Securing the Backpressure Algorithm for Wireless Networks

Zhuo Lu, *Member, IEEE*, Yalin E. Sagduyu, *Senior Member, IEEE*, and Jason H. Li, *Member, IEEE*

Abstract—The backpressure algorithm is known to provide throughput optimality in routing and scheduling decisions for multi-hop networks with dynamic traffic. The essential assumption in the backpressure algorithm is that all nodes are benign and obey the algorithm rules governing the information exchange and underlying optimization needs. Nonetheless, such an assumption does not always hold in realistic scenarios, especially in the presence of security attacks with intent to disrupt network operations. In this paper, we propose a novel mechanism, called *virtual trust queuing*, to protect backpressure algorithm based routing and scheduling protocols against various insider threats. Our objective is *not* to design yet another trust-based routing to heuristically bargain security and performance, *but* to develop a generic solution with strong guarantees of attack resilience and throughput performance in the backpressure algorithm. To this end, we quantify a node's algorithm-compliance behavior over time and construct a virtual trust queue that maintains deviations of a give node from expected algorithm outcomes. We show that by jointly stabilizing the virtual trust queue and the real packet queue, the backpressure algorithm not only achieves resilience, but also sustains the throughput performance under an extensive set of security attacks. Our proposed solution clears a major barrier for practical deployment of backpressure algorithm for secure wireless applications.

Index Terms—Backpressure algorithm, routing and scheduling, attacks, security trust, virtual queue, insider threats, network optimization.

1 Introduction

The backpressure algorithm [1]–[4], in theory, achieves the optimal network throughput by dynamically routing and scheduling the network traffic. There have been significant efforts focused on translating and adapting the backpressure concept into a practical system for wireless networks [5]–[9].

In essence, the backpressure algorithm coordinates transmissions and maximizes the amount of total data delivery by adapting scheduling and routing decisions based on each node's per-flow queue backlogs and channel rates when applied to wireless networks. To this end, it presumes that all nodes obey the algorithm rules of information exchange, optimal link activation, and flow selection. However, in practice, a node may deliberately violate any rule to break the underlying premise assumed by the backpressure algorithm. Regardless of its selfish or malicious intent, there are two basic ways for an attacker to pursue: (i) it can falsify any information used in the backpressure algorithm; (ii) it can violate backpressure algorithm based protocols by offering no cooperation and/or not following decisions in routing and scheduling optimization. These potential attacks pose a major obstacle to practical deployment of the backpressure algorithm in real (especially wireless) systems.

As security emerges as a fundamental component of network design [10]–[15], it becomes vital to secure the backpressure algorithm against various security attacks. In the literature, integrating the backpressure algorithm into practical network applications [5]–[7], [11], [16]–[20] and securing routing protocols against various attacks [13], [14], [21]–[23] have been investigated rather orthogonally. Recently, a trust-based routing approach was designed in [11] to help the backpressure algorithm defend against attacks in wireless sensor networks. The focus in [11] was on heuristically balancing trust and throughput components in implementation, but not on providing security guarantee of attack resilience.

In this paper, we aim at *providing a generic framework to secure the backpressure algorithm with guaranteed resilience against information falsification and protocol-violation attacks*. To this end, we propose a novel mechanism, called *virtual trust queuing*. There are three key components supporting our mechanism: (i) when an attacker deviates from legitimate behavior, nodes can collectively observe and quantify such a deviation; (ii) each node individually manages all the deviations in a virtual queue for every neighbor node in a distributed setting, and then the virtual queue size can be limited (i.e., the deviation or damage of an attacker can be bounded) by using the queue-stabilizing technique jointly with a real packet queue; (iii) the service rate in a virtual queue can be adequately designed to mitigate imperfect observations (due to wireless effects) or false accusations (e.g., a benign node is mistakenly found to have abnormal behavior due to observation errors). Therefore, we show that the proposed virtual trust queue mechanism provides two performance guarantees: 1) *Zero penalty*: there is zero throughput performance penalty loss to use the virtual

Zhuo Lu is with the Department of Electrical Engineering, University of South Florida, Tampa FL 33620. Email: z.l.lu@ieee.org.
Yalin E. Sagduyu and Jason H. Li are with Intelligent Automation Inc., Rockville MD 20855. Emails: {ysagduyu, jli}@i-a-i.com.
An earlier version of the work was published in IEEE INFOCOM 2015.
Zhuo Lu was supported by NSF CNS-1464114.

trust queue in the backpressure algorithm when there is no attack; 2) *Impact bounding*: the damage of attacks to the network can be bounded by a given tolerance level for the virtual trust queue.

We use a comprehensive simulation study to show the effectiveness of the proposed mechanism against an extensive set of security attacks, including blackhole, selective-forwarding, on-off, and opportunity-wasting attacks, as well as selfish behavior to prioritize transmission opportunities. Our contributions are as follows: (i) we develop a new mechanism, called virtual trust queuing, to secure the backpressure algorithm; (ii) we show that the virtual trust queue mechanism, as a generic solution to secure the backpressure algorithm, offers guaranteed attack resilience as well as sustains the throughput performance for the backpressure algorithm; (iii) we demonstrate that the backpressure algorithm with virtual trust queuing is a viable secure solution for practical implementation of dynamic scheduling and routing in wireless networks; (iv) we conduct extensive simulations to show that the virtual trust queue mechanism can significantly improve the throughput performance by securing backpressure scheduling against a broad range of malicious attacks and selfish node behaviors.

The rest of this paper is organized as follows. In Section 2, we describe preliminaries and models for the backpressure algorithm and its vulnerabilities. In Sections 3 and 4, we design the virtual trust queue based backpressure algorithm and analyze its security properties, respectively. In Section 5, we present our simulation results. In Section 6, we discuss related work. Finally, we conclude the paper in Section 7.

2 Preliminaries: Backpressure Algorithm and Its Vulnerabilities

In this section, we use an example to introduce the backpressure algorithm and its vulnerabilities, then formulate the backpressure algorithm, and finally discuss attack models.

2.1 Example of Backpressure Algorithm and Vulnerabilities

The backpressure algorithm [1]–[4] is an optimal routing and scheduling policy that stabilizes packet queues with capability to achieve the maximum throughput. The backpressure algorithm dynamically selects the set of links to activate and flows to transmit on these links depending on queue backlogs and channel rates. In the following, we consider its application to a time-slotted wireless network.

Fig. 1 shows an example of how the backpressure algorithm works: nodes A, B, C, and D form a three-hop wireless network with two flows. Each node has the same transmission rate and cannot transmit and receive at the same time slot. At a given time slot, the backlog

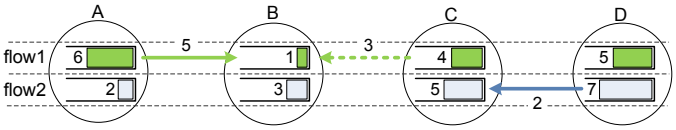

Fig. 1. Example of the backpressure algorithm.

of each node for each flow is illustrated in Fig. 1. The backpressure algorithm works as follows. First, compute the maximum differential queue backlog between each node pair as a link weight; i.e., A→B is 5 for flow 1, C→B is 3 for flow 1, and D→C is 2 for flow 2, and select these three links. Second, list all non-conflicting link sets, i.e., {A→B for flow 1, D→C for flow 2} and {C→B for flow 1}. Finally, choose the set that maximizes the sum of all link weights, i.e., {A→B for flow 1, D→C for flow 2}.

Now suppose node C is malicious and declares that its queue backlog for flow 1 is 100. Then, the maximum differential queue backlog between nodes B and C becomes 99, which makes backpressure scheduling choose only one link {C→B for flow 1}, thereby giving all the transmission opportunity to the malicious node C.

2.2 Backpressure Algorithm Formulation

More formally, we consider a network denoted by $(\mathcal{N}, \mathcal{F})$, where $\mathcal{N}$ and $\mathcal{F}$ are the sets of network nodes and flows, respectively. At time slot $t = 0, 1, 2, \cdots$, node $i \in \mathcal{N}$ has a queue backlog $Q_i^f(t)$ for flow $f \in \mathcal{F}$. The backpressure algorithm makes routing and scheduling decisions based on

$$\mathbf{u}^*(t) = \arg\max_{\mathbf{u}(t)\in\mathcal{R}(t)} \sum_{u_{i,j}(t)\in\mathbf{u}(t)} u_{i,j}(t) w_{i,j}(t), \tag{1}$$

where $u_{i,j}(t) \in \mathbf{u}(t)$ is the link rate from node i to j, $\mathbf{u}(t)$ is a feasible rate vector in the set of all feasible rate vectors $\mathcal{R}(t)$ in the network, and $w_{i,j}(t)$ is the maximum differential queue backlog

$$w_{i,j}(t) = \max_{f\in\mathcal{F}} (Q_i^f(t) - Q_j^f(t)), \tag{2}$$

where $Q_i^f(t)$ and $Q_j^f(t)$ are the backlogs for flow f at nodes i and j, respectively.

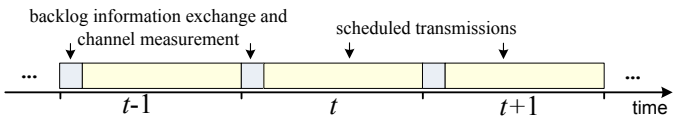

Fig. 2. Information exchange and transmission scheduling in the backpressure algorithm.

The backpressure algorithm in (1) is the optimal solution that requires centralized coordination. In practice, a centralized controller (e.g., [17]) will collect information from all nodes then make the scheduling decision. There also exist low-complexity, distributed solutions (e.g., [3], [5]–[7], [16]) with performance close to the optimal solution (1). As our focus is not to solve (1) optimally in a

distributed way, but to develop a generic framework that provides security guarantee integrated into the backpressure framework, we choose to integrate security into the optimal formulation (1). In other words, we consider a centralized scenario (e.g., [17]) in which there exists a centralized controller in a multi-hop wireless network. Accordingly, our theoretical results are based on the optimal backpressure scheduling formulation.

To this end, we adopt a generic implementation model for the backpressure algorithm shown in Fig. 2: at the beginning of each time slot, nodes send information to the controller for centralized coordination (e.g., [17]). The information includes queue backlogs for computing the differential queue backlog $w_{i,j}(t)$ in (2) and channel state information based on channel measurements for obtaining the best channel rate $u_{i,j}(t)$ from any node i to node j in (1). Then, scheduled transmissions occur at the rest of the time slot.

Note that our security solution based on the global optimization (1) does not require extra centralized or global information, but introduces new local information. Therefore, it can be readily extended to distributed versions that rely on exchange of local information only.

2.3 Insider Threats and Assumptions

We consider insider attackers that are nodes also involved in the routing and scheduling decisions in the network. In general, the behavior of an insider attacker can be classified to one or both of the following two categories.

1) *Information-falsification attack*: this happens during the information exchange phase at the beginning of each time slot shown in Fig. 2, where the attacker intentionally sends false information to others to adversely affect backpressure routing. For example, the attacker broadcasts false backlog information or false channel state information to result in wrong differential backlog computation in (2). As the backpressure algorithm is reactive to node queue backlogs and channel state information, its routing decisions can be significantly affected by information-falsification attacks.
2) *Protocol-violation attack*: this happens in the scheduled transmission phase shown in Fig. 2, where the attacker does not obey backpressure routing decisions. For example, an attacker does not transmit although it is scheduled to transmit at that time slot.

We assume that attackers can neither modify information (e.g., queue backlog) inside other nodes nor change the wireless channel characteristics (e.g., channel gain).

It is worth mentioning that the backpressure algorithm is resilient in terms of throughput performance to imperfect queue backlog estimation, as long as the error is bounded by a constant [24]. In this paper, we consider a different scenario, where attackers can arbitrarily manipulate queue backlogs and channel state information, and launch various attacks using falsified information to degrade the network performance.

3 Securing the Backpressure Algorithm

An attacker can launch either information-falsification or protocol-violation attacks, or both. To clearly present our solution, we first handle information-falsification attacks. In particular, we discuss how a node can observe information-falsification attacks, then we design our solution to secure the backpressure algorithm. Finally, we extend our solution to protocol-violation attacks.

3.1 Behavior of Attackers

We first address information-falsification attacks. Such attacks can have at least one of two intents: (i) *selfish behavior*: if the attacker is selfish, it is interested in its own performance gain without care for others in the network; (ii) *malicious behavior*: if the attacker is malicious, it aims to degrade the throughput of others in the network as much as possible.

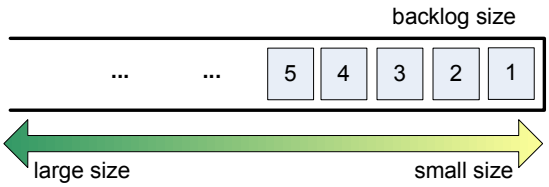

Fig. 3. Low or high backlog broadcasting can be used to affect backpressure routing.

As the backpressure algorithm requires nodes to broadcast their queue backlogs and channel state information, one effective way for an attacker to fulfill its selfish or malicious intent is to falsify its queue backlogs or channel state information.

1) *Manipulating backlogs*. If the attacker wants to send its own packets immediately instead of receiving packets from others, it can broadcast falsified higher backlogs than actual ones as shown in Fig. 3. Then, it has a higher chance to be scheduled to transmit as a result of backpressure algorithm computation. If the attacker is malicious and tries to destroy packet delivery as much as possible, it can act like a blackhole and broadcast falsified, lower or zero, backlogs. Thus, it will attract more packets routed to itself under backpressure scheduling, then drop some or all of them. In summary, an attacker can manipulate its backlog information arbitrarily to affect the optimization solution in the backpressure algorithm.
2) *Falsifying channel state information*. If the attacker wants to gain the transmission opportunity, it can broadcast higher channel gains than the actual ones (so its link rate $u_{i,j}(t)$ is considered large in the backpressure optimization (1)). Then, it has

a higher chance to be scheduled to transmit. Although broadcasting false channel information is one type of information falsification, we can categorize attacks that falsify channel state information into protocol-violation attacks. This is because when an attacker cannot transmit with a claimed rate, it violates the scheduling decision. We will show in Section 3.4 that channel state information falsification can be solved jointly with other protocol-violation attacks.

In summary, manipulation of queue backlogs belongs to information-falsification attacks. An attacker can also simply manipulate its backlog information arbitrarily to affect the optimization solution in the backpressure algorithm.

3.2 Identification of Attack Behavior

Backlog manipulation is an essential way to launch various attacks against the backpressure algorithm. The question is how to identify or deduce possible backlog manipulation behavior of a node. To answer this, we first need to understand how the backlog dynamics evolve for benign nodes over time.

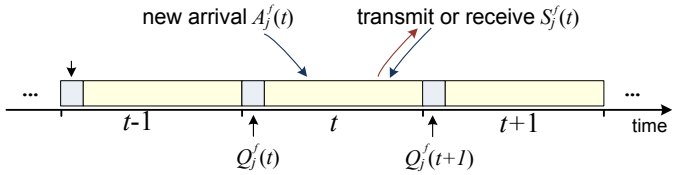

Fig. 4. Backlogs and activities of a node as time evolves.

Suppose node j is benign and obeys the backpressure scheduling. Fig. 4 shows the behavior of node j over time. Node j's backlogs for flow f are denoted by $Q_j^f(t)$ and $Q_j^f(t+1)$ at times t and $t+1$, respectively. The number of new arrivals at node j for flow f is $A_j^f(t)$ during time slot t, whose value is only known to node j and is never broadcast to others. In addition, node j may transmit or receive at time t. Let $S_j^f(t)$ be the amount of data that node j sends or receives during time t for flow f ($S_j^f(t) > 0$ or $S_j^f(t) < 0$ when node j transmits or receives for flow f, respectively). Then, it must hold that node j's next-slot backlog $Q_j^f(t+1)$ equals to its current-slot backlog $Q_j^f(t)$ with its new arrivals $A_j^f(t)$ added and its transmitted data $S_j^f(t)$ removed; i.e.,

$$Q_j^f(t+1) = Q_j^f(t) + A_j^f(t) - S_j^f(t) \tag{3}$$

for each flow $f \in \mathcal{F}$. Note that an implicit assumption in the backpressure algorithm is that all nodes know the source and destination of a particular flow. Therefore, a destination node is known to consume the transmitted data on flows with destination to it. As a result, any consumed data at a destination is not reflected in (3).

From (3), we can get the arrival rate $A_j^f(t)$ as

$$A_j^f(t) = Q_j^f(t+1) - Q_j^f(t) + S_j^f(t) \in [0, A_{\max}], \tag{4}$$

where $A_{\max}$ is the upper limit of the packet arrival rate in the network, depending on a particular application. The value of $A_{\max}$ can be set according to application-layer specifications or a wireless node's link capability. For example, when a network uses G.729 [25] (that is a voice over IP protocol), the packet arrival rate can be 50 or 33.3 packets/second and $A_{\max}$ can be set to be 50 plus a margin (e.g., 5 packets) to tolerant observation errors; when a network uses ZigBee with date rate of 250 kbit/s and the data packet size is 1500 bytes, $A_{\max}$ can be set to be $250000/(1500*8) \approx 21$ also plus a margin of a small number of packets.

The purpose of obtaining the arrival rate is that we will show that any potential backlog manipulation at node j can be identified by a neighbor node i that examines whether node i's estimate of node j's arrival rate estimate is within $[0, A_{\max}]$.

In particular, node j may not be benign and can broadcast false backlogs $\hat{Q}_j^f(t+1)$ and $\hat{Q}_j^f(t)$, instead of its true backlogs $Q_j^f(t+1)$ and $Q_j^f(t)$, respectively. The transmitted/received data $S_j^f(t)$ can be observed as $\hat{S}_{i,j}^f(t)$ by node i.

Due to the coordinated nature of the backpressure algorithm, if node j is scheduled to transmit, a neighbor of node j will know the amount of data $S_j^f(t)$ that node j transmits. Thus, node i's perfect observation is $\hat{S}_{i,j}^f(t) = S_j^f(t)$ if the attacker obeys the scheduling decision to transmit. However, due to wireless channel fading and collision, node i's observation $\hat{S}_{i,j}^f(t)$ may not be exactly equal to $S_j^f(t)$ or and the difference may be large. Moreover, protocol-violation attacks do not need to obey scheduling decisions, which can also lead to $\hat{S}_{i,j}^f(t) \neq S_j^f(t)$. We will handle the imperfect observation case in Section 3.3.2 and protocol-violation attacks in Section 3.4.

For better presentation of our idea, we assume that $\hat{S}_{i,j}^f(t) = S_j^f(t)$ in the following. Thus, given $\hat{Q}_j^f(t+1)$, $\hat{Q}_j^f(t)$ and $\hat{S}_{i,j}^f(t)$, node i (observing node j's transmit/receive behavior) can estimate node j's arrival rate as

$$\hat{A}_{i,j}^f(t) = \hat{Q}_j^f(t+1) - \hat{Q}_j^f(t) + \hat{S}_{i,j}^f(t). \tag{5}$$

We show via examples how node j's selfish or malicious behavior results in node i's arrival rate estimate $\hat{A}_{i,j}^f(t)$ going outside of the normal region $[0, A_{\max}]$.

- *Negative arrival rate*: consider that node j is a black-hole attacker who intends to attract packets to be routed to itself and drop all these packets. To achieve this goal efficiently, node j keeps broadcasting zero backlogs (i.e., $\hat{Q}_j^f(t) = 0$ for all t). Obviously, because of its zero backlog, it should always receive data from another node, and this effect is observed by its neighbor node i as $\hat{S}_{i,j}^f(t) < 0$. It follows that the arrival rate estimate at node i satisfies

$$\hat{A}_{i,j}^f(t) = \hat{Q}_j^f(t+1) - \hat{Q}_j^f(t) + \hat{S}_{i,j}^f(t) = 0 - 0 + \hat{S}_{i,j}^f(t) < 0.$$

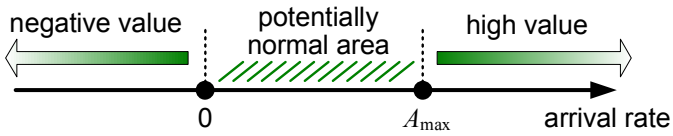

Fig. 5. Negative and high arrival rates indicate attacking behavior.

Accordingly, we see that if a node is deliberately attracting packets, it can exhibit a negative arrival rate.

- *High arrival rate*: consider that node j does not get the chance to transmit at time t (i.e., $\hat{S}^f_{i,j}(t) = 0$ observed by node i), but wants to capture the transmission opportunity at time $t+1$. Thus, it broadcasts a much higher backlog at the start of time $t+1$ (i.e., $\hat{Q}^f_j(t+1) > \hat{Q}^f_j(t) + A_{\max}$). Consequently, its arrival rate estimate can be written as

$$\hat{A}^f_{i,j}(t) = \hat{Q}^f_j(t+1) - \hat{Q}^f_j(t) + \hat{S}^f_{i,j}(t) > A_{\max} + 0 > A_{\max},$$

indicating the arrival rate estimate exhibits a very large value that exceeds its limit $A_{\max}$.

It is easy to verify that either negative or high arrival rate indicates attacking behavior as shown in Fig. 5. Therefore, we say that node i's arrival rate estimate $\hat{A}^f_{i,j}(t)$ of node j for flow f is within the normal region if $0 \le \hat{A}^f_{i,j}(t) \le A_{\max}$.

3.3 Virtual Trust Queue to Defend Backpressure Algorithm

We have already identified that estimating packet arrival rate of a node can help us find out whether the node exhibits backlog manipulation behavior or not. Our next goal is to design a strategy based on estimating packet arrival rates to defend the backpressure algorithm. We first introduce an augmented optimization approach to protect the backpressure algorithm, then present how to build a comprehensive virtual trust queue solution to provide the security guarantee.

3.3.1 Augmented Optimization Approach

As we have already mentioned, the arrival rate estimate $\hat{A}^f_{i,j}(t)$ is critical to determine whether a node is an attacker or not at time t. Either $\hat{A}^f_{i,j}(t) < 0$ or $\hat{A}^f_{i,j}(t) > A_{\max}$ indicates attacking behavior. As shown in Fig. 5, the farther the estimate of node j's arrival rate is from the actual one, the more aggressive the attack behavior becomes, and accordingly there is less trust that another node should put in node j.

Thus, we define the *deviation in arrival* of node j for flow f observed at node i as the distance of arrival rate estimate $\hat{A}^f_{i,j}(t)$ to the normal arrival rate region $[0, A_{\max}]$; namely,

$$d^f_{i,j}(t) = \begin{cases} |\hat{A}^f_{i,j}(t)| & \text{for } \hat{A}^f_{i,j}(t) < 0, \\ 0 & \text{for } 0 \le \hat{A}^f_{i,j}(t) \le A_{\max}, \\ \hat{A}^f_{i,j}(t) - A_{\max} & \text{for } \hat{A}^f_{i,j}(t) > A_{\max}. \end{cases} \tag{6}$$

It is easy to see that $d^f_{i,j}(t)$ is always non-negative. Larger $d^f_{i,j}(t)$ means more deviation from a node's normal behavior. Then, we define the deviation in arrival (for all flows) as

$$D_{i,j}(t) = \sum_{f \in \mathcal{F}} d^f_{i,j}(t) + \epsilon, \tag{7}$$

where $\epsilon > 0$ is a small number serving only as a uniform offset such that all behaviors exhibit strictly positive deviations, i.e., $D_{i,j}(t) \ge \epsilon$ for all $t > 0$.

Accordingly, increasing $D_{i,j}(t)$ decreases node i's trust in node j. If node j is associated with a high value of $D_{i,j}(t)$, it should be attacking other nodes or misbehaving in the network, and thus should be punished during the backpressure scheduling. Thus, the augmented optimization approach adds a penalty term to the backpressure algorithm in (1) as

$$\mathbf{u}^*(t) = \arg\max_{\mathbf{u}(t) \in \mathcal{R}(t)} \sum_{u_{i,j}(t) \in \mathbf{u}(t)} \left(u_{i,j}(t) w_{i,j}(t) - v D_{i,j}(t) \right), \tag{8}$$

where $-vD_{i,j}(t)$ ($v > 0$) is the penalty term. If a node is an attacker with a large value of $D_{i,j}$, the penalty term $-vD_{i,j}(t)$ is then a large negative value, indicating that the link (i,j) is less likely to be scheduled by the maximization in (8). Here, $v > 0$ is a weighted factor for the deviation in arrival, meaning the optimization allows for a flexible tradeoff between the throughput performance and the trust level by adjusting the value of v.

3.3.2 Virtual Trust Queue

There are three major drawbacks of the approach in (8): (i) if an attacker causes a very large value of $D_{i,j}(t)$ at time t (e.g., deliberately dropping all packets) and then returns to legitimate behavior after time t, the penalty in (8) only happens and lasts during time t (i.e., there is no memory in tracking the trust), and therefore may not mitigate the total damage of the attack; (ii) there is no systematic way to determine the value of v; and (iii) there is no systematic way to know, control, or limit the damage that an attacker can cause to the network performance.

To address the first issue, we can define a sliding window to record the history and keep applying the penalty. However, the sliding window method requires careful adjustment of window size and still cannot solve the second and third issues. Another way to remember and use history is to represent it in a queue structure. Mathematical tools in the queuing theory can then provide a theoretical understanding of how we can use a queue to limit an attack's behavior.

As a consequence, we use a queue to manage the deviations of arrival. Motivated by [18] that uses a virtual energy queue to limit the power consumption while at the same time sustaining the throughput performance, our objective is to construct a virtual queue, called *virtual trust queue*, to limit the damage of an attacker while at the same time sustain the throughput performance. We call it virtual because the queue only stores a single trust value. In particular, node i maintains a virtual trust queue for node j and enqueues the deviation in arrival $D_{i,j}(t)$ with constant service rate $\delta > \epsilon > 0$. In other words, the queue size (or the value stored in the queue) $X_{i,j}(t)$ can be written as

$$X_{i,j}(t+1) = \max\left(X_{i,j}(t) - \delta, 0\right) + D_{i,j}(t). \quad (9)$$

It is easy to see that if $X_{i,j}(t)$ becomes larger, node j is less trustable. We then integrate the virtual trust queue into (1) in the backpressure algorithm as

$$\mathbf{u}^*(t) = \arg\max_{\mathbf{u}(t)\in\mathcal{R}(t)} \sum_{u_{i,j}(t)\in\mathbf{u}(t)} \left(u_{i,j}(t)w_{i,j}(t) - X_{i,j}(t)D_{i,j}(t)\right). \quad (10)$$

We next show that (10) provides two essential guarantees, called zero penalty and attack bounding.

- *Zero penalty*: if there is no attack, there is no throughput performance penalty loss to use the virtual trust queue.
- *Impact bounding*: if there is an attack, the virtual trust queue based optimization (10) guarantees the attack's damage to the network is always bounded from above.

In what follows, the two guarantees of the virtual trust queue are presented in Theorems 1 and 2, respectively.

Theorem 1 (Zero Penalty of the Virtual Trust Queue): In a network without any attack, the virtual trust queue based optimization (10) is equivalent to the original backpressure optimization in (1).

Proof: If there is no attack, $D_{i,j}(t) = \epsilon$ for all $t > 0$. Thus, the virtual trust queue size in (9) satisfies $X_{i,j}(1) = \max(0-\delta, 0) + \epsilon = \epsilon$, $X_{i,j}(2) = \max(\epsilon - \delta, 0) + \epsilon = \epsilon$, $X_{i,j}(3) = \max(\epsilon-\delta, 0)+\epsilon = \epsilon$, and so on. Then, $X_{i,j}(t) = \epsilon$ for all $t > 0$.

Inserting $D_{i,j}(t) = X_{i,j}(t) = \epsilon$ into (10) yields

$$\begin{aligned}\mathbf{u}^*(t) &= \arg\max_{\mathbf{u}(t)\in\mathcal{R}(t)} \sum_{u_{i,j}(t)\in\mathbf{u}(t)} \left(u_{i,j}(t)w_{i,j}(t) - \epsilon^2\right) \\ &= \arg\max_{\mathbf{u}(t)\in\mathcal{R}(t)} \sum_{u_{i,j}(t)\in\mathbf{u}(t)} u_{i,j}(t)w_{i,j}(t), \quad (11)\end{aligned}$$

which completes the proof, since (20) is equivalent to (1). □

Remark 1: Theorem 1 ensures that we can always use this virtual trust queue mechanism under any circumstance with and without attacks. It incurs no cost in terms of performance loss when there is no attack in the network. Note that Theorem (1) does not consider the cost induced by communication overhead. The virtual trust queue requires transmissions of information of virtual queue sizes in addition to other communication overhead induced by the original backpressure algorithm. Therefore, it indeed incurs some extra communication overhead in a practical system.

Theorem 2 (Impact Bounding of the Virtual Trust Queue): If a virtual trust queue established in (9) is stable, it is guaranteed that the deviation in arrival $D_{i,j}(t)$ satisfies

$$\limsup_{t\to\infty} \frac{1}{t}\sum_{\tau=1}^{t} \mathbb{E}(D_{i,j}(\tau)) \le \delta. \quad (12)$$

Otherwise, (10) degenerates to an optimization over a subset that excludes from stable rate region $\mathcal{R}(t)$ all links with nodes showing unstable queues as $t \to \infty$.

Proof: The proof consists of two parts: (i) when the virtual queue is stable, and (ii) when the virtual queue is not stable.

Part I: when the virtual queue is stable, it satisfies

$$\limsup_{t\to\infty} \frac{1}{t}\sum_{\tau=1}^{t} \mathbb{E}(X_{i,j}(\tau)) \le \infty. \quad (13)$$

Then, given the relation between queue size and deviation of arrival in (9), it follows from Lemma 3 in [18] that

$$\limsup_{t\to\infty} \frac{1}{t}\sum_{\tau=1}^{t} \mathbb{E}(D_{i,j}(\tau)) \le \delta, \quad (14)$$

which completes the first part.

Part II: when a particular virtual queue $X_{i_0,j_0}(t)$ is unstable, we have

$$\liminf_{t\to\infty} X_{i_0,j_0}(t) = \infty. \quad (15)$$

If link (i_0, j_0) is selected, it follows from (15) that

$$-X_{i_0,j_0}(t)D_{i_0,j_0}(t) = -\infty \quad (16)$$

as $t \to \infty$. This means if link (i_0, j_0) is selected, the term $\sum_{u_{i,j}(t)\in\mathbf{u}(t)} \left(u_{i,j}(t)w_{i,j}(t) - X_{i,j}(t)D_{i,j}(t)\right)$ in (10) becomes $-\infty$. Therefore, an optimal solution to (10) is to find the best link rates in all feasible link sets excluding link (i_0, j_0). □

Remark 2: The virtual trust queue $X_{i,j}(t)$ saved at node i is not a real queue, but a single value whose operation in (9) works similar to a queue. Therefore, maintaining a virtual trust queue only incurs negligible amount of memory at each node. Similarly, broadcasting the trust queue information also incurs negligible amount of additional transmission overhead.

3.3.3 Choosing the Service Rate in Trust Queue

Theorem 2 states that the designed virtual trust queue produces two outcomes for an attacker: 1) if the attacker exhibits mild behavior (e.g., alternating legitimate and selfish behaviors over time) such that the virtual trust queue is still stable (i.e., queue size is small and bounded), we guarantee that the attacker's average deviation in arrival is below the service rate δ. That is, δ serves as a tolerance level to the attacker. 2) if the attacker's behavior is too aggressive (e.g., keeps

attacking all the time) such that the virtual trust queue becomes unstable (i.e., queue size is very large and keeps increasing), we guarantee that the attacker is eliminated from any routing decision.

It is emphasized that our goal is not to detect any possible attack against the backpressure algorithm, but to sustain performance for all legitimate nodes under attacks that can significantly degrade the performance. Given our virtual trust queue mechanism with a tolerance level, there will be some greedy attackers that achieve slight gains without being detected. However, they will be immediately penalized once they go beyond the tolerance level.

We can specify any positive value of the service rate δ in the virtual trust queue, which reflects our tolerance level or bound for attackers as indicated in (14). The virtual trust queue mechanism guarantees that it involves any potential attackers in scheduling under the given tolerance level δ, and at the same time eliminates any potential attackers that go beyond δ.

As we have mentioned, the virtual trust queue mechanism is based on the observations on other nodes, which may have errors in the real world. Such errors may also lead to false accusation to some benign nodes. Therefore, the value of δ can be set proportional to the expected level of observation errors in a practical network scenario to mitigate observation errors. For example, due to possible channel collision, a node may not be able to observe a transmission. Therefore, δ can be set as the amount of data for a single transmission such that the observation error due to channel collision is mitigated by the virtual queue service.

3.4 Handling Protocol-Violation Attacks

To defend against backlog information falsification attacks, we have so far built a virtual trust queue mechanism, in which we assume that an attacker only falsifies backlog information to affect scheduling decisions, but it obeys any scheduling decision. In practice, an attacker has the freedom to both falsify information and violate any scheduling decision. We next show our solution can be directly integrated to combat protocol-violation attacks at the same time.

A protocol-violation attacker is the one that does not obey the backpressure scheduling decision, such as (i) a selective-forwarding attacker that does not transmit packets for a particular flow even when it is scheduled to transmit, (ii) an attacker that claims to have a very good channel rate but uses a much lower rate to transmit, or (iii) an attacker that transmits data to another node that is not scheduled to receive.

To defend against protocol-violation attacks, we adopt a similar neighborhood watch strategy that we used previously for queue backlogs. As shown in Fig. 6, node j is scheduled to transmit data with amount $S_j^f(t)$, which is known to its neighbor node i during the information exchange in backpressure scheduling. If node j obeys the scheduling decision, then node i's observation on the transmission is $\hat{S}_{i,j}^f(t) = S_j^f(t)$. Otherwise, $\hat{S}_{i,j}^f(t) \neq S_j^f(t)$. For example, if node j does not transmit or transmits to a wrong destination, $\hat{S}_{i,j}^f(t) = 0$. If node j uses a lower rate to transmit, $\hat{S}_{i,j}^f(t) < S_j^f(t)$. Note that in this case, imperfect observation or false accusation may also happen, which can also be mitigated by choosing the value of service rate equal to the data amount for a single transmission.

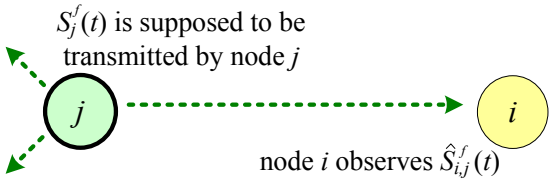

Fig. 6. Node j's scheduled transmission vs. node i's observation.

As a result, we define the *deviation in scheduling* as the absolute difference between node i's observation $\hat{S}_{i,j}^f(t)$ and node j's scheduling $S_j^f(t)$, i.e.,

$$E_{i,j}(t) = \sum_{f \in \mathcal{F}} |\hat{S}_{i,j}^f(t) - S_j^f(t)| + \eta, \tag{17}$$

where $\eta > 0$ is a small offset similar to previously defined ϵ in (7). It is obvious that if a node exhibits large deviations of scheduling in (17), the node may misbehave and should be penalized in the backpressure algorithm.

In a similar way, we construct another virtual trust queue: node i maintains a second virtual trust queue for node j with size $Y_{i,j}(t)$ that enqueues the deviation in scheduling $E_{i,j}(t)$ with constant service rate $\sigma > \eta > 0$; i.e.,

$$Y_{i,j}(t+1) = \max\left(Y_{i,j}(t) - \sigma, 0\right) + E_{i,j}(t). \tag{18}$$

Finally, in order to handle both information-falsification and protocol-violation attacks, we integrate the two virtual trust queues together into the backpressure optimization as

$$\mathbf{u}^*(t) = \arg\max_{\mathbf{u}(t) \in \mathcal{R}(t)} \sum_{u_{i,j}(t) \in \mathbf{u}(t)} (u_{i,j}(t) w_{i,j}(t) - X_{i,j}(t) D_{i,j}(t) - Y_{i,j}(t) E_{i,j}(t)), \tag{19}$$

where $X_{i,j}(t)$ are $Y_{i,j}(t)$ are the virtual trust queues for deviation in arrival $D_{i,j}(t)$ and deviation in scheduling $E_{i,j}(t)$, respectively.

In the following, we show that Theorems 1 and 2 still hold when we integrate two virtual trust queues in (19).

Corollary 1: In a network without any attack, the virtual trust queue based optimization (19) is equivalent to the original backpressure optimization in (1).

Proof: If there is no attack, $E_{i,j}(t) = \epsilon$ for all $t > 0$. Thus, the virtual trust queue size in (9) satisfies $Y_{i,j}(1) = \max(0 - \sigma, 0) + \eta = \epsilon$, $Y_{i,j}(2) = \max(\eta - \sigma, 0) + \eta = \eta$, and so on. Then, $Y_{i,j}(t) = \eta$ for all $t > 0$.

Inserting $D_{i,j}(t) = X_{i,j}(t) = \epsilon$ (already shown in the proof of Theorem 1) and $E_{i,j}(t) = Y_{i,j}(t) = \eta$ into (19) yields

$$\begin{aligned}\mathbf{u}^*(t) &= \arg\max_{\mathbf{u}(t)\in\mathcal{R}(t)} \sum_{u_{i,j}(t)\in\mathbf{u}(t)} \left(u_{i,j}(t)w_{i,j}(t) - \epsilon^2 - \eta^2\right) \\ &= \arg\max_{\mathbf{u}(t)\in\mathcal{R}(t)} \sum_{u_{i,j}(t)\in\mathbf{u}(t)} u_{i,j}(t)w_{i,j}(t), \qquad (20)\end{aligned}$$

which completes the proof. □

Corollary 2: If the two virtual trust queues in (19) are stable, then $D_{i,j}(t)$ and $E_{i,j}(t)$ satisfies $\limsup_{t\to\infty} \frac{1}{t}\sum_{\tau=1}^{t} \mathbb{E}(D_{i,j}(\tau)) \leq \delta$, and $\limsup_{t\to\infty} \frac{1}{t}\sum_{\tau=1}^{t} \mathbb{E}(E_{i,j}(\tau)) \leq \sigma$. Otherwise, (19) degenerates to an optimization over a subset that excludes from stable rate region $\mathcal{R}(t)$ all links with nodes showing unstable queues as $t \to \infty$.

Proof: Similar to the proof of Theorem 2, the virtual queue being stable indicates

$$\limsup_{t\to\infty} \frac{1}{t}\sum_{\tau=1}^{t} \mathbb{E}(Y_{i,j}(\tau)) \leq \infty. \qquad (21)$$

Then, it follows from Lemma 3 in [18] that $\limsup_{t\to\infty} \frac{1}{t}\sum_{\tau=1}^{t} \mathbb{E}(E_{i,j}(\tau)) \leq \sigma$. If any virtual queue in not stable, either $-X_{i,j}(t)D_{i,j}(t)$ or $-Y_{i,j}(t)E_{i,j}(t)$ will be negative infinity as $t \to \infty$, and therefore they are excluded in the $\arg\max$ optimization in (19). □

According to the two corollaries, the designed virtual trust queue mechanism provides a guaranteed (rather than heuristic) solution to defend the backpressure algorithm against both information-falsification and protocol-violation attacks. Moreover, the proposed virtual trust queue mechanism is generic and flexible to accommodate more queues for the quantification of any other attack behavior, and at the same time guarantees attack resilience.

3.5 Discussions

It is possible for an attacker to perform low-rate attacks to bypass the virtual trust queue mechanism without penalty. Consider an attacker in Fig. 7: attacker e wants to degrade the performance. It creates a junk flow to destination d with flow arrival rate smaller than $A_{\max}$. Then, it broadcasts an increased backlog for this flow to neighboring nodes a and b. Because the flow arrival rate is smaller than $A_{\max}$, the deviation of arrival is always 0 according to (6). In this way, as long as attacker e also transmits the same amount of (junk) data for such a flow under backpressure scheduling to avoid protocol violation, it will not be detected and punished under the virtual trust queue mechanism.

It is worth noting that from the view at the network level, such a low-rate attacker exhibits no difference from a normal node maintaining a legitimate flow to another node: the attacker indeed has legitimate behavior because it drops no packets, obeys the scheduling protocol, and maintains network traffic flows with legitimate rates. Therefore, it cannot be detected by any defense methods deployed at the network layer or below, including our virtual trust queue mechanism that is used at the network layer to defend against attacks. The only difference between low-rate attack and legitimate flows is that the application layer at the destination will find data from the attacker does not have suitable content (i.e., detecting junk data).

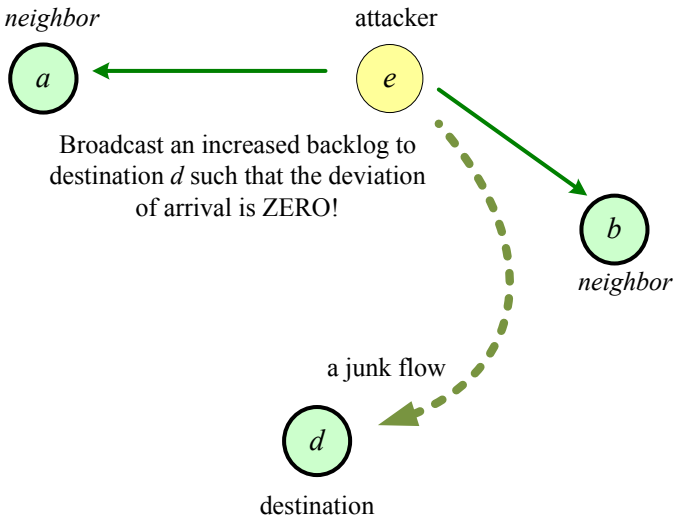

Fig. 7. Low rate attack scenario: attacker e is broadcasting an increased backlog. The backlog is carefully manipulated such that the deviation of arrival is still calculated as zero at neighboring nodes a and b.

Hence, our virtual trust queue mechanism never detects low-rate attacks, not due to design flaws, but because of the deployment nature of all network layer defense. It has to work with existing attack detection methods at the application layer (e.g., [26], [27]) to completely eliminate such low-rate attacks.

In this paper, we focus on insider attacks. There may exist some outsider attackers in the network. Usually an outsider attacker, like a jammer, attempts to disrupt communications in a network. This apparently will cause packet transmission disruptions. Therefore, nodes cannot correctly receive packets or observe their neighbors. The proposed secure backpressure framework cannot defend against such an outsider attack. Therefore our secure backpressure framework should work with defense mechanisms against outsider attacks to fully secure a wireless network.

4 Falsifying Virtual Trust Queue Information and Collusion Attacks

The virtual trust queue mechanism uses (19) to coordinate node transmissions. On one hand, virtual trust queues provide attack resilience; on the other hand, they may introduce another line of vulnerability in the backpressure algorithm. In particular, nodes need to broadcast additional virtual trust queue information

$X_{i,j}(t)$, $D_{i,j}(t)$, $Y_{i,j}(t)$, and $E_{i,j}(t)$ for either distributed or centralized coordination at time t. Nonetheless, it is possible that an attacker can also falsify virtual trust queue information to blame a legitimate node for misbehavior. Or even worse, two or more attackers can collude with each other to make an undetectable scenario, in which one attacker is attacking the network by manipulating information or violating the protocol, and at the same time other attackers obey the schedule but send falsified trust queue information to cover for the attacker. See the example in Fig. 8 for such a case.

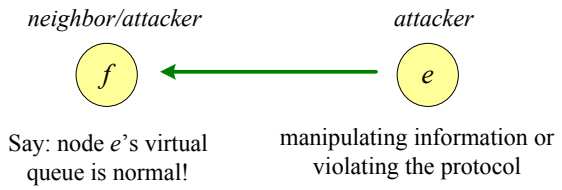

Fig. 8. Two nodes can collude with each other. Node e is attacking the network and at the same time, its partner node f declares that node e's virtual trust queue is normal.

These attacks can be all realized via the same strategy of falsifying virtual trust queue information. Therefore, it is necessary to address such attacks with an effective countermeasure.

4.1 Countermeasure Design

Our countermeasure is based on the idea of comparing different virtual trust queue information received from different nodes. To illustrate our idea, let us focus on the metric of deviation in arrival $D_{i,j}(t)$ and its virtual trust queue $Y_{i,j}$.

First, notice that $D_{i,j}(t)$ reflects node i's observation on node j. This indicates that if node i is an attacker, it can only falsify its own $D_{i,j}(t)$ on node j. However, node i is not the only one providing such information. There are other neighbors that provide their observations on node j at the same time. Denote by $\mathcal{N}_j$ the set of node j's neighbors. Then, there are $|\mathcal{N}_j|$ observations on node j. For any pair of benign nodes $k, l \in \mathcal{N}_j$, they receive the same backlog information from node j and can observe the transmission behavior of node j. Thus, their quantified deviations in arrival $D_{k,j}(t)$ and $D_{l,j}(t)$ should be in close value (due to some observation errors in practice). Accordingly, $X_{k,j}(t)$ and $X_{l,j}(t)$ should also be in close value (since the queue size is solely based on the deviation in arrival as shown in (9)). Therefore, any significant difference among all reported virtual trust queue sizes indicates the behavior of queue size falsification.

For a centralized implementation (e.g., [17]), a virtual trust queue does not need to be maintained at each node. The centralized controller can establish the queue for each node. At each time slot, node k reports its computed deviation $D_{k,j}(t)$ to the controller. Then, the controller chooses the most uniformly agreed value using (22) for its trust queue maintained for node j, i.e., the controller adopts *a nearest-to-average rule* to correct any isolated values in all reported virtual trust queue information using

$$D_{k,j}(t) = \underset{d \in \mathcal{D}_j(t)}{\arg\min} \; |d - D_j(t)| \tag{22}$$

for all $k \in \mathcal{N}_j$, where $\mathcal{D}_j(t)$ is the set of all deviations of arrival reported by node j's neighbors, and $D_j(t)$ is the average of all these deviations. Similar corrections will also be applied to $X_{i,j}$, $E_{i,j}$, and $Y_{i,j}$.

The nearest-to-average rule in (22) is essentially a majority rule. In other words, the rule will choose the virtual queue size that is reported by a majority of a node's neighbors (without considering observation errors).

Fig. 9 shows an example how the correction to falsified virtual trust queue information works. Suppose that node i sends false information ($D_{i,j}(t) = 9, X_{i,j}(t) = 2$). At the same time, node i's other benign neighbors, nodes a, b, c and d, also send $(D_{a,j}(t), X_{a,j}(t))$, $(D_{b,j}(t), X_{b,j}(t))$, $(D_{c,j}(t), X_{c,j}(t))$, and $(D_{d,j}(t), X_{d,j}(t))$, respectively, which satisfy $D_{i,j}(t) \neq D_{a,j}(t) = D_{a,j}(t) = D_{b,j}(t) = D_{c,j}(t) = D_{d,j}(t) = 5$ and $X_{i,j}(t) \neq D_{a,j}(t) = X_{a,j}(t) = X_{b,j}(t) = X_{c,j}(t) = X_{d,j}(t) = 20$. Using (22), the corrections are $D_{i,j}(t) = 5$ and $X_{i,j}(t) = 20$.

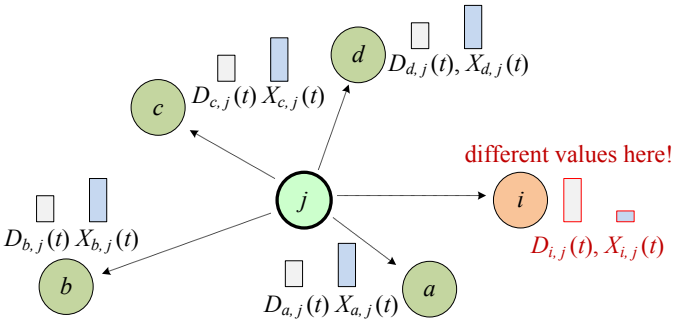

Fig. 9. Node j has neighbor nodes i, a, b, c, and d. Node i falsifies virtual trust queue information. Nodes a, b, c, and d are benign and send correct information.

4.2 Performance Analysis

The nearest-to-average rule may choose a wrong value if a node has more colluding neighbors than benign neighbors. Intuitively, in a scenario where nodes are mobile or randomly distributed, this may be less likely to happen under the typical assumption that there are more benign nodes than attackers in the network. But if the case indeed happens, we are interested in determining the potential effect of attack, namely the probability of a wrong detection decision. By adopting the widely-used random geometric graph network model [28], we state our result in the following proposition.

Theorem 3: Consider a network modeled as a random geometric graph with n nodes and node density λ. Assume that there are $(1 - c)n$ legitimate nodes and

cn colluding attackers ($c \in (0,1)$). The probability that the nearest-to-average rule fails to detect a colluding attack is bounded from above by $e^{-a\left(\sqrt{c}-\sqrt{1-c}\right)^2}$, where $a = \pi\lambda r^2$ is the average number of legitimate neighbors for each node in the network.

Proof: The nearest-to-average rule fails to detect an attacker when the number of its colluding neighbors n_a is greater than that of its legitimate neighbors n_n. Therefore, the probability that the nearest-to-average rule fails is equal to the probability $\mathbb{P}(n_n \leq n_a)$. Under the random geometric graph model, the node distribution on the network follows a Poisson point process [28]. Thus, both n_n and n_a follow the Poisson distribution with parameters $c\pi\lambda r^2$ and $(1-c)\pi\lambda r^2$, respectively. We can then obtain

$$\mathbb{P}(\text{nearest-to-average fails}) = \mathbb{P}(n_n \leq n_a), \qquad (23)$$

which is called a Poisson race. From a Chernoff bound [29], we have

$$\begin{aligned} \mathbb{P}(n_n \leq n_a) &\leq e^{-\left(\sqrt{c\pi\lambda r^2}-\sqrt{(1-c)\pi\lambda r^2}\right)^2}, \\ &= e^{-\pi\lambda r^2\left(\sqrt{c}-\sqrt{1-c}\right)^2} \\ &= e^{-a\left(\sqrt{c}-\sqrt{1-c}\right)^2} \\ &= e^{-a\left(1-2\sqrt{c(1-c)}\right)}. \end{aligned} \qquad (24)$$

□

Theorem 3 shows that the failure probability of the nearest-to-average rule can be bounded by an exponential function of a and c. We show an example of such an upper bound as a function of c (percentage) in Fig. 10. It is noted from Fig 10 that the failure probability is very small when c is small. For example, when $c = 6\%$ (indicating 6% of the network nodes are colluding attackers), the failure probability is smaller than 0.52%. When $c = 20\%$, the upper bound of the failure probability becomes 13.5%. However, in practice, the value of c should be very small because most of the nodes in the network should be legitimate.

4.3 Discussions

For a centralized backpressure implementation, it is easy to let the controller to decide which nodes are falsifying virtual queue information according to (22). The proposed trust mechanism can also be used in a fully distributed scenario. However, a key issue is then who will collect such information and decide which nodes are falsifying the virtual queue information. A natural way is to let every neighbor communicate with each other then decide individually who is falsifying the information. A malicious neighbor may try to send or forward the falsified information to other neighbors to affect their decision. The problem is essentially a Byzantine generals problem [30]. It is known from [30] that if one third or more of the neighbors are malicious, there exists no solution to remove the falsified information injected by malicious nodes. Comparing this with the majority rule (22) in the centralized design, we can conclude that the trust mechanism is less tolerant of the number of malicious nodes in the distributed backpressure framework than it is in the centralized one.

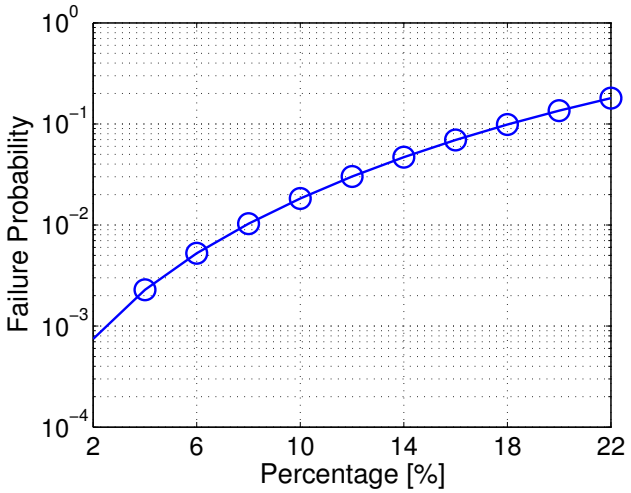

Fig. 10. The failure probability of the nearest-to-average rule as a function of c when $a = 10$.

5 Performance Evaluation

In this section, we conduct an extensive simulation study to evaluate the performance of the designed secure backpressure algorithm with virtual trust queues.

We set up a wireless network with 50 nodes with transmission range 100m uniformly distributed over a 200m-by-200m area. Each node is half-duplex thus cannot transmit and receive at the same time. We adopt the protocol interference model; i.e., if two nodes are within each other's transmission range, their link rate is set to be 100 packets/s; otherwise, the rate is 0. In addition, if a node is receiving from a neighbor at a time slot, none of its other neighbors will be scheduled to transmit.

There are in total 10 end-to-end flows with randomly selected source-destination pairs in the network. Per-flow packet arrival at each node per time slot follows the uniform distribution over time $[0, A_{\max}]$, where $A_{\max} = 5$ packets/s. The simulation starts at time 0 with queue backlogs of all nodes equal to zero. Each benign node uses two virtual trust queues with $\epsilon = \eta = 0.5$ for each of its neighbors along with the data packet queues.

We consider a comprehensive set of attack scenarios in the simulations:

1) *Blackhole attacks*, which always broadcast zero queue backlogs and high channel rates to attract packets to be routed to them, then drop all received packets.
2) *Selective-forwarding attacks*, which keep relatively low profile compared with blackhole attacks. They do not falsify any information and obey the backpressure scheduling, but only drop packets routed to them for particular flows.

3) *On-off attacks*, which act as blackholes or legitimate nodes during on and off periods.
4) *Transmission-opportunity-wasting attacks*, which never falsify information, but simply abandon its scheduled transmission opportunity to degrade the network throughput.
5) *Selfish nodes*, which always attempt to empty its queues by broadcasting high queue backlogs to capture the transmission opportunity.
6) *Heterogeneous attacks*, which include all above attackers at different nodes in the same network.

In our performance evaluation, we define the metric of (normalized) throughput as the average amount of delivered data per time slot normalized by the link rate.

5.1 Blackhole Attacks

We randomly select one node in the network acting as a blackhole. Fig. 11 shows the throughput in the network under the blackhole attack. It is noted from Fig. 11 that initially, the network throughput increases as run time passes. This is because the network is lightly loaded in the initial state. As more packets arrive at each node, the network throughput increases gradually and becomes stable. When there is no attack, the throughput approaches near 1 over time. However, when there is an attacker, we can see over 80% degradation for the throughput in the network due to the blackhole that keeps dropping packets.

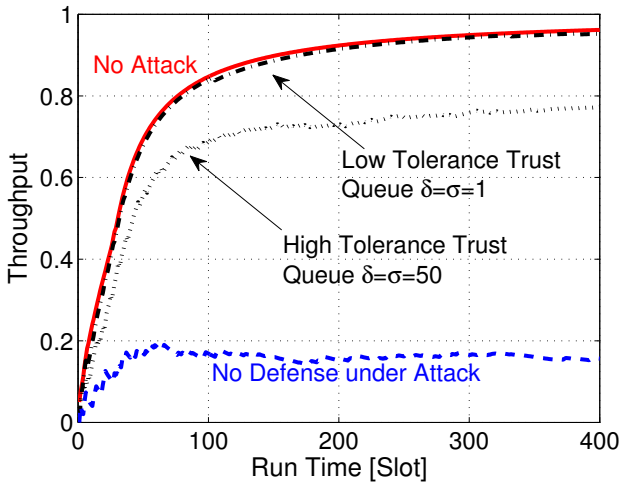

Fig. 11. Throughput over run time for different scenarios.

We deploy the two virtual trust queues with two service rates: (i) $\delta = \sigma = 1$ and (ii) $\delta = \sigma = 50$. As we have discussed, a smaller value of service rate δ or σ mean a smaller tolerance level for attack behavior. If an attacker operates beyond the given tolerance level, which results in an unstable queue, the attacker will be excluded from the routing decision as indicated in Theorem 2. Consequently, it is observed from Fig. 11 that the throughput is substantially improved under the virtual trust queue strategy, in particular for the low tolerance case with $\delta = \sigma = 1$.

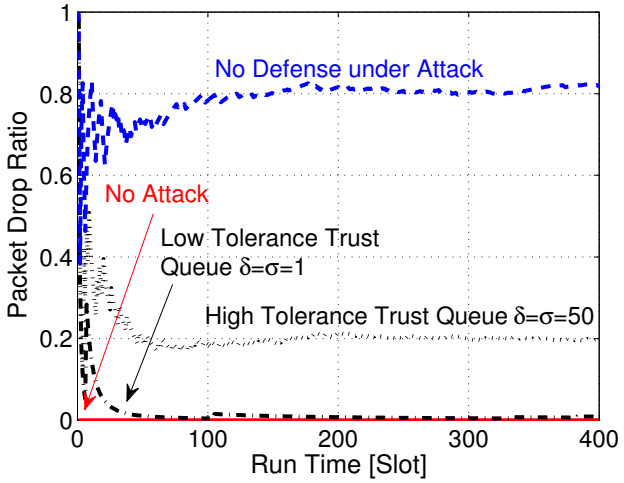

Fig. 12. Packet drop ratio over run time for different scenarios.

Fig. 12 shows the packet drop ratio in network under the same attacks. We observe from Fig. 12 that the packet drop ratio is zero without the attack and becomes around 80% under the attack. However, the trust queue mechanism is able to drag the dropping ratio close to zero. For example, when the low-tolerance virtual trust queue is deployed, the packet drop ratio is reduced to 0.087% at run time slot $t = 400$, which indicates that the neighbors of the two blackholes rarely forward packets to them due to lack of trust.

Figs. 11 and 12 demonstrate that a low tolerance virtual trust queue (i.e., small values of δ and σ) exhibits better performance than a high tolerance queue. As the virtual trust queue mechanism is based on the observations on other nodes, which may have errors in the real world, it is desirable to choose reasonably small values of δ and σ to account for observation errors in practical applications.

We also measure the throughput performance under different numbers of blackholes in Table 1, which shows that when the number of blackholes increases to three, the network exhibits zero throughput. However, when two virtual trust queues are deployed, the throughput is almost not affected by the number of blackholes.

TABLE 1
Throughput at time slot $t = 400$.

Number of Blackholes:	0	1	2	3	4
w/o. virtual queues	0.96	0.17	0.0	0.0	0.0
w. virtual queues	0.96	0.96	0.95	0.95	0.95

5.2 Selective-Forwarding Attacks

Next, we consider selective-forwarding attacks. We randomly select two nodes in the network acting as

selective-forwarding attacks with forwarding ratio $\rho \in [0\%, 100\%]$, which means that an attacker only forwards packets for a percentage ρ of all flows in the network.

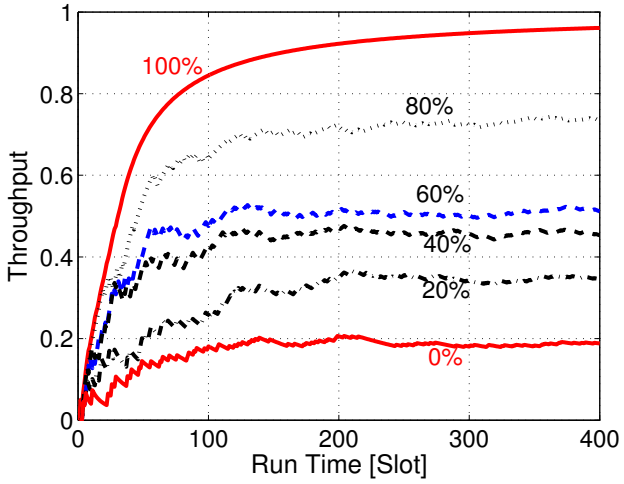

Fig. 13. Throughput under selective-forwarding attacks for forwarding ratio ρ from 0% to 100%.

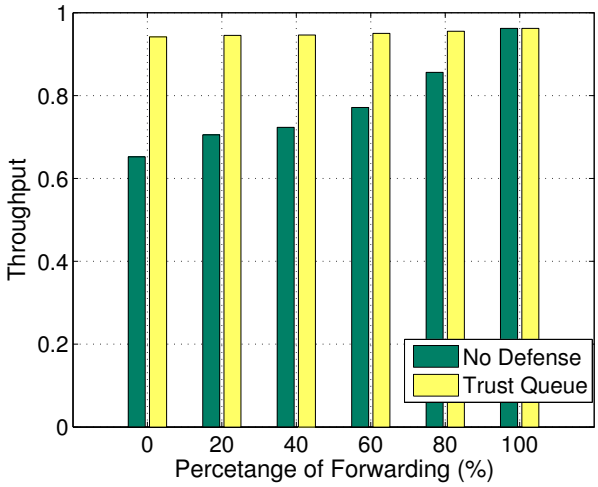

Fig. 14. Throughput under selective-forwarding attacks with and without virtual trust queue.

Fig. 13 depicts the throughput performance under selective-forwarding attacks for forwarding ratio ρ = 0%, 20%, 40%, 60%, 80%, and 100%, where $\rho = 100\%$ means no attack. The performance is measured without any defense in Fig. 13. We can see that the performance is substantially degraded as ρ decreases. For example, as run time goes, we find that the throughput is stabilized at around 0.9622 and 0.1821 for the $\rho = 100\%$ and $\rho = 0\%$ cases, respectively.

Then, we show in Fig. 14 the throughput at time slot $t = 400$ with and without our virtual trust queue mechanism that is set with $\delta = \sigma = 2$. If there is no defense, the throughput increases gradually as the forwarding ratio of attackers ρ increases, and finally reaches the no-attack throughput. If the virtual trust queues are used, we can see that regardless of the value of ρ, the throughput is always close to the non-attack throughput. For example, when $\rho = 0\%$, the virtual trust queue strategy achieves throughput of 0.9417 compared with the non-attack throughput of 0.9622.

5.3 On-Off Attacks

We also consider the on-off attacks, which act as blackholes and legitimate nodes during on and off periods, respectively. We randomly choose one node in the network as an on-off attacker, whose on and off periods are both equal to 50 time slots.

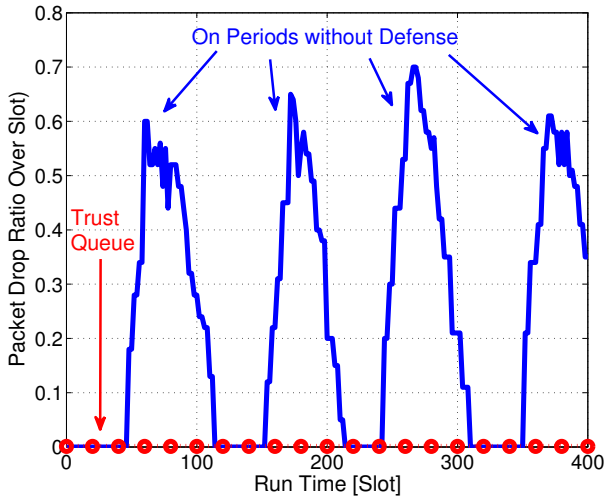

Fig. 15. Packet drop ratio in the network under on-off attacks: virtual trust queue vs. no defense.

To capture how the network reacts in transient states under the on-off attacker, we use packet drop ratio as the performance metric, which is defined as the ratio between the number of dropped packets (by the attacker) and the number of transmitted packets. Fig. 15 illustrates the packet drop ratio in the network under the on-off attack. It is noted from Fig. 15 that if there is no defense in the network, the packet drop ratio increases and decreases sharply, and these trends alternate. This is due to the fact that attackers only attack during the on-periods. In contrast, if the virtual trust queue mechanism with $\delta = \sigma = 1$ is used in the network, we observe in Fig. 15 that the packet drop ratio is always zero, indicating that our mechanisms can immediately find and exclude the attackers when they are on.

5.4 Multiple Heterogonous Attacks

We have shown that the virtual trust queue mechanism is effective to combat (1) blackhole, (2) selective-forwarding, and (3) on-off attacks. We next evaluate its effectiveness against (4) transmission-opportunity-wasting attacks and (5) selfish nodes as part of heterogeneous attacks. We combine the (4) and (5) attacks with

the (1), (2), and (3) attacks to form a more complicated scenario. In the experiments, the forwarding ratio of the selective-forwarding attack is set to be $\rho = 60\%$, and the on and off periods of the on-off attacker are equally set to be 50 time slots.

TABLE 2
Throughput at time slot $t = 400$ under different attacks.

Combination:	4	5	4,5	3,4,5	2,3,4,5	all
w/o. virtual queues	0.23	0.85	0.21	0.15	0.09	0.0
w. virtual queues	0.96	0.96	0.96	0.96	0.95	0.95

We measure the throughput under different attacks shown in Table 2. We can conclude that the virtual trust queue mechanism always achieves near-optimal performance regardless of different combinations of attacks present in the network.

6 Related Work

The backpressure algorithm was originally introduced by Tassiulas and Ephremides in [1]. Since then, significant efforts have been devoted to adapting the backpressure algorithm with realistic network constraints (e.g., [2]–[4], [16], [19], [20], [24]), and developing backpressure based system prototypes for wireless networks (e.g., [5]–[7], [17]) and emerging network systems such as smart grids [31]–[33]. The state-of-the-art greatly enhanced our understanding of the backpressure algorithm, and supported the feasibility of real-system implementation of backpressure-based routing and scheduling solutions in multi-hop wireless networks. However, the backpressure algorithm and its variants remain vulnerable to insider threats with a variety of attacking behaviors. This appears as one major obstacle to practical deployment of the backpressure algorithm.

Another line of work related to this paper has investigated the design of trust-based routing to combat selfish or malicious nodes in the network (e.g., [13], [14], [21]–[23]). However, they are designed for other different routing mechanisms and cannot be readily adapted to the backpressure algorithm due to its specific information exchange and optimization characteristics.

It is worth noting that a recent paper [11] proposed a trust-based backpressure algorithm for wireless sensor networks. But there is no means to track trust over time. In addition, it is still not clear to what extent the approach can limit or control various attack behaviors. The proposed virtual trust queue based mechanism can be regarded as a generic solution to secure any backpressure based routing and scheduling protocols against a variety of attacks with a guarantee of sustaining resilience and throughput.

7 Conclusions

In this paper, we provided a systematic study on securing the backpressure algorithm. We proposed a novel virtual trust queuing mechanism to defend the backpressure algorithm against both information-falsification and protocol-violation attacks. We showed that by jointly stabilizing the virtual trust queue and the real packet queue, the backpressure algorithm achieves guarantees of attack resilience as well as throughput performance. Finally, we conducted extensive simulations to validate the effectiveness of the virtual trust queue mechanism. Our results showed that the virtual trust queue mechanism secures the backpressure algorithm against a variety of attacks. Therefore, our proposed solution clears a major barrier for practical deployment of backpressure algorithm for secure wireless applications.

References

[1] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Automatic Control*, vol. 37, pp. 1936–1948, Dec. 1992.

[2] M. Alresaini, M. Sathiamoorthy, B. Krishnamachari, and M. J. Neely, "Backpressure with adaptive redundancy (BWAR)," in *Proc. of IEEE INFOCOM*, 2012.

[3] A. Warrier, S. Janakiraman, S. Ha, and I. Rhee, "DiffQ: Practical differential backlog congestion control for wireless networks," in *Proc. of IEEE INFOCOM*, 2009.

[4] H. Seferoglu and E. Modiano, "Diff-Max: Separation of routing and scheduling in backpressure-based wireless networks," in *Proc. of IEEE INFOCOM*, 2013.

[5] L. Huang, S. Moeller, M. J. Neely, and B. Krishnamachari, "LIFO-backpressure achieves near optimal utility-delay tradeoff," *ACM/IEEE Trans. Networking*, pp. 831–844, June 2013.

[6] J. Nunez-Martinez, J. Mangues-Bafalluy, and M. Portoles-Comeras, "Studying practical any-to-any backpressure routing in Wi-Fi mesh networks from a Lyapunov optimization perspective," in *Proc. of IEEE MASS*, 2011.

[7] J.-Y. Yoo, C. Sengul, R. Merz, and J. Kim, "Experimental analysis of backpressure scheduling in IEEE 802.11 wireless mesh networks," in *Proc. of IEEE ICC*, 2011.

[8] L. Ying, S. Shakkottai, A. Reddy, and S. Liu, "On combining shortest-path and back-pressure routing over multihop wireless networks," *IEEE/ACM Trans. Networking*, vol. 19, Jun 2011.

[9] S. Liu, L. Ying, and R. Srikant, "Throughput-optimal opportunistic scheduling in the presence of flow-level dynamics," *IEEE/ACM Trans. Networking*, vol. 19, Aug 2011.

[10] Z. Lu, W. Wang, and C. Wang, "From jammer to gambler: Modeling and detection of jamming attacks against time-critical traffic," in *Proc. of IEEE INFOCOM*, 2011.

[11] R. Venkataraman, S. Moeller, B. Krishnamachari, and T. R. Rao, "Trust-based backpressure routing in wireless sensor networks," *Int. J. Sensor Networks*, 2014.

[12] Z. Lu, W. Wang, and C. Wang, "On order gain of backoff misbehaving nodes in CSMA/CA-based wireless networks," in *Proc. of IEEE INFOCOM*, 2010.

[13] A. A. Pirzada, C. McDonald, and A. Datta, "Performance comparison of trust-based reactive routing protocols," *IEEE Trans. Mobile Computing*, vol. 5, pp. 695–710, June 2006.

[14] I. Chen, F. Bao, M. Chang, and J. Cho, "Dynamic trust management for delay tolerant networks and its application to secure routing," *IEEE Trans. Parallel and Distributed Systems*, Apr. 2013.

[15] Y. Liu, P. Ning, H. Dai, and A. Liu, "Randomized differential DSSS: Jamming-resistant wireless broadcast communication," in *Proc. of IEEE INFOCOM*, 2010.

[16] L. Ding, T. Melodia, S. Batalama, J. Matyjas, and M. Medley, "Cross-layer routing and dynamic spectrum allocation in cognitive radio ad hoc networks," *IEEE Trans. Vehicular Technology*, vol. 59, 2010.

[17] R. Laufer, T. Salonidis, H. Lundgren, and P. L. Guyadec, "XPRESS: A cross-layer backpressure architecture for wireless multi-hop networks," in *Proc. of ACM MobiCom*, 2011.

[18] M. J. Neely, "Energy optimal control for time varying wireless networks," *IEEE Trans. Information Theory*, vol. 52, pp. 2915–2934, Jul. 2006.

[19] Y. E. Sagduyu, R. A. Berry, and D. Guo, "Throughput and stability for relay-assisted wireless broadcast with network coding," *IEEE Journal on Selected Areas in Communications*, vol. 31, pp. 1506–1516, Aug. 2012.

[20] Y. Shi, Y. E. Sagduyu, and J. H. Li, "Multi-layer optimization with backpressure and genetic algorithms for multi-hop wireless networks," *Wireless Networks*, vol. 20, pp. 1265–1273, June 2014.

[21] C. Zhang, X. Zhu, Y. Song, and Y. Fang, "A formal study of trust-based routing in wireless ad hoc networks," in *Proc. of IEEE INFOCOM*, 2010.

[22] S. Jain and J. Baras, "Distributed trust based routing in mobile ad-hoc networks," in *Proc. of IEEE MILCOM*, 2013.

[23] F. Bao, I.-R. Chen, M. Chang, and J.-H. Cho, "Hierarchical trust management for wireless sensor networks and its applications to trust-based routing and intrusion detection," *IEEE Trans. Network and Service Management*, vol. 9, pp. 169–183, Mar. 2012.

[24] L. Georgiadis, M. J. Neely, and L. Tassiulas, "Resource allocation and cross-layer control in wireless networks," *Foundations and Trends in Networking*, vol. 1, pp. 1–144, 2006.

[25] A. Benyassine, E. Shlomot, H.-Y. Su, D. Massaloux, C. Lamblin, and J.-P. Petit, "Itu-t recommendation g. 729 annex b: a silence compression scheme for use with g. 729 optimized for v. 70 digital simultaneous voice and data applications," *IEEE Communications Magazine*, vol. 35, no. 9, pp. 64–73, 1997.

[26] S. Ranjan, R. Swaminathan, M. Uysal, A. Nucci, and E. Knightly, "DDoS-Shield: DDoS-resilient scheduling to counter application layer attacks," *IEEE/ACM Trans. Networking*, vol. 17, no. 1, pp. 26–39, Feb 2009.

[27] Y. Xie and S.-Z. Yu, "Monitoring the application-layer ddos attacks for popular websites," *IEEE/ACM Trans. Networking*, vol. 17, no. 1, pp. 15–25, 2009.

[28] M. Penrose, *Random Geometric Graphs*. Oxford Univ. Press, 2003.

[29] G. M. Kamath, E. Şaşoğlu, and D. Tse, "Optimal haplotype assembly from high-throughput mate-pair reads," *arXiv preprint arXiv:1502.01975*, 2015.

[30] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Trans. Programming Languages and Systems*, vol. 4, no. 3, pp. 382–401, 1982.

[31] M. Wei and W. Wang, "Greenbench: A benchmark for observing power grid vlunerability under data-centric threats," in *Proc. of IEEE INFOCOM*, 2014.

[32] B. Hu and H. Gharavi, "Greedy backpressure routing for smart grid sensor networks," in *Proc. of IEEE CCNC*, 2014.

[33] M. Wei and W. Wang, "Toward distributed intelligent: A case study of peer to peer communication in smart grid," in *Proc. of IEEE GlobeCom*, 2013, pp. 2210–2216.