus, the entry must have been removed before the *timer* expired.

As specified in the algorithm in Section 4.2.1, ID_A will remove the entry from its table only when it overhears ID_B transmit the packet, and ID_A has verified that the packet was forwarded correctly. This contradicts the assumption. QED

Theorem 3 Let Identity A (ID_A) and Identity B (ID_B) be on the path with the smallest hop counts. Then, ID_A and ID_B cannot belong to the same device.

Proof (by contradiction):

Assume that ID_A and ID_B are identities of the same device. By assumption, the shortest path includes both ID_A and ID_B, and possibly other identities in between. Without loss of generality, let the transmission power of ID_B be ≥ the transmission power of ID_A. This implies that all neighbors of ID_A are neighbors of ID_B, which in turn implies that ID_A's neighbor, which is on the same shortest path and is not located in between ID_A and ID_B, is also a neighbor of ID_B. Hence, there exists a path between that neighbor and ID_B, excluding ID_A and all other identities in between ID_A and ID_B (if any). This path is shorter since it does not include any identities that were not in the original path, and it excludes at least one (ID_A) that was on the original path. However, this contradicts the assumption that both ID_A and ID_B are on the shortest path. QED

As described in Section 3.2, a TARGETED ANNOUNCEMENT is sent along the path that sender believes has the minimum number of hops. Theorem 4 shows that, as long as this is the case, a device behaving selfishly will be detected by a neighboring device, even if devices are able to assume multiple identities.

Theorem 4 Let Identity A (ID_A) follow the protocol and transmit a TARGETED ANNOUNCEMENT to its neighbor Identity B (ID_B) along a path with the smallest hop count from a source to a destination. Let the protocol require ID_B to forward the TARGETED ANNOUNCEMENT, but ID_B fails to forward the TARGETED ANNOUNCEMENT correctly. Then, an identity on a different device from Identity ID_B identifies ID_B's behavior as selfish using the mechanism in Section 4.2.1.

Proof:

The TARGETED ANNOUNCEMENT is sent along the path that has the minimum number of hops, so by Theorem 3, ID_A and ID_B belong to different devices. Moreover, by Theorem 2, ID_A detects selfish behavior of ID_B. Therefore, it is the *device* with the identity ID_A that detects selfish behavior of another *device* with the identity ID_B.

Theorem 5 Let Identity A (ID_A) follow the protocol, and conclude using the mechanisms in Section 4.2.1 that Identity B (ID_B) has behaved selfishly. Then, (i) ID_B has received a TARGETED ANNOUNCEMENT that ID_B was required to forward correctly, (ii) ID_B failed to forward the packet correctly and at the required power before its behavior is identified as selfish, and (iii) ID_B did not ever intend to forward the packet correctly and at the required power.

Proof:

As described in this section, ID_A detects ID_B's behavior as selfish only when a *timer* expires, and the entry in ID_A's table associated with that *timer* includes ID_B as the *neighbor_list*. A device creates an entry in the Verification Table associated with a given neighbor when and only when it forwards a packet to that neighbor with the *duplicate_flag* not set and the TTL counter of that packet not expiring (if specified). Thus, ID_B must have been ID_A's neighbor, and ID_A must have transmitted a TARGETED ANNOUNCEMENT to ID_B with the *duplicate_flag* not set and the TTL counter greater than one. By assumption, ID_A follows the protocol that requires a device to transmit a TARGETED ANNOUNCEMENT at maximum power, as explained in Section 3. Hence, by Theorem 1, when ID_A transmits a TARGETED ANNOUNCEMENT to ID_B, ID_A must have transmitted at maximum power, and ID_B that is a neighbor of ID_A (by assumption) must have received the packet. Since the *duplicate_flag* was not set and the TTL counter did not reach zero, ID_B was required to forward the packet. Thus, (i) is proven.

As described in Section 4.2.1, ID_A would remove the entry associated with a neighbor if it sees that neighbor forward the associated packet correctly. Since the entry is still there when the *timer* expires, ID_A did not see this occur before the *timer* expires. By Theorem 1, ID_A receives the packet that the neighbor ID_B transmits at maximum power. As described in Section 3, the protocol requires a device to transmit a TARGETED ANNOUNCEMENT at maximum power. Since ID_A follows the protocol and did not see ID_B forwarding the packet correctly, ID_B did not forward the packet correctly at the required power before ID_A's *timer* expired, which is the time at which ID_A identifies ID_B's behavior as selfish. Thus, (ii) is proven.

As described in Section 4.2, ID_A's *timer* expires a period of duration x after it forwards the TARGETED ANNOUNCEMENT. By Assumption 4 in Section 4.1, any packet that ID_B intends to transmit will be transmitted within the period of duration x. Since the packet has not been forwarded correctly and at the required power when the *timer* expires, and the protocol requires that the timer expires a period of duration x after the packet was received, ID_B did not intend to forward the packet correctly and at the required power. Thus, (iii) is proven.

4.2.2 Detection with a GENERAL ANNOUNCEMENT

In this section, we describe the algorithm which detects devices that behave selfishly after receiving a GENERAL ANNOUNCEMENT. Similar to Section 4.2.1, a device that is flooding a GENERAL ANNOUNCEMENT must determine whether it requires any of its neighbors to forward that packet, and if so, the device must monitor to ensure that the packet has been forwarded correctly. As discussed in Section 3.1, the device does not watch that its neighbor(s) forwards the packet if and only if the duplicate_flag is set or the TTL counter (if specified) is expiring, in which case no entry is added to the Verification Table. In other cases where the *duplicate_flag* is not set and the TTL counter (if specified) is bigger than one, the device determines which of its neighbors will be required to forward the packet. The device does not require all of its neighbors to forward the packet; the device does not require further forwarding from those neighbors, including the previous hop that flooded this GENERAL ANNOUNCEMENT, that previously flooded the GENERAL ANNOUNCEMENT and retransmitting this packet will not announce smaller hop counts. For all other neighbors, the device adds an entry to the Verification Table (i.e., the *neighbor_list* for this entry contains all other neighbors).

When a device overhears its neighbor transmit a packet, the device checks that its neighbor transmits correctly. There are two cases to consider. If the overheard GENERAL ANNOUNCEMENT has the hop count that is not increased by more than one, and the sum of the hop count and the TTL counter is the same as what the device previously transmitted, the device confirms that that neighbor follows the protocol. The other case is that the neighbor responds by transmitting the corresponding TARGETED ANNOUNCEMENT. In case the packet contains the TTL counter, the device confirms that the sum of the TTL counter and the hop count of the TARGETED ANNOUNCEMENT is unchanged. Once the device confirms that the neighbor transmits correctly, the device removes that neighbor from the *neighbor_list* of every entry associated with the overheard packet. If the *timer* of an entry expires in the Verification

Table, the device detects that its neighbors that remain in the *neighbor_list* behave selfishly and fail to forward the packet correctly.

We now prove the correctness of our algorithm in a similar manner to Section 4.2.1. We prove that selfish behavior of this type will be identified by the algorithm, and a device's behavior will only be identified as selfish if that device is violating the protocol in a potentially selfish manner. We first prove that if Identity B behaves selfishly with a GENERAL ANNOUNCEMENT, then the neighboring Identity A will detect it. We then prove that even if Device B has multiple identities, it will be identified as selfish by at least one other device if and only if Device B behaves selfishly.

Theorem 6 Let Identity A (ID_A) and Identity B (ID_B) be neighbors. Let ID_A follow the protocol, and transmit a GENERAL ANNOUNCEMENT. ID_B neither forwards the GENERAL ANNOUNCEMENT correctly, nor responds by transmitting a TARGETED ANNOUNCEMENT correctly when ID_B is required to do one of the two. Then, ID_A detects ID_B's behavior as selfish using the algorithm in Section 4.2.2**.**

Proof (by contradiction):

Assume that ID_A does not detect ID_B's behavior as selfish. Thus, no entries in ID_A's Verification Table expire when ID_B is included in the *neighbor_list*. This implies that either the entry, of which the *neighbor_list* contains $ID_{B,}$, was never added to the Verification Table, or the entry was added and ID_B was removed from the *neighbor_list* before the *timer* expires.

As described in Section 3.1, a device is required to forward a GENERAL ANNOUNCEMENT if and only if the *duplicate_flag* is not set and the TTL counter is not 0, or the duplicate_flag is set and the TTL counter is not 0 and the packet announces a better path than ID_A has announced before. If ID_A announces a better path than what ID_A previously announced with the duplicate_flag set, this implies that ID_A has violated the protocol as specified in Section 3.2, contradicting the assumption that ID_A follows the protocol. By assumption, ID_B was required to either forward the packet or respond with a TARGETED ANNOUNCEMENT, which implies that the *duplicate_flag* was not set, and the TTL counter was not 0. Since ID_A is assumed to follow the protocol, this in turn implies that ID_A did create an entry in its Verification Table, and ID_B was in the *neighbor_list*.

As described above, ID_A will remove the entry from its table only when it overhears ID_B transmit either the GENERAL or TARGETED ANNOUNCEMENT, and ID_A has verified that the packet was forwarded correctly. This contradicts the assumption. QED

As described in Section 3.1, a GENERAL ANNOUNCEMENT announces the distance to the source and a device may choose the path with the minimum hop counts using the GENERAL ANNOUNCEMENT. It is the device on the shortest path that must be watched for selfish behavior to evade the watchdog mechanism and avoid forwarding subsequent data packets. Theorem 7 shows that, a device that is on the shortest path and behaves selfishly will be detected by a neighbor, even if devices are able to assume multiple identities.

Theorem 7 Let Identity A (ID_A) and Identity B (ID_B) be neighbors. Let ID_A follow the protocol and flood a GENERAL ANNOUNCEMENT. Let the protocol require ID_B to either forward the GENERAL ANNOUNCEMENT correctly, or responds by transmitting a TARGETED ANNOUNCEMENT, but ID_B fails to do one of the two. If the GENERAL ANNOUNCEMENT travels along a path with minimum hop count, an identity on a different device from ID_B identifies ID_B's behavior as selfish using the algorithm in Section 4.2.2.

Proof:

By Theorem 6, ID_A identifies ID_B's behavior as selfish. By assumption, the GENERAL ANNOUNCEMENT travels on the path that has the minimum hop count. Therefore, by Theorem 3, if ID_A and ID_B are on the shortest path, they belong to different devices. Therefore, it is the device with the identity ID_A that detects selfish behavior of another device with the identity ID_B. QED

Theorem 8 Let Identity A (ID_A) follow the protocol. Let ID_A conclude using the algorithm in Section 4.2.2 that Identity B (ID_B) was behaving selfishly. Then, (i) ID_B has received a GENERAL ANNOUNCEMENT that it was required to either forward, or respond by transmitting the TARGETED ANNOUNCEMENT correctly, (ii) ID_B failed to transmit the packet correctly and at the required power before its behavior is identified as selfish, and (iii) ID_B did not ever intend to transmit the packet correctly and at the required power.

Proof:

As described in Section 4.2.2, ID_A detects ID_B's behavior as selfish only when a *timer* expires, and the entry in ID_A's Verification Table associated with that *timer* includes ID_B in the *neighbor_list*. A device creates an entry in the Verification Table and adds a given neighbor to the *neighbor_list* when and only when it floods a packet with the *duplicate_flag* not set and the TTL counter not expiring, and that neighbor will announce the shortest path by forwarding this packet. Therefore, ID_B must have been a neighbor of ID_A, and ID_A must have transmitted a GENERAL ANOUNCEMENT to ID_B with the *duplicate_flag* not set and the TTL counter greater than one, and forwarding this packet will make ID_B to announce the shortest path.

By assumption, ID_A follows the protocol that requires a device to transmit a GENERAL ANNOUNCEMENT at maximum power, as explained in Section 3. Hence, ID_A must have transmitted at maximum power. Since ID_A and ID_B are neighbors, ID_B must have received this GENERAL ANNOUNCEMENT by Theorem 1. Since the *duplicate_flag* was not set and the TTL counter was not 0, ID_B was required to either forward the GENERAL ANNOUNCEMENT, or transmit the corresponding TARGETED ANNOUNCEMENT. Thus, (i) was proven.

As described in Section 4.2.2, ID_A would remove a neighbor from an entry's *neighbor_list* if it sees that neighbor transmit a corresponding packet for that entry correctly (i.e., that neighbor either forwards the GENERAL ANNOUNCEMENT or transmits the TARGETED ANNOUNCEMENT associated with that entry). Since the entry still contains ID_B in the *neighbor_list* when the *timer* expires, ID_A did not see this occur before the *timer* expires. By Theorem 1, ID_A receives the packet that the neighbor ID_B transmits at maximum power. As described in Section 3, the protocol requires a device to transmit both GENERAL and TARGETED ANNOUNCEMENT at maximum power. Since ID_A follows the protocol and did not see ID_B forwarding the packet correctly, ID_B did not transmit the packet correctly before ID_A's *timer* expired, which is the time at which ID_A identifies ID_B's behavior as selfish. Thus, (ii) is proven.

As described in Section 4.2, ID_A's *timer* expires a period of duration x after it forwards the GENERAL ANNOUNCEMENT. By Assumption 4 in Section 4.1, any packet that ID_B intends to transmit will be transmitted within the period of duration x. Thus, since the packet has not been transmitted correctly when the *timer* expires, and the protocol requires that the timer expires a period of duration x after the packet was received, ID_B did not intend to transmit the packet. Thus, (iii) is proven. QED

4.2.3 Verifying a Packet was Correctly Forwarded

In cases where Device A expects its neighboring Device B to forward a routing packet, Device A monitors the neighbor's transmissions until it observes the expected packet. Most of the fields in a routing packet do not change en route, so Device A expects to see that all of these fields exactly match the packet it just transmitted. The greater challenge for a watchdog mechanism designed to detect all selfish behavior is to detect incorrect changes to fields that are allowed to change en route.

Of the fields presented in Sections 3.1 and 3.2, there are only two that change en route, and that can be altered selfishly. One is the *hop_count*, which is addressed in Sections 4.2.1 and 4.2.2 for the separate cases of TARGETED and GENERAL ANNOUNCEMENTs, respectively. The other is the TTL counter, which we explicitly allow but do not require. Although devices are expected to change the TTL, we require the sum of the *hop_count* and TTL to be the same as that of the packet that the device received, as described in Sections 3.1 and 3.2. Thus, if the *hop_count* can be verified, then the TTL can also be easily verified.

For the remaining *next_to_source, next_to_destination*, and *duplicate_flag* fields described in Sections 3.1 and 3.2 that change en route, we now show that any modifications that are inconsistent with the protocol are not selfish acts, since the device that made these modifications would not benefit.

Incorrectly changing fields related to the path the packet has taken or will take can prevent some packets from reaching their destinations. For example, Device A may insert an incorrect address in the *next_to_source*, which is in both the GENERAL and TARGETED ANNOUNCEMENT. Alternatively, Device A may insert an incorrect address in the *next_to_destination*, which is only in the TARGETED ANNOUNCEMENT. Changing these fields causes the packet to be mis-routed, and even though the source of the packet believes that the packet is traveling on a valid path, the packet may not reach the intended device. We also allow (but do not require) packets to contain addresses of devices on the path. Altering one or more addresses will have the same effect of mis-routing packets as described above. These actions are disruptive, but none benefits Device A; because none affects the hop count, so changing these fields incorrectly does not make the announced paths through Device A less attractive. Moreover, if a path through Device A is selected, the fact that this path could be invalid is unlikely to reduce the number of packets that Device A must handle. If a path through Device A does not work, the source may repeatedly attempt to reestablish the path, creating additional packets that watchdog devices will expect Device A to carry. In addition, as described in Section 3.2, all of Device A's neighbors promiscuously monitor transmissions, and can therefore learn the valid paths through Device A. Thus, the next time one of Device A's neighbors receives a GENERAL ANNOUNCEMENT, it may inform the source about the path that goes through Device A, and there is nothing Device A can do to prevent this.

A device may violate the protocol and set the *duplicate_flag* on the packet that is not a duplicate, expecting the neighbor to drop the packet. However, our algorithm specifies that the neighbor does not take the option of dropping but instead forwards the packet if it announces the shortest path. Therefore, as long as neighbors follow the protocol, the device gains no benefit by setting the *duplicate_flag* when the packet is not a duplicate. Similarly, if the device does not set the *duplicate_flag* for a packet that is a duplicate, this may increase the control traffic, but it does not change any device's estimation of the paths available or their lengths.

4.2.4 Detection When Forwarding an UPDATE

Failing to respond to an UPDATE is not a selfish act. As described in Section 3.3, UPDATE packets are used to announce routes that are broken, or have increased in length. When Device A fails to transmit an UPDATE even though there are changes in the paths that go through Device A, other devices that use those paths do not learn about the changes; those devices still consider the paths that go through Device A as good as previously known, and transmit packets to Device A, expecting Device A to forward packets. Thus, not responding to an UPDATE packet can only increase a device's burden.

4.3 Selfishness in Power Manipulation

As explained in Section 1, when a device knows that its behavior is watched by a neighbor, the device can behave selfishly and pretend to cooperate by forwarding a packet at a power level that is observable to the watchdog and not to other devices. As described in Section 3, our algorithm requires devices to transmit control packets, which include GENERAL ANNOUNCEMENTs, TARGETED ANNOUNCEMENTs, and UPDATEs at maximum power so that detection of the selfish behavior of manipulating transmission power is observable. However, if the threshold power to reach the watchdog device is greater than the threshold power to reach the next hop to the destination of the packet, the device may transmit at a reduced and sufficient power to reach the watchdog device, instead of transmitting at maximum power such that the watchdog fails to identify the device's behavior as selfish.

Devices keep track of the maximum received power for all neighbors in the Power Table. This table consists of two columns; the first column, denoted as *address,* records the IP address of the neighbor from which the packet is transmitted, and the second column, denoted as P_{max}, represents the maximum received power at which the device has ever received packets.

Here is the description of the algorithm. Whenever a device receives a packet from a neighbor, the device compares the received power with that neighbor's stored maximum power P_{max} in the Power Table. If the received power of the packet is smaller than P_{max}, then the device identifies that neighbor's behavior as selfishly reducing transmission power.

Whenever a device receives a packet from its neighbor*,* if the received power is greater than that neighbor's P_{max}, then the device sets P_{max} equal to this received power. When updating P_{max}, the device not only considers control packets but also data packets from the neighbor because the neighbor may increase transmission power for the neighbor's own data packets to reduce the chance of retransmission.

We now prove that if a device transmits a control packet at a power at which at least one neighbor but not all neighbors are able to receive the transmission, then at least one neighbor will detect this selfish behavior.

Theorem 9 Device B manipulates power such that at least one of its neighbors does not receive Packet P, and at least one neighbor does receive the packet. Without loss of generality, let Device A receive the packet, and let Device C not receive the packet. Then Device A detects that Device B has manipulated transmission power of the packet in a selfish manner using the mechanism in Section 4.3.

Proof: By assumption, Device A received Packet P. This implies that packet P was transmitted at power that is greater than or equal to the power threshold P_{BA} from Device B to Device A by Assumption 1. By assumption, Device C did not receive Packet P, and by Assumption 1, Packet P was transmitted at power below the power threshold P_{BC} from Device B to Device C. Therefore, $P_{BA} < P_{BC}$.

By assumption, Device B and Device C are neighbors, so by Assumption 2, Device C must have received a packet successfully from Device B. Without loss of generality, let this packet be labeled as Packet Q. By Assumption 1, Packet Q must have been transmitted at power greater than or equal to P_{BC}. Since $P_{BA} < P_{BC}$, Packet Q was also transmitted at a power greater than P_{BA}. By Assumption 1, Device A also received Packet Q. Since Device A received both Packets P and Q, and Packet P was transmitted at power less than P_{BC}, and Packet Q was transmitted at power greater than or equal to P_{BC}, by Assumption 2, the received power for Packet Q was greater than received power for Packet P.

Device A maintains the maximum received power P_{max} it has observed from Device B in its Power Table. This P_{max} must be greater than or equal to the received power for Packet Q, which is greater than the received power for Packet P. Thus, P_{max} is greater than the received power for Packet P. As described above, whenever Device A receives Packet P from Device B at a power less than P_{max}, Device A concludes that Device B is selfishly manipulating power. QED

Theorem 10 Let Device A and Device B be neighbors. Let Device A detect that Device B has manipulated power in a selfish manner when transmitting Packet P using the mechanism in Section 4.3. Then, Device B must have violated the protocol.

Proof:

For Device A to conclude that Device B has selfishly manipulated transmit power for Packet P, Packet P must be a routing packet, and the received power of Packet P must be less than the P_{max} in Device A's table.

When Device A sets the P_{max} associated with Device B, Device A sets P_{max} equal to the received power of the packet that Device A just received from Device B. Thus, Device A once received a packet from Device B at received power P_{max}. Without loss of generality, let this packet be labeled as Packet Q. By Assumption 3, the fact that received power for Packet Q is greater than the received power for Packet P implies that the transmit power for Packet Q is greater than the transmit power from Packet P. This implies that Device B transmitted Packet P at power less than Device B's maximum power. The protocol requires that all routing control packets be transmitted at maximum power. Thus, Device B violated the protocol. QED

4.4 Selfishness in Packet Fabrication

As explained in Section 1, a device may make the path through the device look longer than it really is. This can be done without responding to a received packet as in Section 4.2; the device can simply generate a packet that announces arbitrarily high distance to a given destination.

In contrast**,** it is not a selfish act if the announced path is better than the best path that the device knows. Such actions can only increase the load on the device. Thus, we focus on mechanisms that can detect the announcement of worse paths.

As discussed in Section 3, we allow (but do not require) packets to carry some field indicating the freshness of the information on path length, such as sequence number in AODV [14]. If this field used, the announced path is better than the real one if there is more recent information on the former, e.g. the sequence number in AODV is higher, or if the information is equally recent and the hop count is smaller. The announced path is worse if there is equally recent information, and the hop count is larger, so this is the case we must detect reliably. If the information on the announced path is less recent than a previously announced path, e.g. the AODV sequence number is lower, then the announcement will simply be ignored.

As described in Section 3.2, all devices listen promiscuously to all routing packets generated by their neighbors. When a device overhears its neighbor transmitting a TARGETED ANNOUNCEMENT that announces a worse path than what the neighbor has learned about, the device transmits the TARGETED ANNOUNCEMENT with the superior path information to that neighbor (i,e. the *recipient*). The device sets the TTL of this packet as one plus the TTL of the overheard TARGETED ANNOUNCEMENT so that all devices that learned the worse path will learn the better path. The device does not set the *duplicate_flag* since the device requires its neighbor to forward the packet. The *next_to_source* is the next hop to reach the source of the TARGETED ANNOUNCEMENT, which can be found in the routing table, and the *next_to_destination* is the same as that from the original TARGETED ANNOUNCEMENT that announced worse path. This device creates an entry in the Verification Table and watches if that neighbor forwards the TARGETED ANNOUNCEMENT correctly, using the mechanism described in Section 4.2.1.

Similarly, when a device overhears its neighbor transmitting a GENERAL ANNOUNCEMENT or an UPDATE, and the device knows a better path to the same source or destination, respectively, the device transmits a GENERAL ANNOUNCEMENT that announces the better path. The TTL of this packet is one plus the TTL of the overheard packet for the same reason as explained above, and the next_to_source of this packet is the next hop to the the source of the GENERAL ANNOUNCEMENT, as can be found in the routing table. The device does not set the *duplicate_flag* on this announcing packet. This device creates an entry in the Verification Table for that neighbor, and watches if that neighbor forwards the GENERAL ANNOUNCEMENT correctly using the mechanism presented in Section 4.2.2.

We now prove the correctness of our algorithms. We prove that if a device fabricates any routing control packet that is either a GENERAL ANNOUNCEMENT, a TARGETED ANNOUNCEMENT, or an UPDATE, and that announces a path that the device knows is not the best, some neighbor will detect it. Moreover, we prove that if a device's behavior is identified as selfish, that device has violated some requirement of the protocol. We assume that the device acting selfishly still transmits routing packets at maximum power, as selfish power manipulation is already detected by the algorithm in described Section 4.3.

Theorem 11 Let all of Device A's neighbors follow the protocol. Device A transmits a routing packet at maximum power to one or more of its neighbors indicating a path to Device D (where D≠A), although Device A knows about a superior path to Device D. Moreover, Device A does not follow this routing packet within a period of duration 2x by transmitting another routing packet at maximum power to the same neighbor or neighbors, where this other routing packet announces the superior path. Then one or more of Device A's neighbors will identify Device A as selfish using the mechanism in Section 4.4.

Proof:

As specified in Sections 3.2 to 3.4, a device sends a GENERAL ANNOUNCEMENT, a TARGETED ANNOUNCEMENT, or an UPDATE to announce the path from the device to the source that initiates the packet. By assumption, Device A knows a superior path to Device D (where D≠A) that Device A has previously learned about. This implies that there is a neighbor that is the next hop to Device D on the superior path, and Device D must have learned about the superior path when that neighbor announced the superior path.

By assumption, the packet was transmitted at maximum power, so by Theorem 1, all neighbors received the packet. By assumption, all Device A's neighbors follow the protocol, including the neighbor that is the next hop to Device D on the superior path. As described in Section 4.4, when a device overhears its neighbor transmitting a packet, and the device knows a better path to the same source, the device transmits a packet at maximum power with the better path information to that neighbor within the period of duration x after receiving the packet. Thus, that neighbor transmits a packet with the superior path information at maximum power within the period of duration x, which means within duration 2x after Device A transmitted its packet. By theorem 1, Device A must receive the packet, which occurs within 2x after the incorrect packet was generated by Device A. Because it is assumed that all neighbors follow the protocol, if the packet that Device A transmitted to announce a worse path was a TARGETED ANNOUNCEMENT, then the neighbor will transmit a TARGETED ANNOUNCEMENT with the superior path information, and if the packet Device A transmits was a GENERAL ANNOUNCEMENT or an UPDATE, then the neighbor will transmit a GENERAL ANNOUNCEMENT, as specified in Section 4.4.

As described in Section 4.4, the GENERAL or TARGETED ANNOUNCEMENT will be sent with the *duplicate_flag* not set. The protocol therefore requires Device A to forward this packet within the period of duration x upon receiving the packet. By assumption, Device A does not follow this routing packet within a period of duration 2x with another routing packet to the same neighbor or neighbors, where this other routing packet announces the superior path. Therefore, if the packet that the neighbor transmits to Device A is a TARGETED ANNOUNCEMENT, the neighbor detects selfish behavior of Device A by Theorem 2, and if the packet that the neighbor transmits is the GENERAL ANNOUNCEMENT, the neighbor detects selfish behavior of Device A by Theorem 6. QED

Theorem 12 Let Device B identify Device A's behavior as selfish using the mechanism n Section 4.4, where Device B follows the protocol. Then (i) Device A must have transmitted a packet indicating a path to some destination Device D (where D≠A), although Device A knew a superior path to Device D, (ii) Device A received a packet that it was supposed to transmit, (iii) Device A failed to forward the packet correctly before its behavior is identified as selfish, and (iv) Device A did not ever intend to transmit the packet correctly.

Proof:

As described in Section 4.4, Device B concludes using this mechanism that Device A is acting selfishly only if either Device A has sent a TARGETED ANNOUNCEMENT to Device A and concluding using the mechanism in Section 4.2.1 that Device A selfishly failed to forward the packet correctly at maximum power, or Device A has sent a GENERAL ANNOUNCEMENT to Device A and concluding using the mechanism in Section 4.2.2 that Device A selfishly failed to forward the packet correctly at maximum power. In the case of a TARGETED ANNOUNCEMENT, by Theorem 5, (i) Device A has received a TARGETED ANNOUNCEMENT that Device A was required to forward correctly, (ii) Device A failed to forward the packet correctly and at the required power before its behavior is identified as selfish, and (iii) Device A did not ever intend to forward the packet correctly and at the required power. In the case of a GENERAL ANNOUNCEMENT, by Theorem 8, (i) Device A has received a GENERAL ANNOUNCEMENT that it was required to either forward, or respond by transmitting the TARGETED ANNOUNCEMENT correctly, (ii)

Device A failed to transmit the packet correctly before its behavior is identified as selfish, and (iii) Device A did not ever intend to transmit the packet correctly. Thus, in either case, the theorem is proven.

5. Conclusion

A cooperative commons is a promising and novel approach to reduce the cost of blanketing an area with coverage and to use spectrum more efficiently, if all devices cooperate [1-3, 10]. However, the approach may not be viable if devices do not cooperate by behaving selfishly or maliciously [1, 2]. A particularly effective form of selfish behavior is for a device to handle (or mishandle) routing packets in such a way that far fewer routes through that device are selected. The device can then avoid forwarding a potentially large number of data packets for others. Given the lack of rewards for cooperating with devices belonging to other owners and other administrative domains, selfish behavior must somehow be deterred if the cooperative commons is ever to become as valuable as its proponents hope. The first step towards deterrence is reliable detection of selfish behavior.

We have demonstrated that it is not possible to reliably detect selfish behavior in the routing protocols used today in ad hoc networks. Typical watchdog approaches are unable to detect some forms of selfish behavior, such as when power control is used to conceal that behavior. Perhaps even worse, these approaches can falsely label some behavior as selfish when the devices are following the protocol, as often occurs with duplicate suppression. Other approaches assume trust relationships that are appropriate in other contexts, but may not be achievable in a cooperative commons. Thus, in addition to developing effective watchdog algorithms, it is necessary to adopt routing protocols that are suitable for a cooperative commons in which selfish behavior is observable.

We have shown that effective detection of selfish behavior in a cooperative commons is possible, as long as devices do not use rapidly steerable antennae and devices are generally not able to collude with neighbors. These two exceptions are unavoidable limitations in a cooperative commons. We have defined broad categories of routing control packets in which selfishness can be detected, and shown that packets in common routing protocols can be extended to fit into these categories.

For any routing protocol that meets our requirements, including the extended versions of DSR [13] and AODV [14], our proposed new watchdog algorithms can detect every instance of selfish behavior under some simplifying assumptions, i.e. where there are no packet losses, devices are not moving, and queuing delay is bounded. Moreover, the watchdog algorithms never identify a device's behavior as selfish if that device is complying with the protocol. In a more complex and realistic environment where those simplifying assumptions are relaxed, our approach is still effective in detecting devices that repeatedly exhibit selfish behavior.

A follow-on paper will describe how our approach can be extended to identify devices that repeatedly behave selfishly in a more complex environment in which packets are lost, queuing delay can grow large, and path loss between two devices can change over time due to mobility or other causes. For example, to accommodate shadowing and multipath, we include a margin for error so that received power can fluctuate even when the transmit power does not.

Of course, detection of selfish behavior is only the first step. Our future work will address deterrence, and how devices can respond after determining that a neighbor is behaving selfishly.

8. References

[1] J. M. Peha. "Emerging Technology and Spectrum Policy Reform," International Telecommunications Union (ITU) Workshop on Market Mechanism for Spectrum Management, January 2007. www.ece.cmu.edu/~peha/wireless.html

[2] J. M. Peha, "Sharing Spectrum through Spectrum Policy Reform and Cognitive Radio," to appear in *Proceedings of the IEEE.* www.ece.cmu.edu/~peha/wireless.html

[3] D. Reed. "Comments for FCC Spectrum Task Force on Spectrum Policy," www.reed.com/OpenSpectrum/FCC02-135Reed.html, July 10, 2002.

[4] Meraki Wireless Network, http://meraki.com/.

[5] NetEquality Equal Access Community Internet, http://www.netequality.org/.

[6] The Champaign-Urbana Community Wireless Network, http://www.cuwin.net/

[7] R. Bruno, M. Conti, and E. Gregori. "Mesh Networks: Commodity Multihop Ad Hoc Networks." In IEEE Communication Magazine, Vol. 43, Issue 3, pp. 123-131, March 2005.

[8] T. Fujii, Y. Suzuki. Ad-hoc Cognitive Radio-Development to Frequency Sharing System by Using Multi-hop Network-. Proc. IEEE DySPAN 2005, pp.589-92, Nov. 205.

[9] K. Hamdi, and K. B. Letaief. "Cooperative Communications for Cognitive Radio Networks." PGNet 2007, Jun. 2007.

[10] J. M. Peha, B. Gilden, R. Savage, S. Sheng, B. Yankiver, "Finding an Effective Sustainable Model for a Wireless Metropolitan-Area Network: Analyzing the Case of Pittsburgh," *35th Telecommunications Policy Research Conference,* (TPRC) Sept. 2007. www.ece.cmu.edu/~peha/wireless.html

[11] A. Agarwal and P. R. Kumar. "Capacity Bounds for Ad hoc and Hybrid Networks," ACM SIGCOMM Computer Communications Review, Vol. 34, No. 3, July 2004.

[12] Y.-C. Hu and A. Perrig. "A Survey of Secure Wireless Ad Hoc Routing." In *IEEE Security & Privacy*, special issue on Making Wireless Work, 2(3):28-39, May/June 2004.

[13] D.B. Johnson, D. A. Maltz, and J. Broch. "DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks." In Ad Hoc Networking, edited by Charles E. Perkins, Chapter 5, pp. 139-172, Addison-Wesley, 2001.

[14] C. E. Perkins and E. M. Royer. "Ad Hoc On Demand Distance Vector (AODV) Routing." IETF Internet Draft, http://www.ietf.org/rfc/rfc3561.txt, July 2003.

[15] C. E. Perkins and P. Bhagwat. "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers." In Proceedings of the ACM SIGCOMM: Computer Communications Review, Vol. 24, No. 4, pp. 234-244, Oct. 1994.

[16] S. Marti, T.J. Guili, K. Lai, and M. Baker. "Mitigating routing misbehavior in mobile ad hoc networks." In Proceedings of MOBICOM 2000, p. 255-265, 2000.

[17] P. Michiardi and R. Molva. “Core: a Collaborative Reputation Mechanism to Enforce Node Cooperation in Mobile Ad hoc Networks,” In Proceedings of the IFIP TC6/TC11 Sixth Joint Working Conference on Communications and Multimedia Security, p. 107-121, Deventer, The Netherlands, 2002.

[18] S. Buchegger and 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, Network-based Processing, p. 403-410, January 2002.

[19] S. Buchegger and J.-Y. Le Boudec. “Performance Analysis of the CONFIDANT protocol.” In Proceedings of 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing, June 2002.

[20] S. Buchegger, C. Tissieres, and J.-Y. Le Boudec. “A Test-Bed for Misbehavior Detection in Mobile Ad-hoc Networks – How Much Can Watchdogs Really Do?” In Proceedings of the Sixth IEEE Workshop on Mobile Computing Systems and Applications, 2004.

[21] Y. Rebahi, V. Mujica, and D. Sisalem. “A Reputation-Based Trust Mechanism for Ad hoc Networks.” In Proceedings of the 10th IEEE Symposium on Computers and Communications, p. 37-42, 2005.

[22] Q. He, D. Wu, and P. Khosla. “SORI: A Secure and Objective Reputation-based Incentive Scheme for Ad-hoc Networks.” In Proceedings of IEEE WCNC2004, March 2004.

[23] S. Bansal and M. Baker. “Observation-based Cooperation Enforcement in Ad-hoc Networks.” Technical Report, Stanford University, 2003.

[24] R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan. “Sustaining Cooperation in Multi-Hop Wireless Networks.” In Proceedings of the 2nd Conference on Symposium on Networked Systems Design Implementation, Vol. 2, pp. 231-244, 2005.

[25] Y. Yoo and D. P. Agrawal. “Why Does It Pay to Be Selfish in a MANET?” IEEE Wireless Communications, Vol. 13, issue 6, pp. 87-97, Dec. 2006.

[26] K. Balakrishnan, J. Deng, and P. K. Varshney. “TWOACK: Preventing Selfishness in Mobile Ad Hoc Networks, “in Proceedings of IEEE Wireless Communications and Networking Conference, vol. 4, p. 2137-2142, 2005.

[27] K. Liu, J. Deng, and K. Balakrishnan. "An Acknowledgement-Based Approach for the Detection of Routing Misbehavior in MANETs," IEEE Transaction on Mobile Computing, Vol. 6, No. 5, May 2007.

[28] D. Djenouri and N. Badache. “New Approach for Selfish Nodes Detection in Mobile Ad hoc Networks.” Workshop of the 1st International Conference on Security and Privacy for Emerging Areas in Communication Networks, p. 288-294, September 2005.

[29] J. R. Douceur. “The Sybil Attack.” In First International Workshop on Peer-to-Peer Systems (IPTPS ’02), Mar. 2002.

[30] S. Singh, M. Woo, and C. S. Raghavendra, "Power aware routing in mobile ad hoc networks." In Proceedings of ACM MOBICOM, 1998, pp. 181-190.

[31] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, L. Viennot. "Optimized Link State Routing Protocol for Ad Hoc Networks." In Proceedings of the 5th IEEE Multi Topic Conference (INMIC 2001), 2001.