# Watchdogs to the rescue: Securing wireless TCP

Shehla S Rana
Department of Electrical and
Computer Engineering
University of Illinois at Urbana-Champaign, USA
Email: ssrana2@illinois.edu

Nitin H Vaidya
Department of Electrical and
Computer Engineering
University of Illinois at Urbana-Champaign, USA
Email: nhv@illinois.edu

***Abstract*—In this paper, we make a case for using watchdogs to protect against misbehavior in dense wireless networks. We introduce "Generalized Watchdogs" and identify when and how watchdogs can be necessary and sufficient against misbehavior. We study feasibility of watchdog approach and show that the order of capacity bounds is preserved asymptotically even with watchdogs. We use generalized watchdogs to design protocols to improve *both* security and performance of TCP over wireless networks such that the application at the destination never accepts a corrupted packet and we achieve this without modifying TCP. We show that a strict dependence on availability and success of watchdogs can lead to "watchdog induced losses" and establish their effects on TCP throughput. We then propose solutions to deal with these losses and make watchdogs intelligent so they can tune the overheads incurred. With hop-by-hop verification of packet correctness, we ensure that tampered packets are not forwarded in the network and thus save potential wastage of network resources. We use NS-2 simulations of both controlled as well as realistic network scenarios, to show that watchdogs can provide simple, lightweight and reliable means of misbehavior detection, tolerance and most importantly "deterrence" while saving costs of security infrastructure. With a combination of intelligent watchdogs and source coding, and by leveraging route adaptation, our scheme achieves *twice* the throughput of a cryptographic alternative and that too in presence of as high as $30\%$ packet tampering.**

## I. Introduction

Wireless networks are known to suffer from security issues primarily because of their broadcast nature. One example of misbehavior in multi-hop wireless networks is the *dropping attack* where an intermediate node drops packets that it is "expected" to forward. In a more malicious attack, sometimes called the *tampering attack*, an intermediate node modifies(corrupts) contents of a packet before forwarding. This may not only render the data useless, but may also make the application react to it in unexpected ways.

The same overhearing capabilities that raise security concerns can and have been used for detection of misbehavior. Ideally, in a multihop environment with omnidirectional antennas, a sender can overhear its next-hop node forwarding the packets. This ability, termed in literature as *watchdog* functionality, was first introduced for misbehavior detection by [1] where packet senders act as watchdogs.

After [1], a lot of work on misbehavior detection using watchdogs followed (refer to Sec. VI). However, all these protocols tend to use UDP and CBR for traffic [1], [2], [3], [4], [5], [6], [7] and any work on TCP with watchdogs is completely missing to the best of our knowledge. Some of these explicitly mention that they chose UDP to avoid particularities of a complicated protocol like TCP [3]. There may be several reasons why TCP is considered more challenging than UDP and we highlight some of them below:

1) TCP is meant to guarantee reliable and in-order delivery whereas UDP makes no such guarantees.
2) Time-consuming and "eventually" correct and accurate misbehavior detection schemes may be appropriate for UDP in the long run, but since TCP provides guarantees on reliable and in-order delivery, correctness of every single packet must be ensured to satisfy TCP semantics.
3) Because of TCP's congestion control algorithm, delays in misbehavior detection can result in fast throughput decay.

In view of above constraints, we lay down the requirements on a misbehavior detection system suitable for TCP in particular. First up, per-packet monitoring and checks are needed. Secondly, detection should be fast and must not cause TCP's slow-start to kick in too often. This requirement means that existing schemes for UDP that require many stages of observation collections and exchanges of suspect lists over time before making any decision are clearly unsuitable for TCP.

Since another major component of this work relates to source coding, we remark here that much of existing work there also explicitly recognizes that improving TCP throughput is complicated because TCP's congestion control interacts badly with delays required for packet accumulation at nodes [8], [9]. In summary, while watchdogs have been studied extensively for UDP, TCP has been left rather un-explored. Similarly, while network and source coding has found much use in improving TCP performance under the "good" network assumption, it has not been used to combat misbehavior in wireless TCP.

It was for these reasons, that we chose to work with TCP and as our results show, watchdog based security has considerable potential. Using watchdogs and coding, we were able to improve TCP throughput in presence of fairly high percentages of interference and packet tampering. We tested our schemes in both small, controlled environments as well as large, realistic network scenarios. Simulations under controlled conditions gave us ideas about fundamental behavior of the protocol while those for larger, realistic scenarios helped make

generalizations. We also allowed dynamic routing which helps the protocol recover against excessive tampering on a certain path. It also suggests that our scheme can work well in presence of mobility. To the best of our knowledge, this is the first work on misbehavior detection and tolerance in TCP using watchdogs and coding. The rest of this paper is organized as follows. In Section. II, we study watchdog feasibility and availability in random networks and discuss situations where watchdogs can be necessary and sufficient against misbehavior. Section. III presents a simple watchdog based security mechanism called *WD-TCP*. Section. IV defines and explains watchdog induced losses and establishes their severity. Then in Section. V, we propose solutions to deal with throughput degradation and watchdog induced losses in TCP. Section. VI provides an overview of the related work and we conclude in Section. VII.

## II. Generalized Watchdogs and their Feasibility

Watchdogs were introduced by Marti et al. [1] primarily for packet dropping attacks where the receiver either gets correct data or no data at all and hence does not need explicit information about misbehavior. We use the network of Figure. 1 to explain the approach. All antennas are omnidirectional. Suppose **S** sends traffic destined for **D** relying upon **R** to forward it. Using the sender as watchdog, if **S** can overhear **R** forwarding the message or otherwise, it can detect misbehavior. In essence, *misbehavior detection* is all that is required for this attack and *misbehavior tolerance* can be achieved with actions of **S** alone, by choosing an alternate route for example.

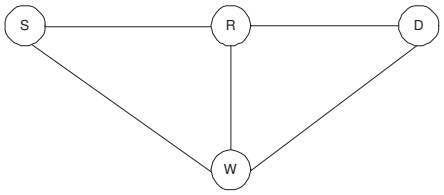


Fig. 1. An example 4-node Network

Suppose **R** forwards the packets but corrupts the contents before forwarding. Suppose further that the packet belongs to a TCP connection. Then, even if **S** can detect packet modification, **D** will still accept it and pass it on to the application. Afterwards, even if **S** retransmits the packet explicitly asking **D** to accept the new copy, it may be too late if the application has already consumed corrupted packets. To detect and tolerate such packet modification attacks, we introduce the notion of *Generalized Watchdogs* where a watchdog must not necessarily be sender of a packet but may be any node that can hear both the source and the intermediate node e.g. **W** in Fig. 1.

Further, since a watchdog at the source can not protect the destination from accepting corrupted packets, we propose that **W** should not only compare the packets it overhears from **S** and **R**, but also inform **D** about it. Meanwhile, **D** does not pass packets on to the application until it has heard from **W**. We incorporate these ideas to come up with simple yet efficient misbehavior tolerance schemes in Sec. III.

### *A. Availability of Generalized Watchdogs in Random Networks:*

Before we build misbehavior detection mechanisms employing generalized watchdogs, we need to establish their easy availability in commonly encountered network scenarios.

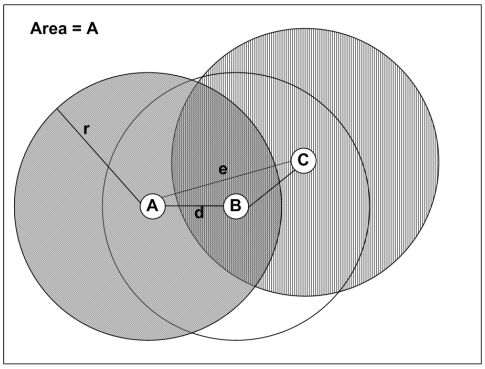


Fig. 2. Requirements for Watchdog Availability

We begin by investigating feasibility of watchdog based approaches for use in reasonably dense networks. Refer to Fig. 2 and consider placing $N$ wireless nodes randomly in a region of area $A$. All nodes have a transmission range of $r$. Average number of nodes in a disc of radius $r$ will then be $\pi r^2 \frac{N}{A}$. For a watchdog to be available for transmission from A to B, there must be at least one node in the area of intersection of two circles of radii $r$, centered at A and B respectively(bigger lens in Fig. 2). However, since we require the watchdog to also notify the next-hop, then if B relays A's packets to C, we need at least one node in area of intersection of all three circles (darkest portion of Fig. 2). Since B lies strictly inside communication range of both A and C, there must be some non-zero intersection between all three circles. For watchdog availability, we need at least one node in this non-zero intersection and this is likely to hold w.h.p when $N$ is large in random networks for a given $A$.

Watchdog availability does not imply watchdog success because interference constraints for watchdog to successfully hear *both* transmissions from A to B and B to C may be different from those required for success of individual transmissions from A to B and B to C. However, we also claim that asymptotically, requirements for watchdog availability and success preserve the order of capacity bounds for random networks and a brief proof sketch is provided in Appendix.

For further insights, we did NS-2 simulations with the objective of quantifying watchdog availability for monitoring a given communication over two hops. We simulated 10 random network topologies each for 20, 30, 50 and 75 nodes placed randomly in a 1000x1000m area and did 10 runs for each topology with a random seed. All nodes have a communication range of 250m. We set up 15 UDP flows between randomly

selected source-destination pairs. We chose to work with a network density of 50 or more nodes in a 1000mx1000m area since similar scenarios can be seen often in misbehavior detection and opportunistic routing literature [6], [1], [2]. We record how many packets are overheard over two successive hops or equivalently, the number of intermediate relays whose behavior was observed by a generalized watchdog and averaged this over all runs. The results are shown in Fig. 3.

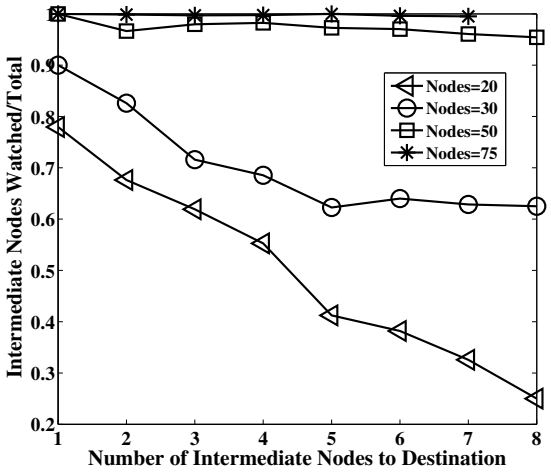


Fig. 3. The Availability of Watchdogs

The X-axis shows number of intermediate nodes that forward a packet until its final destination and Y-axis plots ratio of the number of intermediate nodes that were watched with respect to the total. A value of $Y = 0.95$ corresponding to $X = 2$ means that $95\%$ of all packets that were forwarded by two relays were watched at both hops by some generalized watchdogs. As our results show, for reasonably dense networks, more than $95\%$ packets were successfully watched by some node even over long paths. These results strongly back the claim that watchdogs are readily available. Not surprisingly, Fig. 3 also suggests that watchdogs based schemes are not well suited to sparsely populated networks.

We emphasize on use of watchdogs primarily because they do not require additional device or computation level capabilities for misbehavior detection. There are other reasons too though. For example, without watchdogs, it's not straight forward to isolate a single node as bad with absolute certainty and it is common to localize misbehavior to granularity of a "nexus" i.e. two nodes and link between them [10]. We now discuss how watchdogs can become necessary and sufficient for isolating misbehaving nodes.

### B. Watchdogs can be necessary

Cryptographic schemes including end-to-end (E2E) and hop-by-hop authentication are popular for misbehavior detection. However, with E2E encryption, for example, neither end of the transmission can localize the fault on a path and might shun the whole path including possibly many fault-free nodes. Furthermore, over long routes, valuable network resources are wasted when a packet that was corrupted in earlier part of the route still goes all the way to the destination only to be dropped.

However, watchdogs can localize misbehavior and routes may be updated accordingly. Another advantage of watchdogs over encryption is "deterrence" against misbehavior because possibility of being caught can discourage malicious activity. With hop-by-hop authentication like that suggested in [11], misbehavior can be localized to two adjacent nodes but overhead of establishing and maintaining authentication relationships, (especially with mobility) combined with key management, make it expensive for per-packet monitoring and delay-sensitivity required for TCP.

### C. Watchdogs can be sufficient

Consider the network as a connected graph $G = (V, E)$ where $V$ is set of vertices(nodes) and $E$ the set of edges(links). For any three nodes forming a clique, when any one of them uses the other to relay its packets, the third can function as a watchdog. Therefore, a sufficient condition for watchdogs to successfully detect misbehavior is if number of faulty nodes in every clique of $size = 3$ in $G$ is limited to $\leq 1$.

For example, under single failure, accusation from one watchdog is sufficient to detect misbehavior and those from two watchdogs are sufficient to conclude that accused node is indeed faulty. Therefore, with constraints on allowable number of failures in the network and across neighborhoods, watchdogs can be sufficient to detect and localize misbehavior to one node.

## III. WD-TCP: A Protocol for TCP security with Generalized Watchdogs

With the objective of ensuring that TCP-based application never has to accept a corrupted packet, we now propose and evaluate a simple watchdog protocol *WD-TCP*. We propose that if a node receives a packet from a sender other than its IP source (e.g. **D** receives **S**'s packets from **R** in Fig. 1), it should wait to hear from a watchdog before forwarding the packet onwards or passing it up to application. From watchdogs, we require that they send a *notification* like that in Fig. 4 to *next-hop* of the packet, containing information about the forwarding node, observed packet sequence number and a 1-bit indicator of whether the packet is good or bad.

The algorithm at each node is explained in Algorithm. 1. Each node maintains two sets of packets: *PendingNotif* for packets whose notifications have not been received yet and *HeardOnce* for packets it has overheard once. Both sets are initially empty. When node *i* receives data packet *p* meant for itself (line 3), this implies *i* is either a relay or final destination for *p*. *i* checks whether a notification for *p* has been received (line 4). If notification was positive (line 5), *i* forwards *p* otherwise it drops *p*. If *i* has not received a notification for *p* yet, it adds *p* to *PendingNotification* (line 10). On the other hand, if *i* receives a notification (line 11), then it forwards or drops the corresponding packet from the set *PendingNotification* depending upon whether notification

**Algorithm 1** *WD-TCP* Algorithm

1: $At\ node\ i:$
2: $Received\ pkt\ =\ p$
3: **if** $(dst(p) == i) AND (type(p) == data)$ **then**
4:  **if** $(notif(p) == received)$ **then**
5:  **if** $((notif(p) -> is\_Bad == 0))$ **then**
6:  $fwd(p)$;
7:  **else**
8:  $drop(p)$;
9:  **else**
10:  $PendingNotification \leftarrow p$;
11: **else if** $(dst(p) == i) AND (type(p) == Notif)$ **then**
12:  $notif(p) = received$;
13:  **if** $(notif == good)$ **then**
14:  $fwd(PendingNotif(p))$;
15:  **else**
16:  $drop(PendingNotif(p))$;
17: **else if** $(dst(p)! = i) AND (type(p) == Data)$ **then**
18:  **if** $(p \in HeardOnce)$ **then**
19:  **if** $(p == HeardOnce(p))$ **then**
20:  $sendNotif(0)$;
21:  **else**
22:  $sendNotif(1)$;
23:  $Delete(HeardOnce(p))$
24:  **else**
25:  $HeardOnce \leftarrow p$;

is positive or negative (lines 13-16). Finally, if *i* is not the packet's destination (implying *i* overheard *p* and may be a watchdog), if it has heard this packet before, it compares the two versions and sends a notification accordingly by setting $Is\_Bad$ to $'0'$ or $'1'$ (lines 18-23), otherwise *i* buffers *p* in *HeardOnce*.

We implement *WD-TCP* without modifying TCP by inserting watchdog functionality with MAC layer to intercept packets and buffer them. Each node only forwards packets to next-hop (or hands them to application layer at the destination), after a notification verifying their correctness has been received from a watchdog. Therefore, effectively, TCP sink releases acknowledgements only for verified packets.

We now evaluate *WD-TCP* in a 4-node network of Fig. 1 for a TCP flow from **S** to **D** via **R**. There is no tampering. These settings help us compare the compromise we make by adding a dependence of TCP on watchdogs. We simulated this scenario in NS-2 with RTS/CTS disabled. A single simulation lasts 100 seconds for all results reported in this work. The results, averaged over several runs, are shown in Fig. 5. Throughout this paper, we use number of untampered TCP packets received at the destination as an indicator of throughput. We compare this metric for *WD-TCP* and basic TCP (TCP without watchdogs) in top two curves in Fig. 5. Not surprisingly, watchdog based TCP performs worse than unmodified TCP because of per-packet notification overhead but it still seems that *WD-TCP* has not compromised too badly in terms of throughput. However, note that this degradation came in the best case scenario i.e. when there was neither interference nor tampering/misbehavior in the network.

| Watched Id | Packet Seq Num | IsBad? |
|---|---|---|

Fig. 4. Contents of Per-Packet Watchdog Notification

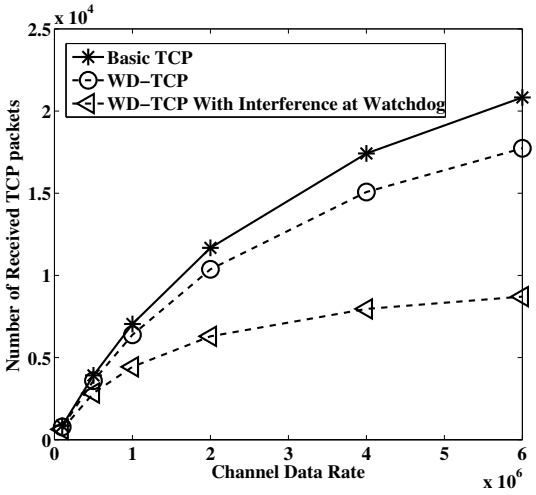


Fig. 5. A comparison of TCP performance with and without Watchdog for 4-node network

In real networks, even if misbehavior were uncommon, interference is still the norm rather than the exception implying that *WD-TCP* has a high cost which may soar further when there is in fact a real threat. To validate this, we simulated the 4-node network by adding cross traffic to cause interference only at watchdog. Results form the lowest curve in Fig. 5 showing disappointing performance of *WD-TCP*. We had expected that overhead of notifications will degrade TCP throughput but as we now elaborate on, the effects of dependence upon watchdogs manifest themselves in some other interesting ways too.

## IV. Watchdogs Induced losses

Watchdogs notifications inject non-data traffic. Since *WD-TCP* is constrained such that intermediate nodes can only forward packets after receiving WD notifications, there arises a new problem. We call it the *watchdog induced loss* (WD-induced loss) problem and explain it below.

Consider, 4-node network of Fig. 1 and assume that *WD-TCP* with per-packet notifications from watchdogs is used. In this scenario, a watchdog induced loss can happen as follows:

1) **Type-I WD-Induced Loss:** Suppose **R** forwards the packet and **D** receives it but **W** fails to overhear it due to collision/cross traffic at **W**. **W** can therefore not send a notification and even though **D** received the (possibly untampered) packet correctly, **D** can not acknowledge it until **S** times out and retransmits the packet giving

TABLE I
TCP RETRANSMISSIONS AND WATCHDOG INDUCED LOSSES FOR 4-NODE NETWORK

| Data Rate Mbps | 0.5 | 1 | 2 | 4 | 6 |
|---|---|---|---|---|---|
| Total Retransmissions | 319 | 517 | 725 | 943 | 1041 |
| TypeI WD-Induced Loss | 294 | 483 | 670 | 872 | 980 |
| TypeII WD-Induced Loss | 8 | 21 | 23 | 31 | 46 |

**W** another chance to observe it. The same happens if **W** misses overhearing the packet from **S** to **R** and only overhears it from **R** to **D**.

2) **Type-II WD-Induced Loss:** Suppose **W** completely misses out on overhearing some packet over both **SR** and **RD**. **W** will realize this when it overhears a packet with a sequence number $N_2$ while the last packet it overheard had a sequence number $N_1$ such that $N_2 > N_1 + 1$.
3) **Type-III WD-Induced Loss:** Finally, suppose **W** has overheard the packet over the links **SR** and **RD** but finds the channel busy due to cross traffic and its notification must wait. If this takes long enough, **S** might timeout waiting for the ACK and retransmit a packet that has already been received and even monitored.

We saw how watchdog dependent *WD-TCP* may *create* artificial packet losses that can severely degrade throughput. This is aggravated further under heavy traffic, channel errors and interference and renders our simple scheme very inefficient. We now evaluate these effects and try to quantify them.

Notice that WD-induced losses would cause a retransmission of a packet that may well have been received at the destination. We therefore compare total number of retransmissions with ones that were triggered by WD-induced losses. The results in Table. I show that the biggest contributor to WD-induced losses were of Type I. Also, the total number of retransmissions is only slightly larger than the sum of Type I and Type II WD-induced losses. Since throughput of *WD-TCP* suffers badly because of increased notification overhead and watchdog induced losses, it is imperative to overcome with these limitations. This is our focus in the next section.

## V. COPING WITH WATCHDOGS INDUCED LOSSES AND THROUGHPUT DEGRADATION

We wanted our watchdog based protocols to be reliable, lightweight and that their overhead must not outweigh the benefit. Therefore, in this section, we propose measures to help ameliorate throughput degradation and watchdog induced loss problem in *WD-TCP*.

### A. Decreasing Watchdog Notification Frequency

One way to reduce throughput degradation is by decreasing frequency of watchdog notifications i.e. send cumulative notifications. The notification content had to be modified accordingly as shown in Fig. 6. Two new fields were added: $Last\ Notification$ field contains sequence number of the last notification sent by watchdog. Using $Packet\ Sequence\ Num$ and $Last\ Notification$ fields, the receiver can update verification status of all packets with sequence numbers $S_N$ where $Packet\ Sequence\ Num \geq S_N > Last\ Notification$. This holds true because we only allow watchdogs to delay notifications for untampered, in-order packets. Upon detecting packet tampering, a new notification is sent promptly with $IsBad$ field set to 1. This way, we only need to indicate goodness of current packet with delayed notifications instead of all packets in the interval. The second field, $Missed\ Packet$, is a 1-bit field used when a watchdog misses overhearing a packet or overhears out of order packets. It then sends two prompt notifications. One about all consecutive packets it heard and for the second, it sets $Missed\ Packet$ to 1 indicating that it overheard all good packets with sequence numbers from $Last\ Notification$ onwards but missed the packet(s) with sequence number(s) $S_n$ such that $Packet\ Sequence\ Num - 1 \geq S_N > Last\ Notification$

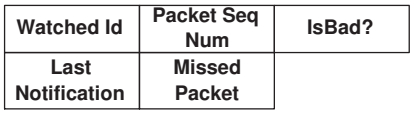


Fig. 6. Contents of Delayed Watchdog Notification

We experimented with fixed notification intervals and saw throughput improvements since more time could be used for sending TCP data and ACKs. But instead of a hard notification interval, we decided to use information available to the watchdog to adaptively decide when to send notifications. We propose a scheme similar to Additive Increase Multiplicative Decrease (AIMD), where watchdogs increase notification interval additively upon overhearing in-order packets and bring it down to 1 upon hearing retransmissions or out of order packets. This essentially estimates congestion window size at the sender so watchdogs can delay notifications for as long as the sender can keep releasing packets without receiving an ACK.

With adaptive notification interval at W, we again run *WD-TCP* from S to D (Fig. 1) and show how *WD-TCP* outperforms basic TCP in Fig. 7.

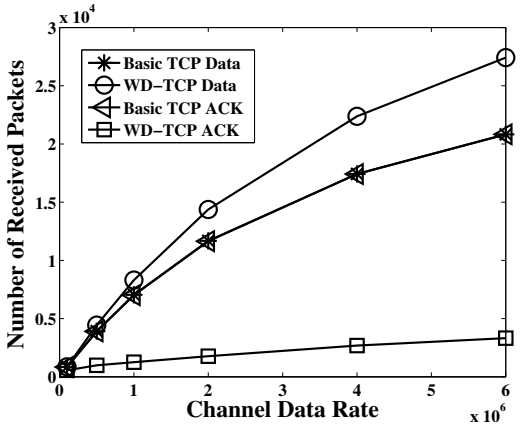


Fig. 7. Performance of *WD-TCP* with Adaptive Notification Interval for 4-node network

Throughput of *WD-TCP* improved because when **W** sends cumulative notifications, watchdog shim at **D** hands packets over to the application only after receiving notifications for them. Therefore, TCP sink is bound to send cumulative acknowledgements. As congestion window size grows, so does the notification and cumulative ACK interval. This reduces ACK traffic (lowest curve in Fig. 7) and channel contention for *WD-TCP*.

We also show in Fig. 8 ($datarate = 1Mbps$), that notification interval adaptation algorithm follows congestion window at the source well even in presence of interference. For robustness, we also added a time-out based notification. This way, if the source starts sending packets slower, or if watchdog misses many packets, it does not wait indefinitely to send a notification.

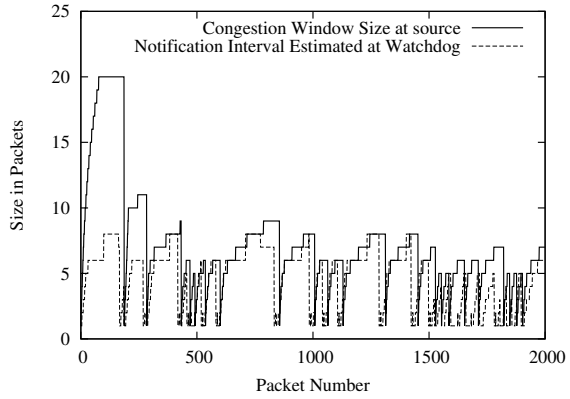


Fig. 8. Congestion Window at the TCP source and its estimate at the watchdog under interference for 4-Node network

We also ran *WD-TCP* with adaptive notifications for a large network of 75 nodes spread randomly over a 1000mx1000m area. The transmission radius is fixed at 250m. We set up 10 TCP flows between randomly chosen source-destination pairs and repeat the simulation 10 times with random seeds. There is interference between flows and also at watchdogs causing WD-induced losses. There is no tampering. Any node can be a generalized watchdog and multiple watchdogs may also be available for some packets. At each hop, the packet is forwarded only after a notification for its correctness is received. Otherwise, it is dropped right there. For multiple watchdogs e.g. **W1** and **W2**, suppose **W1** sends notification first and **W2** overhears it. Then, if **W2**'s own observation conflicts with that of **W1**, **W2** sends a prompt notification, otherwise it sends the notification with some probability (set to 0.5 in simulations). There are several advantages of this. First, it reduces overhead in case of multiple watchdogs. Second, if **W1** and **W2** conflict, and **W2** is malicious, then **W1** can be certain of this and does not trust **W2** anymore. Third, with notifications from multiple watchdogs, the receiver can also find out which node is actually malicious based on majority. We use AODV for routing and when intermediate nodes drop packets because of negative reports from watchdogs, AODV finds another route. We compare unmodified TCP with *WD-TCP* for varying data rates. We could have used static routing but we wanted to keep the simulation realistic because the protocol's performance in face of changing routes also gives ideas about its performance with mobility. The results for aggregate throughput over all flows in the network are shown in Fig. 9. Surprisingly, *WD-TCP* seems to perform better in the larger network than it did in the 4-node network with interference(refer to Fig. 5). This is because interference is no longer limited to the watchdog alone but traffic sources, destinations and relays also experience interference which is more realistic. Furthermore, it turned out that sometimes, watchdog requirement lead to use of routes that were better (shorter or had lesser interference) than those used with basic TCP. However, lesser opportunities were available for accumulating several notifications. This was because a significant amount of time was spent waiting for channel access and watchdogs timed out in the meanwhile and sent notifications.

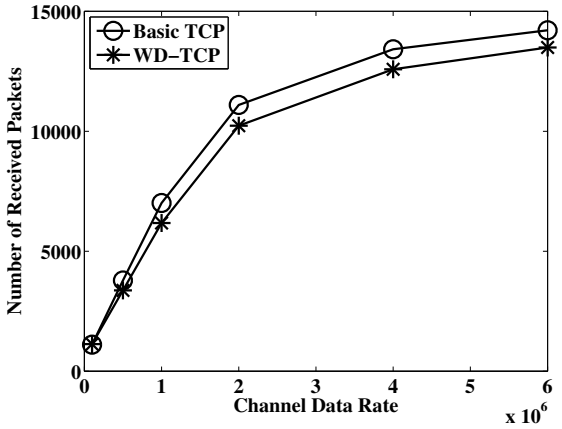


Fig. 9. Aggregate throughput for 75-node network with 10 flows

To get more insights into whether *WD-TCP* compromised on route length and forced flows to use much longer sub-optimal routes, we also plot the quantity *packet-relays product* which gives the number of relays traversed by a packet summed over all the packets. This is shown in Fig. 10 for both basic TCP and *WD-TCP* for the 75-node network. The fact that bars for *WD-TCP* are not significantly higher than those for basic TCP suggests that the routes chosen by *WD-TCP* were not very suboptimal.

To sum up, in this section, we incorporated intelligence in the watchdogs so they can reduce overheads and still perform well. Next we look at WD-Induced loss problem and a solution that works well in presence of interference and packet tampering.

### *B. Increasing Watchdog Success Probability*

The major contributor to WD-induced losses is failure of a watchdog to overhear packets because of cross traffic interference. Therefore, one way to deal with WD-induced losses is by minimizing probability of this event or maximizing the probability of a watchdog successfully overhearing a packet.

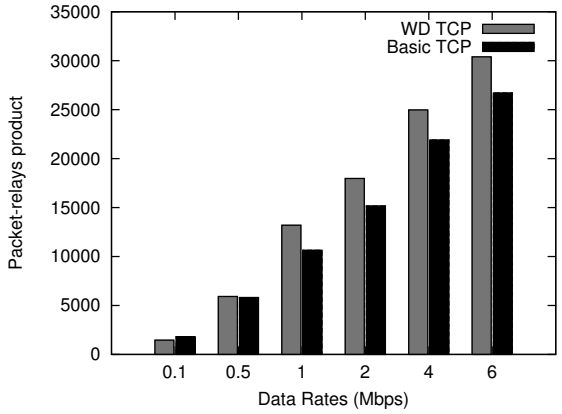


Fig. 10. Number of PacketXRelays transported in the network for 75-node network

We propose a combination of source coding and watchdog mechanism to design a scheme that ameliorates WD-induced loss problem. We add a coding shim below TCP layer at the sender and receiver. At the sender, the coding layer uses an $(n, k)$ online code to encode $k$ data packets followed by $n-k$ coded packets calculated from the $k$ uncoded packets just sent. It also adds a *sub-sequence number* to each packet to let the watchdog and receiver identify packets correctly. If watchdog can then observe and notify about any $k$ of the $n$ packets from one generation, the receiver can decode the verified packets before passing them to application layer. The added redundancy increases watchdog success probability since it no more has to observe *each and every* packet. Therefore, even though it may still fail to overhear some packets, WD-induced retransmissions can be prevented to a great extent. This idea of using source coding with watchdogs is similar to [12] but their work did not present any results. We show actual results of running TCP to get an understanding of TCP's interaction with watchdogs and source coding. As part of implementation in NS-2, our coding layer adds a new coding header containing the subsequence number and information about the code. The watchdog notifications, Fig. 11 also include subsequence number to indicate whether notification is about a raw packet or a coded packet.

| Watched Id | Packet Seq Num | IsBad? |
|---|---|---|
| Last Notification | Missed Packet | Subsequence Num |

Fig. 11. A Watchdog Notification Packet with Coding

We now compare three different versions of TCP:

- *TCP with End-to-end encryption:* Here, the TCP sink can detect tampered packets and reject them soliciting a retransmission.
- *WD-TCP:* Is our simple watchdog based TCP protocol where watchdogs send a notification for every packet.
- *(6,5) Coded WD-TCP:* Is *WD-TCP* with (6,5) source coding such that if the watchdog can monitor any 5 out of 6 packets of the same generation, it can help the receiver decode the whole generation.

Our threat model is characterized by malicious relays modifying contents of TCP data before forwarding them with a probability $p$ for all three versions of TCP compared. We first experiment with the 4-Node network. Our objective is to infer whether degradation in throughput is due to watchdogs or will it be fundamental to TCP if it only accepts good packets. A comparison for the three variations of TCP is shown in Figs. 12 with $p = 0.1$. We mention here that for coded *WD-TCP*, we did not count redundancy packets among goodput and therefore the number shown in figure represents only uncoded data. Comparing *WD-TCP* with coded *WD-TCP*, we see the improvement brought about by source coding especially at higher rates. More surprisingly, despite WD-induced losses and tampering, coded *WD-TCP* even outperforms TCP with E2E encryption. This is a direct consequence of the fact that coding saves many timeouts and retransmissions because tampered or missed packets can be extracted from added redundancy.

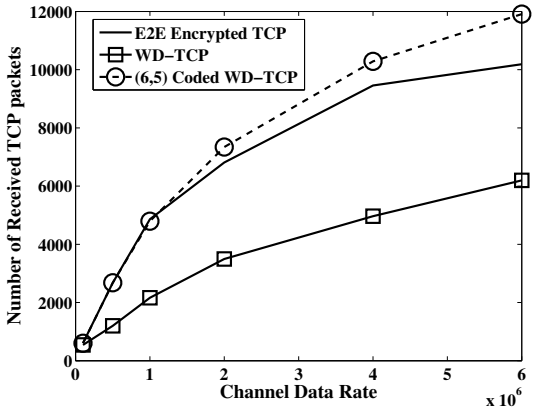


Fig. 12. A Comparison of E2E Encrypted TCP with uncoded *WD-TCP* and (6,5) Coded *WD-TCP* in 4-node network

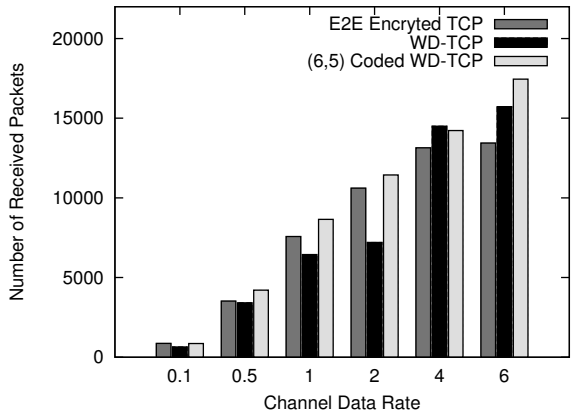


Fig. 13. A Comparison of aggregate throughput of E2E Encrypted TCP, uncoded *WD-TCP* and (6,5) Coded *WD-TCP* in 75-node network with 10 flows

We also simulated coded *WD-TCP* protocol for the 75 node network. 20 nodes were chosen randomly to do packet tampering. These remain same across all three versions of TCP. The results for aggregate throughput are shown in Fig. 13. It can be seen that coded *WD-TCP* outperforms other TCP versions except at data rate of 4Mbps. Our investigation into this scenario revealed, that for one particular flow, uncoded *WD-TCP* was repeatedly able to find a route without packet tamperers and therefore achieved slightly higher aggregate throughput.

Next, in Fig. 14, we show how throughput degrades when probability of packet tampering is increased. Nearly a quarter (20/75) of the nodes tamper packets with increasing probability. Data rate is set to 1Mbps. As can be seen, coded *WD-TCP* performs much better than E2E encrypted TCP especially for $p \geq 0.2$. We studied these scenarios carefully and found that because tampered packets are dropped as soon as they are detected at relays, AODV considers the route to be broken and looks for alternate routes. For this reason, with $p = 0.5$ for example, AODV constantly modified routes for coded *WD-TCP* unless a path with least packet tamperers was found. For E2E encrypted TCP however, the packets are forwarded all the way to the destination leading to poor response time towards route adaptation.

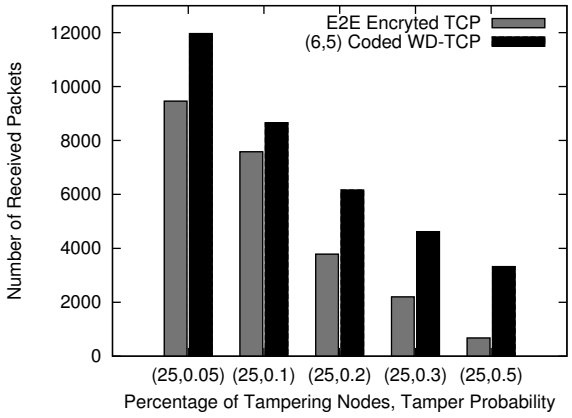


Fig. 14. Relationship between Throughput and Tamper Probability. Data rate set to 1Mbps

With the modifications made to WD-TCP, we have now shown results to support our claim that a combination of watchdogs and coding can provide strong protection against hostile channel conditions as well as malicious network elements.

## VI. Related Work

Wireless misbehavior detection and deterrence has long interested researchers. A complete watchdog based misbehavior detection and tolerance protocol was first proposed by [1]. It used NS-2 simulations to show improvements in UDP throughput. Since they strictly avoid suspected nodes, their protocol must minimize frequency of false detections. Following that, many *trust* and *reputation* based protocols became popular. CORE [13] presents a theoretical framework for building reputation with watchdogs and reputation tables. They use watchdog observations to formulate direct, indirect and functional reputation such that a node's positive reputation keeps on increasing as long as it behaves well, but if its reputation goes negative beyond a threshold, the node is shunned and excluded from the network. The paper is theoretical and lacks simulation/experimental results. CONFIDANT [3] uses watchdogs to build trust relationships and informs a node-specific "friend list" about observed misbehavior. They do not, however, propose a way to dynamically build/maintain this list and consider it as pre-configured on user-to-user basis. Recently, Zouridaki et.al [14] presented numerical results for trust establishment where watchdogs are used to form *opinions* about others in the network and this opinion is incorporated in routing decisions to protect against misbehavior.

Another component of our work used source coding for protection against watchdog induced losses. This idea of mixing multiple packets is not new especially with regards to opportunistic routing. [8], [15] and [9] use this idea and work on batches of packets and so a source must first accumulate enough packets to encode before it can begin sending them. To avoid delays incurred by waiting for enough packets to accumulate, Sundarajan et. al. [16] showed throughput improvement using an online coding scheme where the source sends out random linear combinations of packets currently in the congestion window. Building upon online coding, CoMP, [17] presents a combination of online coding and multipath forwarding to improve TCP. It proposes heuristics for estimating loss rates for forwarding rate adaptation. Much of the work cited above works with UDP or with TCP without any misbehavior. In this paper, we worked with TCP while allowing packet tampering attacks in order to understand how watchdogs and coding will interact with TCP intricacies.

As a final note, we mention that we provide watchdogs with incentive to not send false notifications. If a watchdog $W1$ sends false notifications, the packet sender, and other watchdogs know for certain that $W1$ is misbehaving and ignore future notifications from it. Also, $W1$ will fail to cause damage by false notifications since good watchdogs around it will realize that their observations conflict with that of $W1$ and will promptly notify the next-hop of it. Also, while *WD-TCP* incurs notification overheads, it makes up by saving resources spent in forwarding of tampered packets towards the destination.

## VII. Conclusion

In this paper we have tried to focus on strengths and weaknesses of using watchdogs against wireless misbehavior with a focus on TCP. We chose TCP as our traffic since the literature so far has shied away from it because of its complex interaction with delays, losses and errors. We show simulation results for both controlled as well as realistic network settings to understand the fundamental and generalizable behavior of our schemes. While E2E encryption can detect packet tampering, it can not localize faults and also incurs logistical overheads like

key distribution, maintenance and protection. Therefore, when key-based logistics are expensive (in high mobility scenarios with high frequency of nodes joining and leaving the network), watchdogs provide a lightweight alternative. Watchdogs do not have to be specialized devices. Another strength is the deterrence factor that E2E encryption lacks. By adapting overheads and using source coding, our protocol achieved remarkable improvement in TCP performance amidst packet tampering. If watchdogs misbehave and accuse fault free nodes, they lose their credibility and can no longer cause any damage. We acknowledge weaknesses of watchdog based schemes in terms of security guarantee and overheads, however, there is no "one-solution-fits-all" here and we believe the problem is still interesting and that there is potential for further research in this direction. In future, we'll investigate overhead tuning, trust issues, dynamic coding rate adaptation, performance based route adaptation and work on more advanced watchdog based protocols to provide guaranteed-protection against misbehavior.

## APPENDIX

A random geometric graph (RGG) is a graph $G = (n, r) = (V, E)$ with $n = |V|$ such that nodes of V are embedded in an area with the property that $e = (u, v) \in E$ if and only if Euclidean distance between $u$ and $v$, $d(u, v) \leq r$ [18]. In wireless networks, $r$ relates directly to broadcast communication range of nodes and so RGGs are considered a standard model in theoretical work on ad-hoc and wireless networks [19]. The following definition is useful:

**Definition 1:** *Let $G = (n, r)$ be an RGG. $G$ is said to be $\mu$-*geo-dense *[20] for some $\mu \geq 1$, if every square bin of area $A \geq \frac{r^2}{\mu}$ has $\Theta(nA)$ nodes.*

Since critical radius for connectivity is $\pi(r_{con})^2 = \frac{\log n}{n}$ [19], it has been shown [20] that for constants $c > 1$ and $\mu \geq 2$, if $r^2 = \frac{c\mu \log n}{n}$, then w.h.p a random geometric graph $\zeta(n, r)$ is $\mu$-geo-dense. That is w.h.p, any bin area of size $r^2/\mu$ has $\Theta(\log n)$ nodes.

Using geo-density of RGG's just established, we come up with conditions required on network density to ensure availability of watchdogs. Let transmission range of nodes be $r_t \geq r_{con}$. Geo-density ensures that at least $\Theta(\log n)$ nodes will be present in every square of area $A \geq \frac{(r_{con})^2}{\mu}$. Then, $r_t = \sqrt{8A}$ will ensure watchdog availability w.h.p. Since this is the same transmission range (leading to similar interference constraints) as [21], we conclude that requirement for availability and success of watchdogs can be satisfied with the similar interference and transmission range constraints and therefore order of capacity bounds is preserved asymptotically.

## ACKNOWLEDGMENT

This research is supported in part by Army Research Office grant W-911-NF-0710287. Any opinions, findings, and conclusions or recommendations expressed here are those of the authors and do not necessarily reflect the views of the funding agencies or the U.S. government.

## REFERENCES

[1] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in *Proceedings of the 6th annual international conference on Mobile computing and networking*, ser. MobiCom '00, pp. 255–265.

[2] Q. He, D. Wu, and P. Khosla, "Sori: A secure and objective reputation-based incentive scheme for ad-hoc networks," in *In IEEE Wireless Communications and Neworking Conference*, 2004, pp. 825–830.

[3] S. Buchegger and J.-Y. Le Boudec, "Performance analysis of the confidant protocol," in *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, 2002, pp. 226–236.

[4] F. Martignon, S. Paris, and A. Capone, "A framework for detecting selfish misbehavior in wireless mesh community networks," in *Proceedings of the 5th ACM symposium on QoS and security for wireless and mobile networks*, ser. Q2SWinet '09, pp. 65–72.

[5] *Cross-layer Analysis for Detecting Wireless Misbehaviour*, January 2006.

[6] I. S. Han, H.-B. Ryou, and S.-J. Kang, "Multi-path security-aware routing protocol mechanism for ad hoc network," in *Proceedings of the 2006 International Conference on Hybrid Information Technology - Volume 01*, ser. ICHIT '06, pp. 620–626.

[7] S. S. Revoti Prasad Bora, Dheeraj Harihar, "Detection, penalization and handling of misbehavior in ad hoc wireless networks," in *In Proceedings of IMCES*, 2006, pp. 949–953.

[8] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft, "Xors in the air: practical wireless network coding," in *Proceedings on Applications, technologies, architectures, and protocols for computer communications*, ser. SIGCOMM '06.

[9] S. Biswas and R. Morris, "Exor: opportunistic multi-hop routing for wireless networks," in *Proceedings on Applications, technologies, architectures, and protocols for computer communications*, ser. SIGCOMM '05, pp. 133–144.

[10] K. Avramopoulos, K. A, and R. Wang, "opt and vent: An efficient protocol for byzantine detection in wireless ad hoc network routing," Princeton University, Dept. of Computer Science, Tech. Rep., 2004.

[11] S. Zhu, S. Setia, S. Jajodia, and P. Ning, "Interleaved hop-by-hop authentication against false data injection attacks in sensor networks," *ACM Trans. Sen. Netw.*, vol. 3, August 2007.

[12] G. Liang, R. Agarwal, and N. Vaidya, "When watchdog meets coding," in *INFOCOM'10: Proceedings of the 29th conference on Information communications*, pp. 2267–2275.

[13] P. Michiardi and R. Molva, "Core: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks," in *Sixth Joint Working Conference on Communications and Multimedia Security: Advanced Communications and Multimedia Security*, 2002.

[14] C. Zouridaki, B. L. Mark, M. Hejmo, and R. K. Thomas, "E-hermes: A robust cooperative trust establishment scheme for mobile ad hoc networks," *Ad Hoc Netw.*, vol. 7, pp. 1156–1168, 2009.

[15] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading structure for randomness in wireless opportunistic routing," in *Proceedings on Applications, technologies, architectures, and protocols for computer communications*, ser. SIGCOMM '07, pp. 169–180.

[16] J. K. Sundararajan, D. Shah, M. Medard, M. Mitzenmacher, and J. Barros, "Network coding meets tcp." in *INFOCOM'09*, 2009, pp. 280–288.

[17] Z. qun Xia, Z. gang Chen, Z. Ming, and J. qi Liu, "A multipath tcp based on network coding in wireless mesh networks," *Information Science and Engineering, International Conference on*, vol. 0, pp. 3946–3950, 2009.

[18] J. Domingo-Pascual, P. Manzoni, S. Palazzo, A. Pont, and C. M. Scoglio, Eds., *NETWORKING 2011 - 10th International IFIP TC 6 Networking Conference, Valencia, Spain, May 9-13, 2011, Proceedings, Part I*, ser. Lecture Notes in Computer Science, vol. 6640.

[19] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE TRANSACTIONS ON INFORMATION THEORY*, vol. 46, no. 2, pp. 388–404, 2000.

[20] C. Avin and G. Ercal, "On the cover time and mixing time of random geometric graphs," *Theor. Comput. Sci.*, vol. 380, July 2007.

[21] P. Kyasanur and N. H. Vaidya, "Capacity of multi-channel wireless networks: impact of number of channels and interfaces," in *Proceedings of the 11th annual international conference on Mobile computing and networking*, ser. MobiCom '05.