COMMUNICATION ALGORITHMS FOR ADVERSARIAL MULTIPLE ACCESS
CHANNELS

by

Lakshmi Anantharamu

B.E., University of Mysore, India, 1994
M.S., University of Colorado, 2007

A thesis submitted to the
Faculty of the Graduate School of the
University of Colorado in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Computer Science and Information Systems
2013
This thesis for the Doctor of Philosophy degree by
Lakshmi Anantharamu
has been approved for the
Computer Science and Information Systems Program
by

Gita Alaghband, Chair

Bogdan S. Chlebus, Advisor

Ellen Gethner

Ilkyuen Ra

Steven Walczak

April 11, 2013
Anantharamu, Lakshmi (Ph.D., Computer Science and Information Systems)  
Communication Algorithms for Adversarial Multiple Access Channels  
Thesis directed by Associate Professor Bogdan S. Chlebus

ABSTRACT

We investigate dynamic broadcasting in multiple access channels. Packet injection and jamming are constrained by adversarial models, which determine injection rates and burstiness of traffic. We develop a number of deterministic distributed broadcast protocols and study their efficiency. The performance of protocols is measured by packet latency and queue size, as functions of the parameters of the underlying adversarial models. We derive worst-case upper and lower bounds on packet latency and queue size, and show impossibility results, all with respect to classes of broadcast protocols and adversaries. We develop a simulation environment for dynamic broadcasting on multiple access channels. We report results of experiments in which the mutual performance of protocols have been compared. The experiments involve both the classical back-off protocols and the protocols we have proposed.

The form and content of this abstract are approved. I recommend its publication.

Approved: Bogdan S. Chlebus
DEDICATION

This work is dedicated to my family, Sahil, Rahul and Prakash for all their love and support.
ACKNOWLEDGEMENT

I cannot thank my advisor Bogdan enough for giving me an opportunity to work with him. Bogdan has been the worst critic of my work and the greatest inspiration to do better always. I am ever grateful for that. This would not have been possible without his support of me through NSF grant. I also want to thank Darek Kowalski and Mariusz Rokicki for collaboration on several research projects. I want to thank the chair of our department, Gita Alaghband for all the support over the years and Ellen Gethner for being an inspiration to women in theory.

I want to thank my family who supported me and loved me the same through the years. I want to thank my husband Prakash for taking on so much more of our family responsibilities, tolerating my long hours of work and supporting me through out. I could not have done this without him. I also want to thank my kids Rahul and Sahil for being so patient and loving. Last but not least, I want to thank my mother who is the greatest inspiration of my life, for all of her support and help over the years while I worked on this.
CONTENTS

Figures ........................................................................................................ viii

Chapter

1. Introduction .............................................................................................. 1

2. Main results ............................................................................................. 4

3. Related work ........................................................................................... 7

4. Technical preliminaries .......................................................................... 15

5. Channels with jamming ........................................................................... 19

5.1 Preliminaries ........................................................................................ 19

5.1.1 Packet latency of full sensing protocols .................................... 25

5.1.2 Packet latency of adaptive protocols ....................................... 29

5.2 Conclusion ............................................................................................. 37

6. Channels without jamming ..................................................................... 39

6.1 Full sensing protocols .......................................................................... 39

6.1.1 Full sensing protocols with collision detection ........................ 40

6.1.2 Adaptive protocol ......................................................................... 44

6.2 Conclusion ............................................................................................. 44

7. Individual injection rates ........................................................................ 45

7.1 Results .................................................................................................. 46

7.2 Preliminaries ........................................................................................ 47

7.3 Impossibilities and lower bounds ..................................................... 50

7.3.1 Impossibility results for acknowledgment based protocols .......... 50

7.3.2 Lower bounds ............................................................................... 54

7.4 Two protocols of small latency ............................................................ 55

7.4.1 A full sensing protocol with collision detection ....................... 56

7.4.2 An adaptive protocol without collision detection .................... 59

7.5 A full sensing protocol without collision detection ......................... 60
7.6 Conclusion ......................................................... 66
8. Experiments ....................................................... 68
  8.1 Simulations without jamming ................................. 68
     8.1.1 Machinery of packet injection: .......................... 69
  8.2 Simulations with jamming .................................... 73
  8.3 Conclusion ....................................................... 76
9. Adhoc multiple access channels ................................. 78
  9.1 Technical preliminaries ....................................... 82
  9.2 Limitations on broadcasting ................................. 86
  9.3 Activation based protocols ................................... 93
     9.3.1 A non-adaptive protocol ................................. 93
     9.3.2 An adaptive algorithm ................................... 98
  9.4 Full sensing protocols ........................................ 103
     9.4.1 A non-adaptive algorithm ............................... 104
     9.4.2 An adaptive algorithm ................................... 109
  9.5 Conclusion ....................................................... 114
10. Open problems and future work ............................... 115
References ............................................................. 116
FIGURES

Figure

8.1 Comparison of maximum packet latency of deterministic and backoff protocols for varying injection rates ........................................ 68
8.2 Comparison of maximum packet latency of deterministic protocols for varying injection rates ................................................. 69
8.3 Comparison of maximum packet latency of regular and old-first deterministic protocols for varying injection rates .......................... 71
8.4 Comparison of maximum packet latency of deterministic and backoff protocols for varying number of stations ............................ 72
8.5 Comparison of maximum packet latency of deterministic and backoff protocols for varying injection and jamming rates ......................... 75
8.6 Comparison of maximum packet latency of deterministic protocols for varying injection and jamming rates ................................. 76
9.1 Algorithm counting-backoff .................................................. 94
9.2 Algorithm queue-backoff ................................................... 99
9.3 Algorithm tripped-rounds .................................................. 105
9.4 Algorithm paired-rounds .................................................. 111
1. Introduction

We study dynamic broadcasting on multiple access channels. Multiple access channels are a model of medium access control in some implementations of local area networks. Such implementations include wired Ethernet, as represented by the IEEE 802.3 collection of standards, and wireless networks in the restricted case when the available band allows to use just one radio frequency. Abstracting medium access control as multiple access channel allows to study optimality of communication protocols in a clearly defined communication environment, without constraints of the ever evolving technologies and IEEE standards. The goals of this work include demonstrating that deterministic distributed protocols have untapped potential, especially for heavy broadcast loads. We evaluate the performance of protocols in terms of the worst-case queue sizes and packet latency.

The traditional approach to dynamic broadcasting on multiple access channels uses randomization to arbitrate for access to the channel in a distributed manner. Typical examples of randomized protocols include backoff protocols, like the binary exponential backoff employed in the Ethernet. It has been generally considered inevitable to have to resort to randomization to be able to cope with bursty traffic on such channels: the underlying justification is that in real-world applications most stations stay idle for most of the time so that periods of inactivity are interspersed with unexpected bursts of activity by unpredictable configurations of stations. This perception contributed to ingrain the prevalent opinion that deterministic solutions have little potential.

One could try to surmise the reasons why the traditional approach to broadcasting via randomization has been considered as essentially the only viable one. An enormous success of the Ethernet as a real-world implementation of local area networks has contributed significantly to this attitude. The methodological underpinnings of the key performance metrics like stability and latency have been normally studied with stochastic assumptions in mind. This especially regards stability understood as ergodicity of the associated Markov chains. The methodology of simulations appears to have been biased
towards models of data injection defined by simple stochastic constraints. Yet another reason has been the perception of an apparent lack of alternatives in both theoretical modeling and simulations. The success of the Ethernet has had negative consequences on the development and exploration of alternative solutions, as success often breeds inertia so that one does not want to change a solution that appears to work. The exponential backoff embedded in the Ethernet has two properties: it is a randomized acknowledgment based protocol of backoff type and it is an exponential backoff. It appears that the former property is essentially responsible for the success of Ethernet, as Håstad et al. [42] demonstrated that the polynomial (quadratic) backoff outperforms the exponential one.

We are motivated to consider deterministic algorithms for a number of reasons. The basic philosophical question of ‘why not determinism?’ is intriguing. We want to take this work beyond the IEEE standards. We want to study the theoretical underpinnings in a deterministic framework. Considering this as a problem in science, rather than engineering, we want to ask the very fundamental questions of science, like ‘what is and what is not possible?’.

There are various approaches to model multiple access channels in terms of what is known to stations. Historically, the first approach was to use the queue-free model, in which each injected packet is treated as if handled by an independent station without any name and no private memory for a queue. In this model, the number of stations is not set in any way, as stations come and go similarly as packets do; see [35] for the initial work on this model and [20] for recent one. Unpredictability of access to the channel among the stations can be handled by using randomization in resolving conflict for access, as implemented in carrier-sense multiple access protocols, see the book by Keshav [46] for an overview. Considering deterministic protocols and their worst-case performance requires methodological setting specifying worst-case bounds on how much traffic a network would need to handle. This can be accomplished formally through suitable adversarial models of network traffic demands. An alternative approach is to have a system with a fixed number \( n \) of stations, each equipped with private memory to store packets in a queue.
An attractive feature of such fixed-size systems is that even simple randomized protocols like Aloha are stable under suitable traffic load [65] while in the queue-free model the binary exponential backoff is unstable for any arrival rate [4]. We consider models where the size of the system and names of stations are known and also consider anonymous systems where the size of the system and names of stations are unknown.

To study the worst case performance of deterministic algorithms, we need a methodology for arrival of packets that would allow us to do so. The methodology of adversarial queuing allows the same. In a basic model, an adversary constrained by rate and burstiness controls injection of packets. We consider adversarial queuing as a model to study performance of protocols. Packets are injected into stations by an adversary who is restricted by rate constraints. We consider two types of basic adversarial models - window-type adversary and leaky-bucket adversary.

A protocol in communications is the same as algorithm in computer science and we will use the two interchangeably in this work. We consider restricted classes of protocols, specifically: acknowledgment based, full sensing and general adaptive ones. We consider channels with and without collision detection. An acknowledgment based protocol transmits based on the round of injection and the binary array of bits that contains the schedule. An adaptive protocol can have control bits attached to the transmission where as a full sensing ones does not have any control bits attached. We compare communication environments by investigating the relative power of classes of protocols as expressed by attainable packet latency and queue size for given demand on traffic. We develop tools to perform experiments to compare packet latency of deterministic and randomized backoff protocols. We introduce a novel approach in modeling adversarial packet injection and jamming in simulated experiments to represent the burstiness of traffic. As far as we know modeling such an adversary has not been attempted before. Results of experiments are presented to verify and visualize the abstract upper bounds on packet latency.
2. Main results

Here we summarize some of the main results in chapters 5, 6, 7, 8 and 9.

Some of the results in chapter 5 appeared in a preliminary form as conference paper in Anantharamu et al. [12]. In chapter 5 we extend the basic adversarial model to include jamming, where the 'jammed' round has the same effect as a collision. The goal of this work was to study stability and packet latency of deterministic protocols against leaky-bucket adversaries with combined injection and jamming rates less than 1. The main results include a comparison of performance between full sensing and adaptive protocols in terms of the worst case packet latency and knowledge required to achieve bounded packet latency. We show that a bounded worst-case packet latency is achievable by full sensing protocols against adversaries for whom jamming burstiness is at most a given bound $J$, while $J$ is part of the algorithm. On the other hand we demonstrate that adaptive protocols can achieve bounded packet latency without restricting jamming burstiness of adversaries in the algorithm. For all the deterministic protocols we study, we give upper bounds as functions of the size of the system and the adversary and show tightness of bounds.

Some of the results given in chapter 6 appeared in a preliminary form as conference paper in Anantharamu et al. [11]. We consider worst case packet latency of deterministic distributed protocols, depending on size of the system and an adversary of injection rates less than one, but a channel without jamming. We concentrate on specific deterministic protocols; for them we estimate upper bound on the queue size and packet latency they provide as functions of the size of the system and an adversary at hand. One of the interesting and surprising results as opposed to intuitions was the benefit of a design paradigm, which we call 'old-first' in the design of protocols.

Some of the results given in chapter 7 appeared in a preliminary form as conference paper in Anantharamu et al. [13]. We study the worst-case performance of broadcasting when traffic demands are specified as adversarial environments modeled by window adversaries with individual injection rates associated with stations and when protocols are
both distributed and deterministic. We allow the adversaries to be such that the associated aggregate injection rate (the sum of all the individual rates) is 1, which is the maximum that allows for stability. The goal was to explore what quality of service can be achieved for individual injection rates and compare the adversarial environments defined by individual rates versus global ones, under maximum broadcast loads of one packet per round. The underlying motivation for this work was that individual injection rates are more realistic in moderate time spans and hopefully the limitations on quality of service with throughput 1 discovered in [30] would not hold when the rates are individual. Indeed, bounded packet latency turns out to be achievable with individual injection rates when the aggregate rate is 1. This is in contrast with global injection rates for which achieving finite waiting times is impossible for throughput 1, as was shown in [30]. We show upper and lower bounds on queue size and packet latency, which depend on the classes of protocols and whether collision detection is available or not. The most restricted acknowledgment based protocols cannot achieve throughput 1, which strengthens the result for global injection rates [30]. We show upper and lower bounds on queue size and packet latency, which depend on the classes of protocols and whether collision detection is available or not.

In chapter 8 we have experimental setup and results of experiments. Some of this results appeared in preliminary form in conference papers Anantharamu et al. [11, 12] We present results of experiments in which protocols are assessed by performance metrics evaluated depending on injection rate and burstiness of traffic. Patterns of packet injection in the experiments are structured to capture the corresponding adversarial models while using stochastic components in a limited manner. We ran experiments for the deterministic protocols and also with two randomized protocols, which were chosen to be the binary exponential back-off and the quadratic polynomial back-off. The findings indicate that deterministic protocols compare favorably with the randomized counterparts and can effectively handle varying injection and jamming rates.

In chapter 9 we propose an adversarial model of traffic demands for ad hoc multiple access channels which captures a dynamic scenario in which stations join and leave the
system. Some of the results have been submitted in preliminary form in conference paper Anantharamu et al. [10]. As anonymous systems cannot break symmetry in a deterministic manner, we restrict adversaries by allowing them to activate at most one station per round, which provides potential means for deterministic protocols to be able to handle dynamic traffic. We consider deterministic distributed protocols, which we categorize into acknowledgment based, activation based and full sensing, and independently into adaptive and non-adaptive ones, by the property of either using control bits in messages or not, respectively. We give a number of protocols, for channels with and without collision detection, for which we asses injection rates they can handle with bounded packet latency. More specifically, our non-adaptive activation-based protocol can handle injections rates less than $\frac{1}{3}$ on channels with collision detection, the adaptive activation-based protocol can handle injection rate $\frac{1}{2}$ on channels without collision detection, the non-adaptive full-sensing protocol can handle injection rates less than $\frac{2}{3}$ on channels with collision detection, and the adaptive full-sensing protocol can handle injection rate $\frac{3}{4}$ on channels with collision detection. We also show that the latter protocol is optimal, in terms of injection rate $\frac{3}{4}$ that can be handled with bounded packet latency, as we prove that no protocol can provide bounded packet latency when injection rates are greater than $\frac{3}{4}$. 
3. Related work

Here we survey the previous work on dynamic broadcasting.

**Broadcasting subject to stochastic constraints:** The previous work on dynamic broadcasting in multiple-access channels has been mostly carried out under the assumption that packets were injected subject to stochastic constraints. Such systems can be modeled as Markov chains with stability understood as ergodicity. Alternatively, stability may mean that throughput equals the injection rate. Popular early broadcast protocols like Aloha [1] and binary exponential backoff [56] have been extensively studied with stochastic injection rates; Gallager [35] gives an overview of early work in this direction. For recent papers, see the work by Goldberg et al. [38, 39]. Also Håstad et al. [42], and Raghavan and Upfal [59] are recent papers.

Acknowledgment based protocols include the first randomized protocols studied on dynamic channels. Aloha and binary exponential backoff fall into this category. The throughput of multiple access channels, understood as the maximum injection rate with Poisson traffic that can be handled by a randomized protocol and make the system stable (ergodic), has been intensively studied in the literature. It was shown to be as low as 0.568 by Tsybakov and Likhanov [64]. Goldberg et al. [38] gave such bounds for backoff, acknowledgment-based and full-sensing protocols.

Stability of randomized protocols can be considered in the queue-free model, which represents a large set of stations by having an injected packet associated with a new station that dies after the message has been heard on the channel. Acknowledgment based protocols and full-sensing ones have been identified as natural important subclasses of protocols. Backoff and Aloha protocols have been most popular representatives of acknowledgment based randomized protocols. They were shown to be unstable in the queue-free model; in particular Aldous [4] showed that binary exponential backoff was transient and had zero throughput. Full sensing protocols were shown to fare well in this model; some protocols stable for injection rate slightly below 1/2 were developed; see Gallagher [35].
by a window adversary was studied by Bender et al. [20] in the queue-free model: they showed that exponential backoff is stable for $O(1/\log n)$ arrival rates and unstable for arrival rates of $\Omega(\log \log n / \log n)$. The model of a finite number $n$ of stations with queues got to be considered as a viable alternative, as queues appear to have a stabilizing effect. Håstad et al. [42] showed that the binary exponential backoff was unstable if the arrival rates at stations were equal and their sum exceeded $0.567 + (1/(4n - 2))$, and Al-Ammal et al. [3] showed that the binary exponential backoff was stable for injection rates of $O(n^{-\delta})$, for $\delta > 0.75$. Håstad et al. [42] showed that any superlinear polynomial backoff was stable for any arrival rate less than 1.

The truncated binary exponential backoff is used to implement arbitration for access to the channel in the Ethernet [56]. The performance of binary exponential backoff has been investigated extensively in the saturation model in which each stations has at least one packet in its queue at all times. This model is conducive to decoupling backoff from the environment to study it for its own sake with performance represented as the saturation throughput. Recent work in this direction was performed by Bianchi [24], Hui and Devetsikiotis [43], Kong et al. [48], Kwak et al. [50], Sharma et al. [63], and Ziouva and Antonakopoulos [67]. The stability of a protocol in the adversarial queuing model could be understood to mean that the throughput is as large as an injection rate; such stability of randomized backoff for multiple-access channels was studied by Bender, Farach-Colton, He, Kuszmaul and Leiserson [20]. They showed that the exponential backoff is unstable for rates $\rho \geq c \log \log n / \log n$, for a sufficiently large constant $c$.

**Token based protocols:** Token ring is a protocol that was initially used by IBM computers. This protocol uses a token which is a fixed three byte frame. The token travels around the ring and the processor that has the token gets to transmit. This token passing mechanism is shared by ARCNET and token bus.
**Adversarial queuing:** Adversarial generation of packets was proposed as an alternative methodology to capture the notion of stability of protocols without statistical assumptions about injection rates in store-and-forward routing. Borodin et al. [26] proposed this in the context of greedy contention-resolution routing protocols. They considered rate 1 adversaries, various networks and scheduling policies. Their results included showing stability for every directed acyclic network, instability for the case of graphs with directed cycles for FIFO and Longest-In-System scheduling policies, stability for a unidirectional ring with Farthest-To-Go scheduling policy and instability for Nearest-To-Go scheduling policy. The subsequent work by Andrews et al. [14] concentrated on the notion of universal stability, which for a protocol denotes stability in any network, and for a network denotes stability of an arbitrary protocol executed by the network, both properties to hold under injection rates less than 1. They show that the ring is universally stable with deterministic adversaries with rates less than 1. They also show that certain scheduling policies, such as Newest-in-System (NIS), Longest-in-System (LIS) and Farthest-to-Go (FTG) are universally stable. They also show that certain common scheduling policies, such as FIFO and Nearest To Go(NTG)) are not universally stable.

The quality of work-preserving protocols and the impact of topology of networks has been extensively studied; next we give a selection of representative results. The FIFO protocol was shown to be unstable at arbitrarily low injection rates by Bhattacharjee, Goel and Lotker [23]. Every work-preserving contention-resolution protocol turns out to be stable if injection rate is smaller than $1/(D + 1)$, where $\delta$ is an upper bound on the length of any path that a packet needs to traverse, as was shown by Lotker, Patt-Shamir, and Rosén [53]. Aiello, Kushilevitz, Ostrovsky and Rosén [2] initiated the study of stability of adaptive protocols which have packets carry only their destination addresses, rather than complete routing paths. Many question of how structural properties of networks affect stability of contention-resolution protocols was studied by Koukopoulos, Mavronicolas, Nikoletseas and Spirakis [49].

Rosén and Tsirkin [61] considered routing against rate-1 adversaries; they defined
reliability of a protocol to mean that each packet is eventually delivered and showed that reliability is achievable only in networks with no cycles of length at least 3. Álvarez et al. [6] applied adversarial models to capture phenomena related to routing with varying priorities of packets and to study their impact on universal stability. In Álvarez et al. [7] consider universal stability for different types paths for undirected graphs. For each case they consider they give polynomial algorithms and also show instability for NTG-LIS protocols. In Álvarez et al. [5] address the stability problem when the selected queuing policy is FFS. They show the polynomial time decidability of the stability under FFS in the general case in which the adversary can solve ties arbitrarily. Álvarez et al. [8] addressed the stability of protocols in networks with links prone to failures with adversarial modeling of failures.

Andrews and Zhang [15] gave a universal protocol to control traffic when nodes operate as switches that need to reserve the suitable input and output ports to move a packet from the input port to the respective output one. Andrews and Zhang [16] proposed suitable adversarial models for networks in which nodes represent switches connecting inputs with outputs so that routed packets encounter additional congestion constrains at nodes when they compete with other packets for input and output ports.

Adversarial queuing in multiple access channels: Adversarial queuing as a methodology to study the performance of deterministic distributed broadcast protocols in the model with queues for multiple-access channels was introduced by Chlebus et al. [31]. They re-defined the subclasses of acknowledgment-based and full-sensing distributed protocols in the case when protocols are deterministic. They also defined latency to be fair when it was $O(\text{burstiness}/\text{rate})$ and stability to be strong when queues were $O(\text{burstiness})$, when burstiness was understood as the maximum number of packets that can be injected in a round. It was shown in [31] that no protocol could be strongly stable for $\omega(\frac{1}{\log n})$ rates and full sensing protocols achieving fair latency for $O(1/\text{polylog } n)$ rates were given. Paper [31] showed that no acknowledgment based protocol could be stable for rates larger
than $\frac{3}{1+\lg n}$, and hence that there are no universally stable acknowledgment based protocols. Two acknowledgment based protocols were developed in [31]: one of fair latency for rates at most $\frac{1}{cn\lg n}$, for a sufficiently large $c > 0$, and an explicit one of fair latency for rates at most $\frac{1}{27n^2 \ln n}$. In a subsequent work [30], Chlebus et al. investigated the quality of service for the maximum throughput, that is, the maximum rate for which stability is achievable. They considered two kinds of adversaries, window adversaries and leaky-bucket ones. They developed a stable protocol with $O(n^2 + \text{burstiness})$ queues against leaky-bucket adversaries of injection rate 1, which means that throughput 1 is achievable. They also showed limitations of throughput 1: achieving fairness is impossible, queues need to be $\Omega(n^2 + \text{burstiness})$, and protocols need to be adaptive. Anantharamu et al. [11, 12] studied packet latency of broadcasting on adversarial multiple access channels by deterministic distributed protocols when injection rates are less than 1.

**Jamming:** Gilbert et al. [37] proposed to model disruptive interference in multi-channel single-hop networks by a jamming adversary. This was further investigated by Dolev et al. [34] who considered restricted gossiping in which a constant fraction of rumors needs to be disseminated when the adversary can disrupt one frequency per round. Gilbert et al. [36] who studied gossiping in which the adversary can disrupt up to $t$ frequencies per round and eventually all but $t$ nodes learn all but $t$ rumors, and by Dolev et al. [33] who considered synchronization of a multi channel under adversarial jamming. Awerbuch et al. [18] developed a randomized protocol for multiple-access channels competing against an adaptive jamming adversary that achieves a constant throughput for the non-jammed rounds.

**Multi-hop wireless networks:** Andrews and Zhang [17] investigated routing and scheduling in wireless networks where every node can transmit data to at most one neighboring node per time step and where data arrivals and transmission rates are governed by an adversary; they designed scheduling algorithms that ensure network stability for
the case in which the adversary specifies the paths that the data must follow. Bender et al. [20] applied adversarial approach to study throughput of randomized backoff for multiple-access channels in the queue-free model; they understood stability to mean that the throughput is as large as the injection rate. They showed that the exponential backoff is unstable for $\rho \geq c \log \log n / \log n$ rates, for a sufficiently large constant $c$. Packet latency of routing protocols for store-and-forward wired networks has been a topic of extensive research. A stochastic approach [51, 66] has been dominant prior to introduction of adversarial queuing model. Andrews and Zhang [17] studied routing in wireless networks when data arrivals and transmission rates are governed by an adversary.

**Multiple access channels with jamming:** We know of two papers on jamming in multiple-access channels closely related to this work. Awerbuch et al. [18] studied jamming in multiple access channels in an adversarial setting with the goal to estimate saturation throughput of randomized protocols. They considered an adversarial model with a given jamming rate; the model captures burstiness of jamming by giving the minimum window size that determines the jamming rate. More precisely, a $(T, \lambda)$-type adversary can jam at most $w\lambda$ rounds in any window $w \geq T$. A protocol is $c$-competitive if the nodes manage to perform successful message transmissions in at least a $c$-fraction of the time steps not jammed by the adversary; this is understood to be with high probability or on expectation. The paper presents a medium-access-control protocol that is constant competitive with high probability under any $(T, 1 - \epsilon)$-bounded adversary, assuming the protocol is executed for sufficiently many rounds. The number $\epsilon$ is not known by protocols but it needs to be sufficiently large for the given $n$ and $T$, and protocols are assumed to know approximations of $n$ and $T$ with some precision. That paper was concerned with saturation throughput, which means that packets to be transmitted are available in every station at all times while the goal is to maximize the frequency with which packets are heard on the channel. A main difference of paper [18] with our approach is that their adversary is only busy with jamming and is not involved in injecting packets, while our adversaries
control both aspects of traffic. On the other hand, stations cannot distinguish jamming from the channel being busy due to collisions, which is similar to our assumptions about the model. Another paper, by Bayraktaroglu et al. [19], investigated the performance of the IEEE 802.11 CSMA/CA MAC protocol, as specified in [44], under various jammers. This models of jamming are categorized in two independent dimensions: one is oblivious versus aware and the other is stateful versus memoryless. The former categorization represents the extent to which the jammer can react to the medium state and the latter the ability to maintain an inner state that affects future actions. This results in four kinds of the respective jammers. The goal of that paper was to investigate saturation throughput. The theoretical approach was based on the discrete Markov model as proposed by Bianchi [24] but extended to jammer models. The outcomes of the theoretical analysis in [19] were validated by experiments.

**Wireless networks with jamming and/or failures:** Gilbert et al. [37] studied single-hop multi-channel networks with jamming controlled by adversaries. Communication was represented as a game between the adversary who has a budget of $\beta$ unknown to the stations while the stations need to transmit some set $V$ of values. A representative result showed that communication can be delayed for $2\beta + \Theta(\log |V|)$ rounds. On the methodological side, the paper [37] proposed to measure efficiency of malicious disruption in terms of two new metrics: jamming gain, defined as the ratio of rounds delayed to adversarial broadcasts, and disruption-free complexity, defined as the number of rounds required to terminate in executions with no disruptions.

Meier et al. [55] considered multi-channel single-hop networks in scenarios when an adversary can disrupt some $t$ channels out of $m$ in a round, when $m$ is known while $t$ is not. Single-hop multi channel wireless networks are like a number of multiple-access channels superimposed on the same set of stations. The communication goal is to discover ‘communication partners’ in the following sense: two nodes successfully discover each other when two nodes use the same channel, one of them is transmitting while the
other is receiving, and simultaneously no other node is transmitting in this round on this channel, and the channel is not jammed. A protocol was proposed in [55] to achieve the communication goal, which is validated by experiments. Gilbert et al. [36] considered a multi-channel where the adversary can control information flow on a subset of channels. Bhandari and Vaidya [21, 22] considered broadcast protocols in multi-hop networks where nodes are prone to failures.

Simulations of back-off in multiple-access channels: Binary exponential back-off has been investigated in the saturation model, which has each stations have at least one packet in its queue at all times. The model allows to study the utmost performance of back-off in terms of the saturation throughput. For recent work see papers by Bianchi [24], Hui and Devetsikiotis [43], Kong et al. [48], Kwak et al. [50], Sharma et al. [63], and Ziouva and Antonakopoulos [67].

Misbehavior in wireless networks: General misbehavior phenomena related to media-access control in wireless networks were surveyed by Guang et al. [41]. Jamming in wireless and sensor networks was reviewed by Mpitziopoulos et al. [57] and Pelechrinis et al. [58].
4. Technical preliminaries

A multiple-access channel is designed to support efficient broadcast. Such systems have additional specific properties which we discuss in this section. We also define adversarial models, classes of broadcast protocols and measures of quality of service. The letter $n$ denotes the number of stations attached to a channel. Each station has a unique integer name in the range between 1 and $n$.

**Multiple Access Channels:** A channel operates synchronously. An execution of a protocol is structured as a sequence of rounds. It takes one round for a station to transmit a message, so two overlapping transmissions occur in the same round and are restricted to this round.

A message successfully received by a station is said to be **heard** by the station. A broadcast system is a *multiple-access channel* if communication is governed on the logical level by the following two rules: (1) a message transmitted by a station is delivered to all the stations in the same round, and (2) a transmitted message is heard by all stations if and only if its transmission does not overlap in time with any other transmissions. At most one message per round can be heard, which means that the throughput rate is at most 1.

Multiplicity of transmissions determines three cases: When no stations transmit in a round, then such a round is **silent**, and **silence** is what the stations receive then from the channel as feedback. When exactly one station transmits in a round, then the message is heard by all the stations in the same round. Multiple transmission in the same round result in a **conflict for access** to the channel, which is called **collision**. When a collision occurs, then no station can hear any message. The channel is **with collision detection** when the feedback from the channel allows the stations to distinguish between silence and collision, otherwise the channel is **without collision detection**. If no collision detection mechanism is available, then stations perceive collision as silence.

**Adversaries:** An adversary is defined by a set of allowable patterns of injections of packets into stations. An adversary generates a number of packets in each round and
next, for each packet, assigns a station to inject the packet in this round. We define the
burstiness of an adversary to be the number of packets that can be injected into the system
in the same round. An adversary is defined by constraints on the patterns of injection
expressed in terms of injection rates at stations and burstiness.

Adversaries can be categorized depending on how injection rates are defined. There
are two popular classes of (1) window adversaries and (2) leaky-bucket ones. Packets are
injected into stations by a window-type adversary that is restricted by an injection rate \( \rho \)
and a window \( w \). The window \( w \) is used to define the injection rate \( \rho \) as follows: for any
contiguous time interval \( \tau \) of \( w \) rounds the number of packets injected in \( \tau \) is at most \( \rho w \).
Packets are injected into stations by a leaky-bucket-type adversary that is restricted by an
injection rate \( \rho \) and a burstiness \( b \). For a time interval \( \tau \), the adversary may insert at most
\( |\tau|\rho + b \) packets.

**Broadcast Protocols:** Packets need to be parked at stations when there is contention for
access to the channel or when multiple packets are injected at the station simultaneously.
Each station has its packets stored in a local queue. If a protocol is stable for a multiple
access channel with the FIFO queuing discipline, then it also will be stable with any
other queuing policy. We use protocols with FIFO queuing, as this discipline additionally
provides fairness and minimizes latency. The number of packets stored at a queue in a
given round is called the *size of the queue* in the round. The capacity of every queue is
assumed to be unbounded, so that a queue can store any number of packets in principle.

When stations execute a protocol, then the *state of a station* is determined by the
values of its private variables. The number \( n \) of all stations is known, in the sense that is
may be a part of code. Protocols we consider are not tailored to any specific parameters
of the adversary; in this sense the adversaries are not known. The contents of packets do
not affect execution of a broadcasting protocol, as packets are treated as abstract tokens.
A *message*, either received from the channel and sent to it, may include a packet and
may include control bits. For instance, a protocol may attach an ‘over’ bit to a transmitted
packet to indicate that the station will not transmit in the next round. A message consisting entirely of control bits is legitimate.

A protocol is formally defined by what constitutes the states and what are the rules governing transitions between states. An event in a round consists of the following actions at each station in the order given: (1) the station either performs a transmission or pauses, as determined by its state, (2) the station receives a feedback from the channel, in the form of either hearing a message with contents or collision or silence, (3) new packets are injected into the station, (4) a state transition occurs.

A state transition is determined by the state at the end of the previous round, the packets injected in the round, and the feedback from the channel in the round. State transitions involve the following operations. Injected packets are immediately enqueued by a station. A transmitted packet is discarded and a new packet to transmit is obtained by dequeuing the queue. If a message is to be transmitted in the next round, then it is prepared as a part of state transition. An execution of a protocol is a sequence of events occurring in consecutive rounds; we consider infinite executions.

Natural subclasses of deterministic protocols for adversarial settings in multiple access channel were defined in [31, 30], we use the same classification. We call a protocol full sensing when additional control bits are never used in messages. A general protocol that is not full sensing is called adaptive; such protocols may send control bits in messages. A protocol is acknowledgment based if the decision whether a currently handled packet is to be transmitted or not depends only on which consecutive round it is devoted to broadcasting the packet, counting rounds from the first one assigned for the packet. When the system runs an acknowledgment based protocol, then a station may ignore feedback from the channel, except for detecting whether the packet transmitted was heard on the channel, which serves as an ‘acknowledgment’ from the channel. Formally, an acknowledgment-based protocol is determined by unbounded binary sequences assigned to stations. Each such a sequence is called a transmission sequence; different stations running the same protocol may have different transmission sequences. These sequences are interpreted as
follows: if the $i$th bit of the transmission sequence of a station equals 1, then the station transmits the currently processed packet in the $i$th round, counting rounds from the first one when the packet was started to be processed, while a 0 as the $i$th bit makes the station pause in the corresponding $i$th round.

A protocol is conflict free if in any execution in any round at most one station transmits. A protocol that is not conflict free is called conflict prone. When a protocol is conflict free then we assume that a channel is without collision detection.

The quality of service specifies desirable properties of the functionality of the system. Stability occurs when the total number of packets in queues at the stations is bounded in all rounds. Packet latency means the maximum number of rounds spent by a packet in a queue waiting to be heard on the channel. Observe that a protocol of bounded latency is stable, as latency is an upper bound on queue size.
5. Channels with jamming

We consider multiple access channels with jamming. A “jammed” round has the same effect as a collision, in how it is perceived by the stations attached to the channel. Stations cannot distinguish a jammed round from a round with collision. This property of jamming allows to capture a situation in which jamming occurs because groups of stations execute their independent communication protocols so that for each group an interference caused by “foreign” transmissions is logically equivalent to jamming. A similar motivation comes from a scenario in which a degradation-of-service attack produces dummy packets that interfere with legitimate packets transmitted by the executed protocol.

We consider medium access control on multiple access channels against adversaries that control both injections of packets into stations and jamming of the communication medium. The studied protocols are deterministic and executed with no centralized control. The goal is to investigate queue size and packet latency of such protocols. We use the slotted model of the channel, in which an execution of a protocol is partitioned into rounds, so that a transmission of a packet takes one round.

5.1 Preliminaries

As defined before there are \( n \) stations attached to the channel. Each station has a unique integer name in the interval \([1, n]\). A transmitting station receives an instantaneous feedback through which it can hear its own transmission when it is successful. Stations’ perception of the channel sensed as busy depends on whether a mechanism of collision detection is available or not. By collision detection we mean that the feedback from a busy channel due to multiple simultaneous transmissions is different from one received when no station attempts to transmit. We study channels without collision detection.

**Jamming:** Jamming considered in this paper occurs in a relatively mild form: it is perceived by stations as artificial collisions. Stations cannot distinguish between a collision caused by multiple simultaneous transmissions and one caused by the adversary to jam the channel for the round, in the sense that the channel is sensed as busy in both cases.
Categorizing rounds: We use the following terminology regarding feedback from the channel obtained by stations in rounds. Rounds are categorized into jammed or clear, depending on whether the adversary jams them or not. A round when no message is heard on the channel is called void. A round when no station transmits is called silent. In particular, each jammed round is void and each collision results in a void round. Stations have no means to sense whether a void round is jammed, or clear but silent, or clear due to a collision.

Protocols: We adapt deterministic distributed protocols, as introduced in [30, 31]. They are understood in exactly the same way as for the adversarial model without jamming, as jamming is not recognizable as a special form of feedback from the channel. We consider two classes of protocols: full sensing protocols and adaptive ones, these definitions were introduced for deterministic protocols in [30, 31]. A message transmitted on the channel consists of a packet and control bits. Packets are provided by the transport layer represented by the adversary. We treat packets as abstract tokens, in the sense that their contents do not affect how protocols handle transmissions. On the other hand, control bits may be added by stations, in a way specified by the code of an executed protocol, to facilitate distributed control of the channel. A protocol is called full sensing when control bits are not sent in messages, which is in contrast to general protocols referred to as adaptive.

Jamming adversaries: We extend the adversarial model to incorporate jamming. We recall the definition of an adversary without jamming, as used in [11, 31, 30]. A leaky-bucket adversary of type \((\rho, b)\) may inject at most \(\rho|\tau| + b\) packets in any contiguous segment \(\tau\) of \(|\tau|\) rounds. For such an adversary, the parameter \(\rho\) is called the injection rate.

In this paper we consider an adversary that controls both packet injections and jamming. An adversary is subject to two independent rates for injection and jamming: A leaky-bucket jamming adversary of type \((\rho, \lambda, b)\) can inject at most \(\rho|\tau| + b\) packets and,
independently, it can jam at most $\lambda|\tau| + b$ rounds in any contiguous segment $\tau$ of $|\tau|$ rounds. For this adversary, we refer to $\rho$ as the injection rate, and to $\lambda$ as the jamming rate.

If $\lambda = 1$ then any round could be jammed. Therefore we assume that the jamming rate $\lambda$ satisfies $\lambda < 1$. Stability is not achievable by a jamming adversary with injection rate $\rho$ and the jamming rate $\lambda$ satisfying $\rho + \lambda > 1$: this is equivalent to $\rho > 1 - \lambda$, so when the adversary is jamming with the maximum capacity, then the bandwidth remaining for transmissions is $1 - \lambda$, while the injection rate is more than $1 - \lambda$. It is possible to achieve stability in the case $\rho + \lambda = 1$, similarly as it was shown for $\rho = 1$ in [30], but packet latency is then unbounded. We assume throughout that $\rho + \lambda < 1$.

The number of packets that the adversary can inject in one round is called its injection burstiness. This parameter equals $\lfloor \rho + b \rfloor$ for a leaky bucket adversary. The maximum continuous number of rounds that an adversary can jam is called its jamming burstiness. A leaky-bucket adversary of type $(\rho, \lambda, b)$ can jam at most $\lfloor b/(1 - \lambda) \rfloor$ consecutive rounds, as the inequality $\lambda x + b \geq x$ needs to hold for any such a number $x$ of rounds.

**Performance:** The basic quality for a protocol in a given adversarial environment is stability, understood to mean that the number of packets in the queues at stations stays bounded at all times. This upper bound on the number of packets waiting in queues is a natural performance metric, see [31, 30]. A sharper performance metric is that of packet latency: it means an upper bound on time spent by a packet waiting in a queue, counting from the round of injection through the round when it is heard on the channel.

**Knowledge:** A property of a system or of environment is said to be known when it can be referred to explicitly in code. We assume that the number of stations $n$ is known to the stations. Each individual name is known to the owner of the name. Properties of an adversary are normally not known, unless stated otherwise. The only exception to this rule in this paper occurs for full-sensing protocols that have burstiness $J$ of an adversary as part of code: the protocols attain the claimed packet latency when burstiness is at most $J$. 

21
Three deterministic distributed protocols: We will consider three specific deterministic distributed protocols already known in the literature about channels without jamming. These protocols can be described as follows.

Protocol ROUND-ROBIN-WITHHOLDING (RRW) is a full-sensing protocol for channels without collision detection. It operates in a round-robin fashion: stations gain access to the channel in the cyclic order of their names. Once a station gets access to the channel by transmitting successfully, it unloads all the packets. A silent round is a signal to the next station, in the cyclic order of names, to take over.

Protocol SEARCH-ROUND-ROBIN (SRR) is a full-sensing protocol for channels with collision detection. It proceeds with a systematic sweep across all the stations looking for these with packets, searching in the cyclic order. When a station with a packet is identified, the station unloads all its packets one by one. A silent round triggers the sweep to be resumed. We apply binary search to identify the next station. The binary search is implemented using collision detection. When we inquire about a segment of stations, then all the stations with packets that are in the segment transmit in the round. A search is completed by a packet heard. A silence indicates that the segment is empty. A collision indicates that multiple stations are in the segment: this results in having the segment partitioned into two halves, with one segment processed next immediately while the other one is pushed on a stack to wait. A transition to the next segment occurs when the stack is empty.

Protocol MOVE-BIG-TO-FRONT (MBTF) is an adaptive protocol for channels without collision detection. Each station maintains a list of all the stations in its private memory. A list is initialized to be sorted in the increasing order of names of stations. The lists are manipulated in the same way by all the stations so are all identical. The protocol schedules exactly one station to transmit in a round, so collisions never occur. This is implemented by having a conceptual token assigned to stations, which is initially assigned to the first station on the list. A station $p$ with the token broadcasts a packet, if it has any, otherwise the round is silent. A station considers itself big in a round when it has at least
n packets; such a station attaches a control bit to all the packets it transmits to indicate this status. A big station is moved to the front of the list and it keeps the token for the next round. When a station that is not big transmits, or when it pauses due to a lack of packets while holding the token, the token is passed to the next station in the list ordered in a cyclic fashion.

Protocols RRW and SRR were introduced in [31] and showed to be universal. Protocol MBTF was introduced in [30] and showed to be stable for injection rate 1.

**The old-first paradigm:** We introduce new protocols by modifying RRW and SRR so that packets are categorized into “old” and “new”. “New” packets become eligible for transmissions only after “old” ones have been transmitted. Formally, an execution is structured as a sequence of conceptual *phases*, which are contiguous segments of rounds of dynamic length. The idea is to have packets that have been injected in a given phase transmitted in the next phase. We maintain two categories of packets: the old ones have been injected in the previous phase and the new ones are being injected in the current phase. When a new phase begins, the old packets have been heard on the channel and the new ones immediately graduate to old.

Protocol OLD-FIRST-ROUND-ROBIN-WITHHOLDING (OF-RRW) operates similarly as RRW, except that when a station gets access to the channel by transmitting successfully, then the station unloads all the old packets while the new packets stay in the queue. A phase is determined as a full cycle around all the stations, so no additional communication is needed to mark a transition to a new phase.

Protocol OLD-FIRST-SEARCH-ROUND-ROBIN (OF-SRR) operates similarly as SRR, except that the search is for old packets only. A new phase begins after all old packets have been processed.

**Protocols for channels with jamming:** We introduce a protocol designed for channels with jamming, JAMMING-ROUND-ROBIN-WITHHOLDING(\(J\)), abbreviated as JRRW(\(J\)).
The design of the protocol is similar to that of RRW, the difference is in how the token is transferred from a station to the next one, in the cyclic order among stations. Just one void round should not trigger a transfer of the token, as it is the case in RRW, because not hearing a message may be caused by jamming.

The protocol has a parameter $J$ interpreted as an upper bound on jamming burstiness of the adversary, which is used to facilitate transfer of control from a station to the next one by way of forwarding the token. The token is moved after hearing silence for precisely $J + 1$ contiguous rounds, counting from either hearing a packet or moving the token; the former indicates that the transmitting station exhausted its queue while the latter indicates that the queue is empty. More precisely, every station maintains a private counter of void rounds. The counters show the same value across the system, as they are updated in exactly the same way determined only by the feedback from the channel. A void round results in incrementing the counter by 1. The token is moved to the next station when the counter reaches $J + 1$. When a packet is heard or the token is moved then the counter is zeroed.

Protocol OLD-FIRST-JAMMING-ROUND-ROBIN-WITHHOLDING($J$), abbreviated OF-JRRW($J$), is obtained from JRRW($J$) similarly as OF-RRW is obtained from RRW. An execution is structured into phases, and packets are categorized into old and new, with the same rule to graduate packets from new to old. When a token visits a station, then only the old packets could be transmitted while the new ones will obtain this status during the next visit by the token.

We say that a protocol designed for a channel without jamming is a token protocol if it uses a virtual token to avoid collisions. Such a token is always held by some station and only the station that holds the token can transmit. Protocols RRW, OF-RRW, JRRW, OF-JRRW, and MBTF are token ones. We can take any token protocol for channels without jamming and adapt it to the model with jamming in the following manner. As usual, the token indicates the right to transmit. If there is a packet to be transmitted by a station in the original protocol, then the modified protocol has a packet transmitted as well, otherwise just a control bit is transmitted. A round in which only a control bit is
transmitted by a modified token protocol is called control round otherwise it is a packet round. The effect of sending control bits in control rounds is that if no rounds are jammed then a message is heard in each round. This approach creates virtual collisions in jammed rounds, so when a void round occurs then this round is jammed as otherwise a message would have been heard. Once a protocol can identify jammed rounds, we may ignore their impact on the flow of control. The resulting protocol is adaptive. We will consider such modified version of the full-sensing protocols RRW and OF-RRW, denoting them by C-RRW and OFC-RRW, respectively. Note that protocol MBTF works by having a station with the token send a message even if the station does not have a packet, so enforcing additional control rounds is not needed for this protocol to convert it into one creating virtual collisions.

Protocols with executions structured into phases are referred to as phase protocols. These include RRW, OF-RRW, JRRW, OF-JRRW, C-RRW, and OFC-RRW. When the old-first paradigm is used in such a protocol then it is called the old-first version of the protocol, otherwise it is the regular version of the protocol. In particular, RRW, JRRW and C-RRW are regular phase protocols, while OF-RRW, OF-JRRW and OFC-RRW are old-first phase protocols.

5.1.1 Packet latency of full sensing protocols

We show that a bounded worst-case packet latency is achievable by full sensing protocols against adversaries for whom jamming burstiness is at most a given bound $J$, while $J$ is part of code. On the other hand, we will demonstrate later that adaptive protocols can achieve bounded packet latency without restricting jamming burstiness of adversaries in code. The full sensing protocols we consider are OF-JRRW($J$) and JRRW($J$). Protocols OF-JRRW($J$) and JRRW($J$) include the parameter $J$ as a part of code, but the value of $J$ does not occur in the upper bounds on packet latency in Theorems 1 and 2.

Lemma 1 Consider an execution of protocol OF-JRRW($J$) against a leaky-bucket adversary of jamming rate $\lambda$, burstiness $b$, and jamming burstiness at most $J$. If there are $x$
old packets in the queues in a round round, then at least \( x \) packets are transmitted within the next \( (x + n(J + 1) + b)/(1 - \lambda) \) rounds.

**Proof:** It takes \( n \) intervals of \( J + 1 \) void rounds each for the token to make a full cycle and so to visit every station with old packets. It is advantageous for the adversary to not jam the channel during these rounds. Therefore at most \( n(J + 1) + x \) clear rounds are needed to hear the \( x \) packets. Consider a contiguous time segment of \( z \) rounds in which some \( x \) packets are heard. At most \( z\lambda + b \) of these \( z \) rounds can be jammed. Therefore the inequality \( z \leq n(J + 1) + x + z\lambda + b \) holds. Solving for \( z \) we obtain

\[
z \leq \frac{x + n(J + 1) + b}{1 - \lambda}
\]

as the bound on length of a contiguous time interval in which at least \( x \) packets are heard.

\( \square \)

**Theorem 1** The packet latency of protocol OF-JRRW(\( J \)) is \( O\left( \frac{bn}{(1 - \lambda)(1 - \rho - \lambda)} \right) \), when executed against a jamming adversary of type \((\rho, \lambda, b)\) such that its jamming burstiness is at most \( J \).

**Proof:** Let \( t_i \) be the duration of phase \( i \) and \( q_i \) be the number of old packets in the beginning of phase \( i \), for \( i \geq 1 \). The following two estimates lead to a recurrence for the numbers \( t_i \). One is

\[
q_{i+1} \leq \rho t_i + b ,
\]

which follows from the definitions of old packets and of type \((\rho, \lambda, b)\) of the adversary. The other estimate is

\[
t_{i+1} \leq \frac{n(J + 1) + q_{i+1} + b}{1 - \lambda} ,
\]

which follows from Lemma 1. Denote \( n(J + 1) = a \) and substitute (5.1) into (5.2) to obtain

\[
t_{i+1} \leq \frac{a + q_{i+1} + b}{1 - \lambda} \leq \frac{a}{1 - \lambda} + \frac{b}{1 - \lambda} + \frac{\rho t_i + b}{1 - \lambda} = \frac{a}{1 - \lambda} + \frac{2b}{1 - \lambda} + \frac{\rho}{1 - \lambda} t_i \leq c + dt_i ,
\]
for \( c = \frac{a+2b}{1-\lambda} \) and \( d = \frac{\rho}{1-\lambda} \). Note that \( d < 1 \) as \( \rho < 1 - \lambda \).

We find an upper bound on the duration of a phase by iterating the recurrence \( t_{i+1} \leq c + dt_i \). To this end, it is sufficient to inspect the sequence of consecutive bounds \( t_1 \leq c, \ t_2 \leq c + dc, \ t_3 \leq c + dc + d^2c, \ldots \) on the lengths of the initial phases to discover a general pattern

\[
t_{i+1} \leq c + dc + d^2c + \ldots d^ic \leq \frac{c}{1-d}.
\]

(5.3)

After substituting \( c = \frac{a+2b}{1-\lambda} \) and \( d = \frac{\rho}{1-\lambda} \) into (5.3), one obtains

\[
t_i \leq \frac{a + 2b}{1-\lambda} \cdot \frac{1}{1-\frac{\rho}{1-\lambda}} \leq \frac{a + 2b}{1-\lambda} \cdot \frac{1}{1-\rho - \lambda} \leq \frac{a + 2b}{1-\rho - \lambda}.
\]

(5.4)

Now replace \( a \) by \( n(J+1) \) in (5.4) to expand it into

\[
t_i \leq \frac{n(J+1) + 2b}{1-\rho - \lambda}.
\]

(5.5)

Apply the estimate \( J \leq \frac{b}{(1-\lambda)} \) to (5.5) to obtain

\[
t_i \leq \frac{n\left(\frac{b}{1-\lambda} + 1\right) + 2b}{1-\rho - \lambda} \leq \frac{2(bn + (n+b)(1-\lambda))}{(1-\lambda)(1-\rho - \lambda)},
\]

(5.6)

which is a bound on the duration of a phase that depends only on the type of the adversary, without involving \( J \). The bound on packet latency we seek is twice that in (5.6), as a packet stays queued for at most two consecutive phases. □

**Theorem 2** The packet latency of protocol JRRW(J) is \( O\left(\frac{b n}{(1-\lambda)(1-\rho - \lambda)}\right) \) when executed against a jamming adversary of type \((\rho, \lambda, b)\) such that its jamming burstiness is at most \( J \).

**Proof:** We compare packet latency of protocol JRRW(J) to that of protocol OF-JRRW(J). To this end, consider an execution of protocols JRRW(J) and OF-JRRW(J) determined by some injection and jamming pattern of the adversary. Let \( s_i \) and \( t_i \) be bounds on the length of phase \( i \) of protocols OF-JRRW(J) and JRRW(J), respectively, when run against the considered adversarial pattern of injections and jamming.

Phase \( i \) of OF-JRRW(J) takes \( s_i \) rounds. When protocol JRRW(J) is executed, the total length \( t_i \) of phase \( i \) is at most

\[
s_i + s_i(\rho + \lambda) + s_i(\rho + \lambda)^2 + \ldots = \frac{s_i}{1-(\rho + \lambda)}
\]

(5.7)
rounds. We obtain that the phase’s length of protocol JRRW(J) differs from that of protocol OF-JRRW(J) by at most a factor of \( \frac{1}{1-\rho-\lambda} \).

Protocols JRRW(J) and OF-JRRW(J) share the property that a packet is transmitted in at most two consecutive phases, the first one determined by the injection of the packet. The bound on packet latency given in Theorem 1 is for twice the length of a phase of protocol OF-JRRW(J). Similarly, a bound on twice the length of a phase of protocol JRRW(J) is a bound on packet latency. It follows that a bound on packet latency of protocol J-RRW(J) can be obtained by multiplying the bound given in Theorem 1 by \( \frac{1}{1-\rho-\lambda} \).  

Tightness of the upper bounds: The upper bounds on packet latency given in Theorems 2 and 1 differ by the multiplicity of the factor \( 1 - \rho - \lambda \) occurring in the denominator. This difference between the two bounds reflects the benefit of the paradigm “old-first” applied in the design of protocol OF-JRRW(J), as compared to protocol JRRW(J).

The multiplicities of the factors \( 1 - \lambda \) and \( 1 - \rho - \lambda \) occurring in the bounds of Theorems 1 and 2 are tight. To show this for the bound of Theorem 2, consider protocol JRRW(J) in a channel with jamming. The adversary will inject packets and jam the channel at full power, subject to constrains imposed by its type. The number of clear void rounds in each phase is \( a = n(J + 1) \). The number of packets injected due to these void rounds is \( a\rho \), and it takes \( a(\rho + \lambda) \) rounds to transmit them. In the first phase, \( a\rho \) packets are transmitted in \( a(\rho + \lambda) \) rounds and a queue of \( a(\rho + \lambda)\rho \) packets builds up in station \( n - 1 \). In the second phase, there are \( a\rho \) and \( n(\rho + \lambda)\rho \) packets transmitted by stations \( n \) and \( n - 1 \), respectively; this takes \( a(\rho + \lambda)^2 \) and results in \( a(\rho + \lambda)^2\rho \) packets queued at station \( n - 2 \). This process continues until phase \( n - 2 \) in which the adversary’s behavior changes. The difference is that the adversary injects just one packet into station 2 after the token has passed through that station. What follows is phase \( n - 1 \) in which the queue at station 1 is \( \Omega(\frac{a+b}{1-\rho-\lambda}) \). Let the adversary inject packets at full power only into station 1 in this phase. Consider the latency of packet in station 2. The packets in sta-
tion 1 are unloaded by withholding the channel while the adversary keeps injecting only into the transmitting station. Let us assign a suitably large $J$ to the adversary, for instance $J \geq b/(2(1 - \lambda))$ will do. The latency of the only packet in station 2 is estimated as being

$$\Omega\left(\frac{n(J + 1) + b}{(1 - \rho - \lambda)^2}\right) = \Omega\left(\frac{\frac{b}{2(1 - \lambda)} + 1 + b}{(1 - \rho - \lambda)^2}\right) = \Omega\left(\frac{bn}{(1 - \lambda)(1 - \rho - \lambda)^2}\right).$$

This shows that the upper bound given in Theorem 2 is tight in the sense of the form of the denominator. Regarding the bound of Theorem 1, one possible way to argue about its tightness is to examine its proof to see that the derivation can be mimicked by adversary’s actions. Another approach to argue that the bound in Theorem 1 is tight, in terms of the factors $\frac{1}{1 - \lambda}$ and $\frac{1}{1 - \rho - \lambda}$, is to observe that the bound in Theorem 2 was obtained from the bound in Theorem 1 by multiplying it by the factor $\frac{1}{1 - \rho - \lambda}$, so that any improvement in the bound in Theorem 1 would be reflected in an improvement in the bound in Theorem 2, which has been shown to be tight.

**Knowledge of jamming burstiness:** We have shown that a full sensing protocol achieves bounded packet latency for $\rho + \lambda < 1$ when an upper bound on jamming burstiness is a part of code. Next, we hypothesize that this is unavoidable and reflects the utmost power of full sensing protocols.

**Conjecture 1** No full sensing protocol can be stable against all jamming adversaries with rates satisfying $\rho + \lambda < 1$.

**5.1.2 Packet latency of adaptive protocols**

We give upper bounds on packet latency for each among the following adaptive protocols C-RRW, OFC-RRW, and MBTF. The bounds are similar to those obtained in Theorems 1 and 2, for their full sensing counterparts. The apparent relative strength of adaptive protocols is reflected in their bounds shedding the factor $1 - \lambda$ in the denominators. Each of these adaptive protocols is stable for any jamming burstiness, unlike the
full sensing protocols we consider that have jamming burstiness they can handle in their codes.

**Lemma 2** Consider an execution of protocol OFC-RRW against the leaky-bucket adversary of type \((\rho, \lambda, b)\). If in any round there are some \(x\) old packets in the system, then at least \(x\) packets are transmitted within the next \(\frac{x+n+b}{1-\lambda}\) rounds.

**Proof:** It takes \(n\) control rounds for the token to pass through all \(n\) stations. Consider a contiguous time segment of \(y\) rounds in which some \(x\) packets are heard. At most \(y\lambda + b\) of these \(y\) rounds can be jammed. It follows that the inequality \(y \leq n + x + y\lambda + b\) holds. Solving for \(y\) yields the upper bound \(y \leq (x+n+b)/(1 - \lambda)\), as the bound on length of a contiguous time interval in which at least \(x\) packets are heard. \(\square\)

**Theorem 3** The packet latency of protocol OFC-RRW is \(O\left(\frac{n+b}{1-\rho-\lambda}\right)\) when executed against the jamming adversary of type \((\rho, \lambda, b)\).

**Proof:** Let \(t_i\) denote the duration of phase \(i\) and \(q_i\) be the number of old packets in the beginning of phase \(i\), for \(i \geq 1\). We use the following two estimates to derive a recurrence for the numbers \(t_i\). One is

\[
q_{i+1} \leq \rho t_i + b , \tag{5.8}
\]

which follows from the definition of old packets and the adversary of type \((\rho, \lambda, b)\). The other is

\[
t_{i+1} \leq \frac{q_{i+1} + n + b}{1 - \lambda} , \tag{5.9}
\]

which follows from Lemma 2. Using the abbreviations \(c = \frac{n+2b}{1-\lambda}\) and \(d = \frac{\rho}{1-\lambda}\), substitute (5.8) into (5.9) to obtain

\[
t_{i+1} \leq \frac{n+b}{1-\lambda} + \rho t_i + b = \frac{n+2b}{1-\lambda} + \frac{\rho}{1-\lambda} t_i \leq c + dt_i .
\]

To find an upper bound on the duration of a phase, iterate the recurrence \(t_{i+1} \leq c + dt_i\), which produces

\[
t_{i+1} \leq c + dc + d^2c + \ldots d^i c \leq \frac{c}{1-d} . \tag{5.10}
\]
After substituting \( c = \frac{n+2b}{1-\lambda} \) and \( d = \frac{\rho}{1-\lambda} \) into (5.3), we obtain, by algebraic manipulations, the estimate

\[
t_i \leq \frac{n+2b}{1-\lambda} \cdot \frac{1}{1-\frac{\rho}{1-\lambda}} \leq \frac{n+2b}{1-\lambda} \cdot \frac{1-\lambda}{1-\rho-\lambda} \leq \frac{2(n+b)}{1-\rho-\lambda}.
\]

The bound on packet latency we seek is twice that in (5.11), as a packet is queued for at most two consecutive phases. \( \square \)

**Theorem 4** The packet latency of protocol C-RRW is \( O\left(\frac{n+b}{(1-\rho-\lambda)^2}\right) \) when executed against the jamming adversary of type \((\rho, \lambda, b)\).

**Proof:** We compare packet latency of protocol C-RRW to that of protocol OFC-RRW. Consider the executions of protocols C-RRW and OFC-RRW for the same injection and jamming pattern of the adversary. Let \( s_i \) and \( t_i \) be the bounds on the length of phase \( i \) of protocols C-RRW and OFC-RRW, respectively.

Phase \( i \) of C-RRW takes \( s_i \) rounds, so when protocol OFC-RRW is executed, its phase \( i \) may take up to

\[
s_i + s_i(\rho + \lambda) + s_i(\rho + \lambda)^2 + \ldots = \frac{s_i}{1-(\rho + \lambda)} \tag{5.11}
\]

rounds. So the phase’s length of protocol C-RRW differs from that of protocol OFC-RRW by a factor of at most \( \frac{1}{1-\rho-\lambda} \). An injected packet is transmitted in at most two phases of an execution, for each protocol. The bound of Theorem 3 is twice the bound on the duration of a phase. It follows that a bound on packet latency of protocol C-RRW can be obtained by multiplying the bound given in Theorem 3 by \( \frac{1}{1-\rho-\lambda} \). \( \square \)

The tightness of the bound on packet latency given in Theorem 4 can be established similarly as that for Theorem 2. To this end, replace \( n(J+1) \) by \( n \) in the tightness argument for the bound of Theorem 2 given in Section 5.1.1 to obtain \( \Omega\left(\frac{n+b}{(1-\rho-\lambda)^2}\right) \) as a bound. The tightness of the upper bound of Theorem 3 follows from the property that the derivation of the upper bound can be closely mimicked by the adversary. Alternatively, we observe that an improvement of the bound in Theorem 3 would lead to an improvement to the bound in Theorem 4, which has already been shown to be tight.
Protocol MOVE-BIG-TO-FRONT: Next we estimate packet latency of protocol MBTF against jamming. The analysis we give resorts to estimates of the number of packets stored in the queues at all times.

Let a traversal of the token, starting at the front of the list and ending again at the front station of the list, be called a *pass* of the token. A pass is concluded by either discovering a new big station or traversing the whole list. We measure the number of packets in the queues at the end of a pass, to see if the pass contributed to an increase of the number of packets stored in the queues or not; a former pass is called *increasing* and a latter one *non-increasing*, respectively. We partition passes into two categories, depending on whether a big station is discovered in the pass or not; a former pass is called *big* and a latter *small*. To make this terminology precise, a discovery of a big station contributes to the following two consecutive passes. The first pass is concluded by the discovery of a big station, but the big station just found does not transmit in this pass. The second pass begins by a transmission of the newly discovered big station, just after it has been moved up to the front position in the list.

**Lemma 3** When protocol MBTF is executed by *n* stations against the jamming adversary of type \((\rho, \lambda, b)\) then the number of packets stored in queues in any round is at most

\[
\frac{2p(n+b)}{(1-\lambda)(1-\rho-\lambda)} + O(bn).
\]

**Proof:** We first investigate how many packets can be accumulated in queues when small passes occur. It is sufficient to consider increasing small passes only, as otherwise the previous passes contribute bigger counts of packets. Suppose some *i* stations with nonempty queues are passed by the token. As each of them contributes to a packet heard on the channel, there are *i* packets heard in the pass. A pass that is not jammed takes *n* rounds, but with jamming it may take longer as the jammed rounds slow the protocol down. There are at most

\[
b + n + n\lambda + n\lambda^2 + n\lambda^3 \ldots \leq \frac{n}{1-\lambda} + b
\]

rounds in a small pass. During such a pass, the number of injected packets is at most
\[(\frac{n}{1-\lambda} + b)\rho + b\]. It follows that \(i \leq (\frac{n}{1-\lambda} + b)\rho + b\). Each station passed has at most \(n - 1\) packets, because the pass is small. It follows that a small pass is either increasing or there are up to
\[
\left(\left(\frac{n}{1-\lambda} + b\right)\rho + b\right)(n - 1) = \frac{\rho n(n + b)}{1 - \lambda} + O(bn)
\]
(5.12) packets in the queues when the pass is over.

Next we consider the question how much above the upper bound (5.12) can the queues grow during big passes. Consider a time interval \(T\) when a maximum number of packets is accrued while some \(k\) stations are discovered at least once as big. To obtain estimates from above on the total number of packets in the queues, we conservatively assume the following: (i) when a small station is passed by the token then there are no packets in the queue of the station and the resulting round is a control one, and (ii) when a big station is discovered and moved up to the front of the list, then only one packet is transmitted by the station and then the token immediately moves on to the next station.

Consider a series of consecutive big passes. The number of control rounds in any big pass is at most \(n - 1\) followed by the big station moved to the front of the list. In the first pass, there are at most \(n - 1\) control rounds. In the second pass, the station discovered big in the first pass transmits a packet, which is next followed by at most \(n - 1\) control rounds until a new big station is discovered. In the third pass, the two big stations discovered so far transmit at least one packet each, as they are at the front of the list, which is followed by at most \(n - 2\) control rounds before the third big station is discovered. This pattern continues until the last pass which begins by all the \(k\) big stations residing in the initial segment of the list and so each of them transmitting one packet in this pass followed by at most \(n - k\) control rounds. Let \(t\) be the last round when the above last control round among at most \(n - k\) occurs.

When the next pass begins in round \(t + 1\), then either the pass is small or a big station among the first \(k\) stations in the list is discovered, as these are the stations discovered as big in time interval \(T\). In the former case, the upper bound (5.12) on the number of packets in
the queues applies. In the latter one, there are no control rounds during the pass. It follows that an upper bound is obtained by estimating from above an increase of packets in the time interval $T$ by round $t$ and next adding to it the bound (5.12) as a possible starting point of the process of increasing queues in $T$.

Next we estimate the increase of packets in $T$ by round $t$. There are at most $(k+1)(n-1)$ control rounds and $1+2+3+\ldots+k = k(k+1)/2$ packets transmitted in packet rounds. At the same time, the adversary may be injecting packets and jamming rounds, with a net increase contributed by control and jammed rounds. The increase is at most

$$\frac{\rho}{1-\lambda} \left[ (k+1)(n-1) + \frac{k(k+1)}{2} \right] + b - \frac{k(k+1)}{2}$$

$$= \frac{\rho}{1-\lambda} kn - \frac{1}{2} (1 - \frac{\rho}{1-\lambda}) k^2 + O(n+b).$$

We seek to maximize the function

$$f(k) = \frac{\rho n}{1-\lambda} - \frac{1}{2} (1 - \frac{\rho}{1-\lambda}) k^2,$$

which we find to occur at the argument

$$k_{\text{max}} = \frac{\rho n}{1 - \rho - \lambda}$$

by the standard maximum finding procedure based on differentiation. By algebraic manipulation, we obtain that the values $f(k)$ are at most

$$f(k_{\text{max}}) \leq \frac{\rho n^2}{2(1-\lambda)(1-\rho-\lambda)}$$

and the increase of packets in $T$ by round $t$ is at most this number plus $O(n+b)$. We combine this estimate with bound (5.12) to obtain

$$\frac{\rho n(n+b)}{1-\lambda} + \frac{\rho n^2}{2(1-\lambda)(1-\rho-\lambda)} + O(bn)$$

$$\leq \frac{2\rho n(n+b)}{(1-\lambda)(1-\rho-\lambda)} + O(bn)$$

as a bound on the total number of queued packets. These estimates are with respect to what is the number of packets at the ends of passes. Possible fluctuations of the number of
packets in queues during a pass cannot be more than \( n + b \), which is within the magnitude of the asymptotic component \( O(bn) \) of the bound. □

When big stations get discovered then the regular round-robin pattern of token traversal is disturbed. Suppose a station \( i \) is discovered as big, which results in moving the station back to the front. The clear rounds that follow, before the token either moves to position \( i + 1 \) or a new big station is discovered, whichever occurs first, are called delay rounds. These rounds are spent, first, on transmissions by the new first station, to bring the number of packets in its queue down to \( n - 1 \), and next, the token needs \( i \) additional clear rounds to move to the station at position \( i + 1 \). Given a round \( t \) in which a packet \( p \) is injected, take a snapshot of the queues in this round. Let \( q_i \) be the number of packets in a station \( i \) in this snapshot, for \( 1 \leq i \leq n \). We associate credit of the amount \( \max\{0, q_i - (n - i)\} \) with station \( i \) at this round. A station \( i \) has a positive credit when the inequality \( q_i \geq n - i + 1 \) holds. In particular, a big station \( i \) has credit \( q_i + i - n \geq i \). Let \( C(n, t) \) denote the sum of all the credits of the stations in round \( t \). We consider credit only with respect to the packets already in the queues in round \( t \), unless stated otherwise.

**Lemma 4** If discovering a big station in round \( t \) delays the token by some \( x \) rounds, excluding jammed rounds, then the amounts of credit satisfy \( C(n, t+x) = C(n, t) - x \).

**Proof:** We refer to stations by their positions just before the shift, unless stated otherwise. When a big station \( i \) is moved up to the front of the list, its credit gets decreased first by \( i - 1 \) by the change of position in the list, and next by the amount equal to the number of transmitted packets by the new first station. More precisely, \( q_i - (n - 1) \) packets are transmitted in these many rounds to decrease credit of the new first station from \( q_i + 1 - n \) to zero, a unit of credit spent in each round.

The decrease of the amount of credit by \( i - 1 \) is to pay for the travel of the token to position \( i + 1 \). There is a problem of what happens to the credit at the traversed stations that changed their positions. The stations at the original positions 1 through \( i - 1 \) get shifted by one position down the list to occupy the positions 2 through \( i \). Consider such
a station \( j \), for \( 1 \leq j \leq i - 1 \), when it is shifted one position down the list. Its potential stays equal to 0, if \( q_j < n - j \), but it gets incremented by 1 if \( q_j \geq n - j \), as it then equals \( q_j + (j + 1) - n \geq 1 \).

When a packet is transmitted by a station that does not hold any credit, then this does not affect the total amount of credit, as the credit contributed by this station stays equal to zero. A packet transmitted by a station with a positive credit decrements the credit by 1, which restores the amount of credit held by the station before the shift down. This means that the total decrease of the amount of credit equals the number of rounds in the time period of delay.

\[ \square \]

**Theorem 5** The packet latency of protocol MBTF is at most \( \frac{3n(n+b)}{(1-\lambda)(1-p-\lambda)} + O\left( \frac{bn}{1-\lambda} \right) \) when executed against the jamming adversary of type \((p, \lambda, b)\).

**Proof:** Let a packet \( p \) be injected in some round \( t \) into station \( i \). Let \( S(n,t) \) be an upper bound on the number of rounds that packet \( p \) spends waiting in the queue at \( i \) to be heard when no big station is discovered by the time packet \( p \) is eventually transmitted. Some additional waiting time for packet \( p \) is contributed by discoveries of big stations and the resulting delays; denote by \( T(n,t) \) the number of rounds by which \( p \) is delayed this way. The total delay of \( p \) is at most \( S(n,t) + T(n,t) \).

First, we find upper bounds on the expression \( S(n,t) + T(n,t) \) that does not depend on \( t \) but only on \( n \). The inequality

\[ S(n,t) \leq \frac{n(n-1)}{1-\lambda} \tag{5.13} \]

holds because there are at most \( n-1 \) packets in the queue of \( i \) and each pass of the token takes at most \( n \frac{n}{1-\lambda} \) rounds. The inequality

\[ T(n,t) \leq \frac{C(n,t)}{1-\lambda} \tag{5.14} \]

holds because of Lemma 4 and the fact that iterated delays due to jamming contribute the factor \( 1/(1-\lambda) \). Observe that \( C(n,t) \) is upper bounded by the number of packets in the
queues in round $t$, by the definition of credit. Therefore, we obtain that

$$C(n,t) \leq \frac{2\rho n(n+b)}{(1-\lambda)(1-\rho-\lambda)} + O(bn),$$

by Lemma 3. This, along with the estimate (5.14), implies

$$T(n,t) \leq \frac{2\rho n(n+b)}{(1-\lambda)^2(1-\rho-\lambda)} + O\left(\frac{bn}{1-\lambda}\right). \quad (5.15)$$

Combine (5.13) and (5.15) to obtain that a packet waits at most

$$\frac{n(n-1)}{1-\lambda} + \frac{2\rho n(n+b)}{(1-\lambda)^2(1-\rho-\lambda)} + O\left(\frac{bn}{1-\lambda}\right) \leq \frac{3n(n+b)}{(1-\lambda)(1-\rho-\lambda)} + O\left(\frac{bn}{1-\lambda}\right)$$

rounds, where we used $\rho < 1-\lambda$ and $1-\rho-\lambda < 1$. □

5.2 Conclusion

We introduced a model of adversarial packet injection and jamming to study deterministic distributed protocols for medium-access control. Jamming is represented by collisions that preclude successful transmissions of packets in jammed rounds. Terminals cannot distinguish between collisions caused by simultaneous transmissions and jamming, which captures scenarios in which groups of stations interfere with one another by independent transmissions. Adversaries are restricted only by the requirement that the conglomerate rate obtained by summing the injection rate with the jamming rate is less than one.

We showed that full sensing protocols can have bounded packet latency if only the adversaries are restricted by their jamming burstiness which is part of code of the protocol.
In contrast to that, general adaptive protocols can attain bounded packet latency for any adversary. For each considered protocol we derived worst-case upper bounds on their packet latency in terms of the size of the system and the type of an adversary.
6. Channels without jamming

We consider deterministic distributed protocols in channels with no jamming. We study the performance of full sensing and adaptive protocols against leaky-bucket adversaries with injection rates \( \rho < 1 \). We consider channels with and without collision detection. For each of the protocols we consider, we give upper bounds for packet latency as functions of the number of stations \( n \) and the type \((\rho, b)\) of a leaky-bucket adversary.

6.1 Full sensing protocols

We consider the packet latency for full sensing protocols RRW and OF-RRW. They operate similarly as the adaptive protocols C-RRW and OFC-RRW for a channel with jamming, respectively. The difference is in how the virtual token moves from station to station and what happens in jammed rounds. In adaptive protocols with jamming, the virtual token is moved when the station with token transmits a control bit. In full sensing protocols, the token is moved when the station with token pauses. In adaptive protocols with jamming, the jammed rounds is perceived as silence and just ignored. This facilitates translating the bounds for jammed channel to the channel without jamming, for the respective protocols. The following results are obtained by such translations.

**Corollary 1** The packet latency of protocol OF-RRW is \( O \left( \frac{n+b}{1-\rho} \right) \) when executed against the adversary of type \((\rho, b)\).

**Proof:** The upper bound on packet latency given in Theorem 3 becomes \( O \left( \frac{n+b}{1-\rho} \right) \) for \( \lambda = 0 \). \(\square\)

**Corollary 2** The packet latency of protocol RRW is \( O \left( \frac{n+b}{(1-\rho)^2} \right) \) when executed against the adversary of type \((\rho, b)\).

**Proof:** The upper bound on packet latency given in Theorem 4 becomes \( O((n+b)/(1-\rho)^2) \) for \( \lambda = 0 \). \(\square\)

These bounds are tight. The argument for tightness of bounds for the protocols with jamming hold for the full sensing protocols without jamming when the jamming rate is \( \lambda = 0 \).
### 6.1.1 Full sensing protocols with collision detection

We consider packet latency for protocols SEARCH-ROUND-ROBIN (SRR) and OLD-FIRST-SEARCH-ROUND-ROBIN (OF-SRR) which use collision detection. We begin with a technical estimate that will prove to be useful in proving bounds on packet latency. Let $\log x$ denote $\lceil \log_2 x \rceil$.

**Lemma 5** If there are $x \geq 1$ packets in the system in a round then, for any $1 \leq y \leq x$, protocols SRR and OF-SRR make $y$ packets heard in the next $\min(y \log n, 2n + y)$ rounds.

**Proof:** We estimate in two different ways how much time it takes to identify stations with $y$ old packets. We partition an execution into consecutive *epochs*, where an epoch is defined as a minimum period in which the list of stations cycles. Note that for protocol OF-SRR an epoch is the same as a phase.

We argue first that any of the considered protocols takes no more than $2n + y$ consecutive rounds to transmit successfully at least $y$ packets. It takes at most $n$ rounds with no transmission to identify stations that store at least $y \leq x$ packets for SRR; to this end, let the token make at most one full cycle along the list of stations. Similarly, it takes at most $2n$ rounds with no transmission to identify stations that store at least $y \leq x$ packets for OF-SRR. This is because two cycles are needed in the worst case, as after the first cycle at least $y \leq x$ packets become old at the latest, and the next cycle will reach stations storing them. Having shown this, we can see that uploading these at least $y$ packets, that are old for the OF-versions of the protocols, takes $y$ rounds. This completes the first part of the argument.

Next, we prove that these protocols, based on binary searching, need no more than $y \log n$ consecutive rounds to upload at least $y \leq x$ packets. Indeed, after a packet has been transmitted, it takes time at most the height of a conceptual search tree to identify another station with a packet, that is old for OF-SRR. This height is at most $\log n$, so a packet is heard at least once in any contiguous segment of $\log n$ rounds. Therefore, $y$ packets are transmitted within a segment of $y \log n$ rounds. \qed
Theorem 6 The packet latency for protocol OF-SRR executed against the adversary of type \((\rho, b)\) is at most \(4\min(b \log n, n + b)\) for \(\rho \leq 1/(2\log n)\), and it is at most \(4n + 2b\) for \(\rho > 1/(2\log n)\).

Proof: A packet is transmitted successfully by the end of the phase following the one in which it was injected. So the maximum packet latency is upper bounded by twice the maximum length of a phase. At most \(\rho |\tau| + b\) packets get injected in a time interval \(\tau\) of length \(|\tau|\). The maximum number of packets in the system is upper bounded by the maximum number of packets at the end of an interval plus the number of packets injected within this interval.

The case of \(\rho \leq 1/(2\log n)\): Consider a round. We first argue that at the end of this round there are at most \(\rho \cdot \min(2b \log n, 2n + 2b) + b\) packets in the system. Suppose it is otherwise and consider the first round \(t\) with the number of packets more than \(\rho \cdot \min(2b \log n, 2n + 2b) + b\) at the end of it. Note that \(t > \min(2b \log n, 2n + 2b)\), as at most \(\rho \min(2b \log n, 2n + 2b) + b\) packets are injected into the system in the time interval \([1, \min(2b \log n, 2n + 2b)]\). This follows from the assumption that, at the end of round \(t - \min(2b \log n, 2n + 2b)\), there were at most

\[
\rho \cdot \min(2b \log n, 2n + 2b) + b
\]

packets in the system. From that round until round \(t\), at least this number of packets have been successfully transmitted, by Lemma 5 and the estimate

\[
\rho \cdot \min(2b \log n, 2n + 2b) + b \leq \frac{2b \log n}{2\log n} + b = 2b,
\]

while at most \(\rho \cdot \min(2b \log n, 2n + 2b) + b\) new packets have been injected. Consequently, the number of packets in the system at the end of round \(t\) would be at most \(\rho \cdot \min(2b \log n, 2n + 2b) + b\), which is a contradiction. This completes the proof that the number of packets in the system is at most \(\rho \cdot \min(2b \log n, 2n + 2b) + b\). This bound is also at most \(2b\). By Lemma 5, a phase does not take longer than \(\min(2b \log n, 2n + 2b)\) rounds, as the number of old packets in the system never surpasses \(2b\). Hence, the packet latency is at most \(2 \cdot \min(2b \log n, 2n + 2b) \leq 4 \min(b \log n, n + b)\).
The case of $\rho > 1/(2\log n)$: Partition an execution into consecutive phases, as in the specification of the protocol. Let $x_i$ be the number of packets in the system in the beginning of phase $i$, and let $t_i$ be the length of this phase. By Lemma 5, the inequalities $x_1 \leq b$ and $t_i \leq \min(x_i \log n, 2n + x_i)$ hold, for every $i \leq 1$. Therefore

$$x_{i+1} \leq \rho t_i + b \leq \rho \min(x_i \log n, 2n + x_i) + b.$$ 

Start iterating this formula to obtain

$$x_{i+1} \leq b + \rho \min(x_i \log n, 2n + x_i)$$

$$\leq b + \rho (2n + (b + \rho \min(x_{i-1} \log n, 2n + x_{i-1})))$$

$$= b + \rho (2n + b) + \rho^2 \min(x_{i-1} \log n, 2n + x_{i-1}),$$

which can be continued to obtain

$$x_{i+1} \leq b + (2n + b)(\rho + \rho^2 + \ldots + \rho^{i-1})$$

$$\leq \frac{\rho}{1 - \rho} \left(2n + \frac{b}{\rho}\right) = \frac{2n \rho + b}{1 - \rho}.$$

The packet latency is the maximum of $t_i + t_{i+1}$, over all $i$. By Lemma 5, these numbers are at most

$$\min(x_i \log n, 2n + x_i) + \min(x_{i+1} \log n, 2n + x_{i+1}).$$

Each of them is upper-bounded by

$$4n + 2 \cdot \frac{2n \rho + b}{1 - \rho} \leq \frac{4n + 2b}{1 - \rho},$$

which completes the proof. □

**Theorem 7** The packet latency of protocol SRR executed against the adversary of type $(\rho, b)$ is at most $6b \log n$ for $\rho \leq \frac{1}{2\log n}$, and at most $\frac{4(n+b)}{(1-\rho)^2}$ for $\rho > \frac{1}{2\log n}$.

**Proof:** We partition an execution into consecutive epochs, each being a full sweep of a search for packets across all the stations. Let $x_i$ be the number of packets in the system in the beginning of epoch $i$, and let $t_i$ be the length of this epoch.
We first consider the case of \( \rho \leq \frac{1}{2 \log n} \). By a similar argument as in the proof of Theorem 6, the number of pending packets is never more than

\[
\rho \cdot \min(2b \log n, 2n + 2b) + b \leq 2b .
\]

This argument relies only on Lemma 5. Next, observe that

\[
t_i \geq x_i + \rho t_i + b \geq t_i / \log n .
\]

This is because the number of all packets in the system in epoch \( i \) multiplied by the maximum time \( \log n \) of a void period in epoch \( i \) is an upper bound on \( t_i \). There are at most \( \rho t_i + b \) newly arrived packets in this epoch. Combine this with the estimate \( \frac{t_i}{2 \log n} \geq \rho t_i \) to obtain \( (x_i + b) \log n \geq t_i \). By an upper bound \( 2b \) on the number of packets in the system, in the case of \( \rho \leq \frac{1}{2 \log n} \), we obtain \( t_i \leq (x_i + b) \log n \leq 3b \log n \).

Every packet arriving in epoch \( i \) is transmitted by the end of epoch \( i + 1 \). Hence the maximum packet latency is at most

\[
\max_i (t_i + t_{i+1}) \leq 2 \cdot 3b \log n \leq 6b \log n .
\]

In order to estimate packet latency in the case of \( \rho > 1/(2 \log n) \), observe that \( t_i \geq x_i + (\rho t_i + b) \geq t_i - n \). This is because the number of all packets that are in the system in epoch \( i \) plus the maximum total number \( n \) of void rounds in epoch \( i \) is an upper bound on \( t_i \). There are at most \( \rho t_i + b \) newly arrived packets in this epoch, so \( \frac{x_i + n + b}{1 - \rho} \geq t_i \). Lemma 5 for protocol SRR implies that \( \frac{2pn + b}{1 - \rho} \) is an upper bound on the number of packets in the system in a round. It follows that

\[
t_i \leq \frac{x_i + n + b}{1 - \rho} \leq \frac{n(1 + \rho) + b(2 - \rho)}{(1 - \rho)^2} .
\]

A packet arriving in epoch \( i \) is transmitted by the end of epoch \( i + 1 \), hence the maximum packet latency is upper bounded by

\[
\max_i (t_i + t_{i+1}) \leq 2 \cdot \frac{n(1 + \rho) + b(2 - \rho)}{(1 - \rho)^2} \leq \frac{4(n + b)}{(1 - \rho)^2} ,
\]

which completes the proof. \( \square \)
6.1.2 Adaptive protocol

We consider packet latency of protocol MOVE-BIG-TO-FRONT (MBTF), which is an adaptive protocol for channels without collision detection. This protocol was originally designed for channels without jamming, but using control bits in messages without packets allows to transfer the token without unnecessary delay even when a channel is subject to jamming. We may consider any of these two versions of the protocol when there is no jamming in a channel. A translation of the bound on packet latency applies that is similar to the case of full sensing protocols for channels without collision detection.

**Corollary 3** There are at most \( \frac{2\rho n(n+b)}{1-\rho} + O(bn) \) packets in queues in any round when protocol MBTF is executed against the adversary of type \((\rho, b)\).

**Proof:** Follows from Lemma 3 when \( \lambda = 0 \). \(\square\)

**Corollary 4** The packet latency of protocol MBTF is \( O(\frac{n(n+b)}{1-\rho}) \) when executed against the adversary of type \((\rho, b)\).

**Proof:** The bound on packet latency given in Theorem 5 becomes \( O(\frac{n(n+b)}{1-\rho}) \) for \( \lambda = 0 \). \(\square\)

6.2 Conclusion

We considered deterministic protocols for channels without jamming and leaky bucket adversaries of type \((\rho, b)\), for \( \rho < 1 \). The results show that bounded packet latency can be achieved by full sensing protocols without knowing the types of the adversary. This is in contrast to the channels with jamming, where jamming burstiness was part of the code. The results for the protocols in channels with jamming, that use token traversal translate nicely when jamming rate is 0, to the ones in channels without jamming. We introduced an ‘old-first’ version of protocol that uses binary search when collision detection is available. The results on packet latency again indicate the benefit of the paradigm ‘old-first’ in the design of protocols.
7. Individual injection rates

We consider adversaries with individual stations’ rates; in this model the adversary’s injection rate is constrained for each station separately. For recent work on adversarial queuing for multiple access channels, see [11, 30, 31], concerned adversaries constrained by ‘global’ injection rates, in the sense that the adversary is restricted by the number of packets injected into all stations but not by how many packets are injected into any specific station. An adversary with injection rates associated with individual stations is more restricted than one with a ‘global’ injection rate. One may observe that the (global) injection rate of one packet per round is the maximum rate that allows for a stable protocol. We understand the throughput of a protocol as the maximum (global) rate for which it is stable, which can be less than 1.

In the context of adversarial queuing in store-and-forward networks, ‘globally constrained’ window adversaries were first used by Borodin et al. [26] and similar leaky-bucket ones by Andrews et al. [14]. For the multiple access channel, Chlebus et al. [31] used window adversaries with injection rates less than 1, which does not restrict the generality of results for such rates. Chlebus et al. [30] used both models for ‘global’ injection rate 1; for such ‘global’ rate the window adversarial model is strictly weaker than the leaky-bucket one [30, 60].

We consider constraints on adversaries formulated as restrictions on what can be injected separately at each station and also by what can be injected into all the stations combined. Constraints are global if they are determined only by the numbers of packets injected per time intervals, without any concern about the stations in which the packets are injected. Traffic constraints are local when the patterns of injection are considered separately and independently for each station. Constraints for the whole network implied by local ones are called aggregate in this paper. In particular, we may have local and aggregate and global burstiness, and local and aggregate and global injection rates. Observe that global constraints are logically weaker than aggregate constraints. Global and aggregate injection rate 1 is the maximum that a channel can handle in a stable way, since
stability provides that the throughput rate is at least as large as the injection rate. We refer to adversaries defined only by global restrictions on injection rates as the model of global injection rate, these adversaries were used in [30, 31] in the context of multiple access channels. In this paper we introduce adversaries of local injection rates.

To categorize protocols, we use the terminology similar to that in [31, 30]. General protocols are called full sensing. Protocols may use control bits piggybacked on transmitted packets; such protocols are called adaptive.

7.1 Results

We study the worst-case performance of broadcasting when traffic demands are specified as adversarial environments modeled by adversaries with individual injection rates associated with stations and when protocols are both distributed and deterministic. We allow the adversaries to be such that the associated aggregate injection rate (the sum of all the individual rates) is 1, which is the maximum that allows for stability. The goal is to explore what quality of service can be achieved for individual injection rates and compare the adversarial environments defined by individual rates versus global ones, under maximum broadcast loads of one packet per round. The underlying motivation for this work was that individual injection rates are more realistic in moderate time spans and hopefully the limitations on quality of service with throughput 1 discovered in [30] could be improved when the rates are individual. Indeed, bounded packet latency turns out to be achievable with individual injection rates when the aggregate rate is 1. This is in contrast with global injection rates for which achieving bounded waiting times is impossible for throughput 1, as was shown in [30].

We show upper and lower bounds on queue size and packet latency, which depend on the classes of protocols and whether collision detection is available or not. We have the following results for a window-type adversary. The acknowledgment based protocols cannot achieve throughput 1, which strengthens the result for global injection rates [30]. We give a non-adaptive full sensing protocol of $O(\min(n + w, w \log n))$ packet latency when collision detection is available. An adaptive full sensing protocol can achieve similar
performance without collision detection, as we show that control bits allow to simulate collision detection with a constant overhead per round. Bounded packet latency can also be achieved by non-adaptive full sensing protocols in channels without collision detection. More precisely, we develop a non-adaptive full sensing protocol with $O(n+w)$ size queues and $O(nw)$ packet latency. The optimality of our non-adaptive full sensing protocol for channels without collision detection in terms of packet latency is left open.

7.2 Preliminaries

The specifications of the channels and classes of protocols remain same as before. Next we explain constraining the adversary with individual injection rates at stations.

**Adversaries with individual injection rates:** An adversary is defined by a set of allowable patterns of injections of packets into stations. An adversary generates a number of packets in each round and assigns to each packet the station into which the packet is injected.

We first review the traditional globally restricted adversaries. Global constraints on packet injection are usually modeled by either window or leaky-bucket adversarial models. A window-type adversary that is restricted by an injection rate $\rho$ and a window size $w$. The window size $w$ is used to define the injection rate $\rho$ as follows: for any contiguous time interval $\tau$ of $w$ rounds the number of packets injected in $\tau$ is at most $\rho w$. A leaky-bucket type of an adversary is restricted by an injection rate $\rho$ and a burstiness $b$. The meaning of these parameters is such that the adversary may insert at most $|\tau|\rho + b$ packets during any time interval $\tau$. We restrict such constraints further by specifying how packets are injected into each station separately; in this we consider only window adversaries.

**Window adversaries with individual injection rates** are defined as follows: Let there be given a positive integer number $w$ called window size. Each station $i$ has its share $s_i$, which is a non-negative integer. The shares satisfy the requirement $\sum_{i=1}^n s_i \leq w$. The adversary may insert up to $s_i$ packets into station $i$ in a time interval of $w$ contiguous rounds. The local injection rate of station $i$ is defined to be the number $\rho_i = s_i/w$. The aggregate injection rate is defined to denote the number $\rho = (\sum_{i=1}^n s_i)/w$. We refer to such
a window adversary as being of *local type* \((s_i)_{1 \leq i \leq n}, w\) and of *aggregate type* \((\rho, w)\).

The notion of aggregate type is similar to that of global type as considered in [30, 31]. For a window adversary of a local type \((s_i)_{1 \leq i \leq n}, w\) with the aggregate rate \(\rho\), the *local burstiness at station* \(i\) is defined to be its share \(s_i\) and the *aggregate burstiness* is defined to be the number \(\delta = \sum_{i=1}^{n} s_i\), so that \(\rho = \delta / w\).

**Protocol design:** Now we give an overview of design principles of our protocols. It is a natural approach to have stations work to discover the parameters defining the adversary at hand. The stations could gradually improve their estimates of the shares \(s_i\) and the aggregate maximum burstiness \(\delta = \sum_{i=1}^{n} s_i\). The stations would adjust the number of their individual transmissions to the knowledge gained in the process of discovery of the adversary. The minimum-size window of an adversary is determined by the aggregate burstiness. Observe that the aggregate burstiness \(\delta\) in the case of injection rate 1 equals the minimum window size for the adversary at hand.

For a given window adversary of local injection rates, station \(i\) is *active* when its share \(s_i\) is a positive number, otherwise, when \(s_i = 0\), the station \(i\) is *passive*. Station \(i\) has been *discovered* in the course of an execution when a packet transmitted by \(i\) has been heard on the channel. A discovered station is clearly active. In the context of a specific execution, a station that has not been discovered by a given round is called a *candidate* in this round. A passive station is doomed to be candidate forever. We describe two data structures used when stations work to discover their shares.

In one approach, each station \(i\) has a private array \(C_i\) of \(n\) entries. The entry \(C_i[j]\) stores an estimate of the share \(s_j\) of station \(j\), for \(1 \leq j \leq n\). Every station \(i\) will modify the entry \(C_i[k]\) in a round in the same way as other stations do. Therefore we may drop the indices and refer to the entries of the arrays as \(C[j]\) rather than as \(C_i[j]\). The array \(C\) is initialized to \(C[j] = 0\) for every \(1 \leq j \leq n\). The sum \(\gamma = \sum_{i=1}^{n} C[i]\) is an estimate of the aggregate burstiness \(\delta = \sum_{i=1}^{n} s_i\).

The stations running a protocol keep adjusting the estimates stored in the array \(C\). When station \(i\) enters a state implying \(s_i > C[i]\) then \(i\) considers itself *underestimated.*
Detecting underestimation is implemented by having every station keep track of the transmissions in the past $\gamma$ rounds. When a station $i$ detects that some $k > C[i]$ packets have been injected within the respective $\gamma$ consecutive rounds, where $k$ is maximum with this property so far, then $i$ decides it is underestimated by the amount $k - C[i]$ in this round. This conservative mechanism of estimating the shares is safe, in that the shares are never overestimated, as we show next.

**Lemma 6** If a protocol against a window adversary has the property that $C[i]$ is increased at most by the amount that station $i$ considers to be underestimated by, then the inequality $C[i] \leq s_i$ holds at all times, for every $1 \leq i \leq n$.

**Proof:** We show that the invariant ‘the inequality $C[i] \leq s_i$ holds for every $1 \leq i \leq n$’ is preserved throughout the execution. Initially $C[i] = 0 \leq s_i$ for every $1 \leq i \leq n$, so the invariant holds true in the first round. Observe that if the inequalities $C[i] \leq s_i$ hold for every $1 \leq i \leq n$, then also the inequalities

$$\gamma = \sum_{i=1}^{n} C[i] \leq \sum_{i=1}^{n} s_i = \delta \leq w$$

hold true. The adversary can inject at most $s_k$ packets into a station $k$ during any segment of $\gamma$ contiguous rounds, as the inequality $\gamma \leq w$ means that any segment of $\gamma$ contiguous rounds could be extended to one of $w$ rounds of the execution. A station $k$ can detect to be underestimated by at most $s_k - C[k]$ in any round while the invariant holds. This makes the invariant still hold in the next round, because an update of $C[k]$ raises $C[k]$ to a value that is at most $s_k$. $\square$

Another approach to protocol design is to have each station $i$ use a list $D_i$ of bits. Such a list $D_i$ has its terms listed as follows $\langle d_i(1), d_i(2), \ldots, d_i(\ell) \rangle$, where the length $\ell$ may be modified by it is the same at all stations, and additionally the inequality $\ell \leq w$ holds at all times. The lists are initialized to be empty. The lists are maintained so as to satisfy the following invariants:

(1) for each $1 \leq j \leq \ell$, when $d_i(j)$ has been determined for all $i$, then $d_i(j) = 1$ for
(2) the number of occurrences of 1 at \(D_i\) has the same meaning as \(C[i]\) and denotes the current estimate of the share of \(i\).

The number \(\ell\) will be an estimate on the burstiness of the adversary, similarly as the number \(\gamma\) is for the array \(C\). When lists \(D\) are only used and arrays \(C\) are not explicitly maintained, then \(C[i]\) will denote the number of occurrences of 1 in \(D_i\). We will use \(\ell\) and \(\gamma\) interchangeably when lists \(D_i\) are used. If a protocol has the property that \(C[i]\) is increased at most by the amount that station \(i\) considers to be underestimated by, then the inequality \(C[i] \leq s_i\) holds at all times, for every \(1 \leq i \leq n\). Our protocols will have the property that the assumptions of Lemma 6 hold true, and for them the inequality \(\gamma \leq \delta\) holds.

We use arrays \(C\) when each station knows of the station for which update of an estimate of its share is to be performed. It may happen that such knowledge is lacking: an underestimated station transmits successfully but the other stations do not know which station transmitted. This is a scenario in which to use lists \(D\), as a transmitting station can append 1 at the end of \(D\) while every other station appends 0.

Our protocols have stations manipulate auxiliary lists. Whenever we use such lists, there is a main pointer associated with each list pointing at a current entry. The main pointer either stays put in a round or it is advanced by one position in the round in the cyclic ordering of the entries on the list.

7.3 Impossibilities and lower bounds

In this section, we present impossibility results for acknowledgment-based protocols and lower bounds on packet latency.

7.3.1 Impossibility results for acknowledgment based protocols

We begin by examining acknowledgment-based protocols. The action that a station performs when it begins processing a new packet is always the same, as it is determined by the initial state. Such an action maybe either ‘to transmit’ or ‘to pause’ in the first round.
Lemma 7 Consider an acknowledgment-based protocol executed by a set of stations. If there is a station which pauses in the first round after starting to process a new packet then, for any number $\rho > \frac{1}{2}$, the protocol is unstable for some adversary with the aggregate rate equal to $\rho$.

Proof: Suppose that some station $p$ pauses in the first round of processing a new packet. Consider an adversary injects only into such a station $p$ as often as possible subject to an individual injection rate that is between $\frac{1}{2}$ and $\rho$. This results in an execution in which a packet is heard not more often than every second round, while in aggregate rate is greater than $\frac{1}{2}$, so the queue at $p$ grows unbounded. \hfill \Box

Theorem 8 Any acknowledgment-based protocol is unstable in a system of just two stations and a multiple-access channel with collision detection against some window adversary of burstiness 2 and aggregate injection rate 1.

Proof: Define a round to be void when no packet is heard in the round. Consider an acknowledgment-based protocol for two stations $p$ and $q$. Suppose, to arrive at a contradiction, that the protocol is stable for the aggregate injection rate 1. This implies, by Lemma 7, that a station transmits a new packet immediately.

We define an execution of the protocol with an infinite sequence of rounds $t_1, t_2, \ldots$ determined so that there are at least $i$ void rounds by round $t_i$. The adversary will inject two packets in each odd-numbered round, a packet per station, and no packets in even-numbered rounds. This means that each station has its individual injection rate equal to $\frac{1}{2}$. The aggregate injection rate of the adversary is 1, and so there are at least $i$ packets in queues in round $t_i$.

Define $t_1$ to be the first round. This round is silent as stations did not have any packets prior to this round. A collision occurs in the second round: each station got a new packet in the first round, so it transmits it immediately. Define $t_2$ to be the second round. Each of the round numbers $t_i$ will even for $i > 1$. The adversary will not inject packets in these specific rounds $t_i$, as packets will not be injected in any even-numbered round.
Suppose the execution has been determined through an even-numbered round $t_i$. If $t_i + 1$ is void then define $t_{i+1} = t_i + 2$. Otherwise, some station, say $p$, transmits in round $t_i + 1$. The station $p$ will continue transmitting for as long as it has packets, because it transmits a new packet immediately. If in one such a round $t$ the station $q$ transmits concurrently with $p$, then this results in a collision and we define $t_{i+1}$ to be $t$ if $t$ is even or $t + 1$ otherwise. Suppose this is not the case, so $p$ keeps transmitting alone. The station $p$ will not have a packet to transmit in some round after $t_i + 1$, since its injection rate is $\frac{1}{2}$; let $t' > t_i + 1$ be the first such a round. Observe that $t'$ has to be an odd number, as otherwise a new packet would have got injected into $p$ in round $t' - 1$ so $p$ would have a packet to transmit in round $t'$. If $q$ does not transmit in round $t'$, then this round $t'$ is silent and we are done; in this case define $t_{i+1}$ to be $t' + 1$, as a round number is to be even. Otherwise, the station $q$ transmits in round $t'$ successfully. Simultaneously both stations obtain new packets in round $t'$, so each has at least one available packet at the end of this round. Each station transmits in round $t' + 1$, as for each of them it is a first round of processing a new packet. This is because $p$ did not have a packet in round $t'$ and $q$ transmitted in round $t'$. Define $t_{i+1}$ to be the void round $t' + 1$. This completes the inductive construction of the sequence $t_i$ and by this produces a contradiction with the assumption that the protocol is stable.

Next we investigate how large could injection rate be, as a function of the number of station, what can be handled in a stable manner by the stations executing an acknowledgment-based protocol. It was shown by Chlebus et al. [31] (theorem 5.1) that an oblivious acknowledgment-based protocol cannot be stable when the global injection rate is at least as large as $\frac{3}{1 + \lg n}$, for $n \geq 4$ stations. We show a result for individual injection rates and (not necessarily oblivious) acknowledgment-based protocols. The following theorem was inspired by the related result in [31]; the proof is similar to that given in [31].

**Theorem 9** If an acknowledgment-based protocol is executed by $n \geq 4$ stations on a multiple-access channel with collision detection against adversaries whose window size $w$
can be as small as $1 + \lceil \lg n \rceil$, then the system is unstable against an adversary for which only two stations have positive shares, one such share equal to 1 and the other to 2.

**Proof:** Let $\mathcal{A}$ be a specific acknowledgment-based protocol for the $n$ stations available. For each station $p$, consider an execution of protocol $\mathcal{A}$ when $p$ starts to work on a new packet and the execution is such that each time $p$ transmits then $p$ hears collision and when $p$ does not transmit then the round is silent. Such an execution determines a sequence of bits $s(p) = (b_1, b_2, \ldots)$ such that $b_i = 1$ when $p$ transmits in the $i$th round and $b_i = 0$ when $p$ pauses in the $i$th round, counting rounds from the first one of working on the packet. Let $s(p, i)$ be this sequence truncated to the first $i$ terms, for $i \geq 1$. There exist two stations, $p_1$ and $p_2$, for which $s(p_1, \lceil \lg n \rceil - 1) = s(p_2, \lceil \lg n \rceil - 1)$ by the pigeonhole principle, because $\lceil \lg n \rceil - 1 \geq 1$ for $n \geq 4$ and $2^{\lceil \lg n \rceil - 1} = 2^{\lceil \lg n \rceil} / 2 < n$.

Let $k \geq \lceil \lg n \rceil$ be the first position in which the sequences $s(p_1)$ and $s(p_2)$ differ. Without loss of generality, suppose that the $k$th term of $s(p_1)$ is 1 and the corresponding term of $s(p_2)$ is 0. Let $j$ be the smallest position of the sequence $s(p_2)$ such that $j > k$ and there is 1 at this position in $s(p_2)$. We determine an adversary as follows. The window size $w = j$ and station $p_1$ has share 1 and station $p_2$ has share 2, which is possible for $n \geq 4$ because then $j \geq 3$.

This adversary may inject the packets as follows. At the round number 0, the adversary injects the number of packets equal to each stations share into this station. This is followed by $w$ rounds so that two packets are successfully transmitted, one at round $k$ by station $p_1$ and the other in round $w = j$ by station $p_2$, while station $p_2$ still has one packet. At the round $w = j$, the adversary injects the number of packets equal to each stations share into this station. This makes the behavior of the channel during the rounds $w + 1$ through $2w$ be the same as during the rounds from 1 through $w$, with the only difference that at the round $2w$ the station $p_2$ has two packets pending transmissions. This pattern can be iterated forever with the result that at the round number $\ell w$ the station $p_2$ has $\ell$ pending packets.

$\square$
7.3.2 Lower bounds

Next we present two lower bounds on packet latency. We begin by showing that a protocol with performance close to optimal needs to have bounds $\Omega(n + w)$ or $\Omega(w \text{ polylog } n)$ on packet latency.

**Theorem 10** For any protocol for a system of $n$ stations for an adversary of window $w$, the packet latency is $\Omega\left(w \frac{\log n}{\log w}\right)$ when $w \leq n$ and it is $\Omega(w)$ when $w > n$, in some execution of the protocol.

**Proof:** For $w \leq n$ we consider adversaries for which each station has share either 0 or 1. Greenberg and Winograd [40] considered a static version of the broadcast problem, in which some $k$ packets are located initially among $k \leq n$ stations, at most a packet per station, and the goal is to have all of them heard on the channel. They showed that for any protocol it takes $\Omega\left(k \frac{\log n}{\log k}\right)$ time to achieve this goal in some execution of the protocol. Protocols that we use handle dynamic broadcast, but can be applied to the static version. A translation to the static version of broadcast is as follows: let the adversary inject $w$ packets in the first round, at most one packet per station.

When $w > n$, then the adversary may inject $w$ packets in the first round and it takes $\Omega(w + n) = \Omega(w)$ rounds to hear them all. \[\square\]

The next observation is that a possibility of conflict for access to occur in an execution is a necessary intrinsic property of a protocol to achieve latency asymptotically better than $wn$.

**Theorem 11** For any conflict free protocol for $n$ stations, there is an execution in which a packet is delayed by $\Omega(nw)$ rounds against a window adversary with $w = \Theta(n)$ and an aggregate injection rate less than 1.

**Proof:** Let $n = k + 3$ for a positive integer $k$. Let there be an arbitrary conflict-free protocol for $n$ stations. We will consider an adversary with $w = n - 1$, and sometimes denote $w$ as $w = k + 2$. Let us declare one station to be heavy and the remaining ones to be mavericks. The share of the heavy station is $k$, and one of the mavericks has its share equal to 1. This...
means the aggregate injection rate is \((k + 1)/(k + 2) < 1\). Let us partition any execution into disjoint segments of consecutive \((k + 1) \cdot (k + 2)\) rounds.

Consider an execution \(\mathcal{E}_1\) in which the adversary injects only into the heavy station with full capacity of \(k\) packets per \(k + 2 = w\) consecutive rounds. The adversary injects \(k(k + 2)\) packets into the heavy station during each segment. This leaves a room for \(k + 2\) ‘exploratory’ rounds during a segment of \(\mathcal{E}_1\), available to locate the maverick with a positive share. The idea is that a protocol cannot locate such a maverick without incurring a \(\Omega(n^2)\) delay. We specify an execution \(\mathcal{E}_2\) which has the same prefix as \(\mathcal{E}_1\) until the first injection into a maverick.

Suppose there is a segment \(S\) of \(\mathcal{E}_1\) in which less than \(k + 2\) mavericks are scheduled to transmit. Take an execution \(\mathcal{E}_2\) in which the adversary injects into a maverick that is not scheduled to transmit in \(S\) in the round just before \(S\) is to begin. This packet waits \((k + 1)(k + 2)\) rounds. Otherwise, suppose every maverick is scheduled to transmit at least once during every segment of \(\mathcal{E}_1\). If some maverick is scheduled to transmit more than once, then the number of packets in the heavy station increases during this segment. If only such segments are in \(\mathcal{E}_1\) then the protocol is not stable and packet delays are arbitrarily large. Otherwise there is a segment \(S\) during which each maverick is scheduled to transmit exactly once. Partition \(S\) into first and second halves, each of \((k + 1)(k + 2)/2\) rounds. If the last maverick is to transmit in the first half, then consider \(\mathcal{E}_2\) in which the adversary injects into this last maverick just after its scheduled transmission in \(S\). If the last maverick is to transmit in the second half, then consider \(\mathcal{E}_2\) in which the adversary injects into this last maverick just before \(S\) is to begin. In each of these two cases, the packet injected into the last maverick will wait for at least \((k + 1)(k + 2)/2\) rounds. \(\square\)

7.4 Two protocols of small latency

In this section we present protocols with the packet latency that is close to optimal. They are developed for the two scenarios when either (1) collision detection is available or (2) control bits can be sent in messages.

7.4.1 A full sensing protocol with collision detection
We develop a non-adaptive full-sensing protocol UPGRADE-COLLISION which uses collision detection to provide small latency. Protocol UPGRADE-COLLISION in turn uses two protocols BINARY-SEARCH-COLLISION and CYCLIC-UPDATE-COLLISION executed in sequence. Protocol BINARY-SEARCH-COLLISION is executed first and we switch to CYCLIC-UPDATE-COLLISION provided that sufficiently many collisions occur in an execution of the former.

A station \( i \) uses a list \( D_i \) of bits to implement its estimate of shares. Each such a list \( D_i \) is initially empty. A station \( i \) that upgrades its share appends a 1 to the end of its list \( D_i \), while at the same time all the other stations append 0’s to their lists. The lists \( D_i \) are used to structure transmissions as follows: the station whose current entry in \( D_i \) is 1 transmits, while the other stations pause.

**Protocol BINARY-SEARCH-COLLISION:** Now we explain in detail how protocol BINARY-SEARCH-COLLISION works. The protocol runs a main thread that uses the lists \( D_i \) with the pointers advanced cyclically. Any underestimated station becomes persistent in the sense that it keeps transmitting as long as it has packets with the only exception when being scheduled to transmit in the main thread. This continues as long as silence or a packet are heard. If a collision occurs then the main thread pauses for a duration of binary-search thread and resumes only after the binary-search thread terminates; in particular, the main pointers associated with the lists \( D_i \) are not advanced while the main thread pauses. The binary-search thread performs a search over the range of all the stations using intervals of names of stations, starting with the interval of \([1,n]\) comprising all the names. At each step only stations in the current interval that want to upgrade their shares transmit. If a collision is heard then the interval is partitioned and processed recursively using a stack. The left interval is pursued first while the right interval is pushed on the stack. If a silence is heard for the current interval then such an interval is abandoned and the next interval is popped from the stack and processed. If a packet is heard then a station that transmitted the packet withholds the channel and transmits a number of packets up to its share’s upgrade followed by silence. Such a station appends a 1 to its list \( D_i \) for each
transmission of a packet is this situation, while the other stations append 0’s each. The binary-search thread terminates after the stack becomes empty. The main thread resumes after this.

**Lemma 8**  
Packet latency of Binary-Search-Collision is at most $O(w \log n)$ when it is executed against an adversary of window $w$ in a system of $n$ stations.

**Proof:** Packet delay may be caused by either the burstiness or the time spent to upgrade the shares. The former contributes at most $w$ to the delay. The latter contributes at most $1 + \lg n$ per each upgrade, where $\lg n$ is the binary logarithm of $n$, and these void rounds for each share upgrade occur only once. As there are at most $w$ upgrades, the total packet delay is at most $w(2 + \lg n)$. □

**Protocol Cyclic-Update-Collision:** Now we explain in detail how protocol Cyclic-Update-Collision operates. It contains two threads: the main thread and the update one. Both the threads work simultaneously unless paused for upgrading shares. The main thread uses the lists $D$. A station $i$ that has a 1 as a current entry in its list $D_i$ in a round transmits if it has a packet. The update thread works by using a cyclic list of the names of all the stations with a main pointer associated with it. A station that is current on this list is referred to as current in the thread. Such a station transmits only if it finds itself underestimated and has a packet to do so. If such a packet in update thread is heard then the station turns persistent in the sense that it keeps transmitting a packet in every round. A station that is persistent turns back to non-persistent if its queue becomes empty or if there is a collision or if it becomes current again in the update thread. Notice that there is only one persistent station at any time. If a collision is heard then both the main and update threads pause to upgrade shares. Now the current station in the update thread is given a chance to transmit. If it wants to upgrade its share then it transmits a number of times up to its share’s upgrade followed by silence. This is performed together with appending 1’s to the list $D_i$ of the transmitting station and 0’s to such lists of the other stations, an
entry for each transmission of a packet. After a silent round any persistent station is given a chance to upgrade its share. If such a station exists then it upgrades up to its new share followed by a silent round. Again, this is done along with appending the respective binary digits to the lists $D$. Both the main thread and update one resume after two silent rounds.

**Lemma 9** Packet latency of CYCLIC-UPDATE-COLLISION is at most $q + 4w$ when the protocol is executed against an adversary of window $w$ in a system of $n$ stations with an initial queue of size $q$.

**Proof:** Packet delay may be attributed either to the number of packets inherited in the queue or to the burstiness or to time spent to upgrade the shares. The queue contributes $q$ and burstiness contributes at most $w$. An update costs at most three void rounds, as they comprise one collision and two silences. These void rounds for each share’s upgrade happen only once. As there are at most $w$ upgrades, the total packet delay is as claimed. □

**The ultimate protocol UPGRADE-COLLISION:** Protocol UPGRADE-COLLISION starts by invoking protocol BINARY-SEARCH-COLLISION. A count of the total number of collisions heard is maintained throughout an execution. If the total count of collisions reaches $n$ then protocol BINARY-SEARCH-COLLISION is switched off and protocol CYCLIC-UPDATE-COLLISION takes over.

**Theorem 12** Protocol UPGRADE-COLLISION provides $O(\min(n + w, w \log n))$ packet latency for a channel with collision detection against the adversary of window $w$ in a system of $n$ stations.

**Proof:** If protocol CYCLIC-UPDATE-COLLISION is not invoked, then packet latency is given by Lemma 8. When CYCLIC-UPDATE-COLLISION is invoked, then there are $O(n)$ packets in queues in the round of invocation. Then $O(n + w)$ becomes a bound given by Lemma 9, which is an upper bound on packet latency. □
7.4.2 An adaptive protocol without collision detection

Next we show how to simulate protocol UPGRADE-COLLISION to obtain an adaptive full-sensing protocol for channels in which collision detection is not available. We call the simulating protocol UPGRADE-SILENCE as silences trigger upgrades of shares. This protocol simulates the two protocols in UPGRADE-COLLISION by running two simulating protocols Binary-Search-SILENCE and Cyclic-Update-SILENCE, respectively. The simulation proceeds as follows.

Regarding the main thread, a station without packets scheduled to transmit transmits a control bit to indicate this, rather than pause and make the round silent. So a silence occurs only when the modified protocol causes collision. The binary-search thread in Binary-Search-COLLISION relies on collision detection but now collision is heard as silence. It is possible that the silence heard in round $t$ during the binary-search thread actually indicates no transmissions or collision. We resolve whether this is the case in the next $t+1$-st round as follows. All the stations that transmitted in round $t$ transmit together with station 1, when station 1 simply transmits a control bit. There are two possible events occurring in round $t+1$: either the round is silent or a message is heard. Silence indicates that more than one station transmitted, as station 1 certainly did transmit. This means that there was collision in round $t$. If a message is heard in round $t+1$ then this means that round $t$ was silent, as station 1 managed to have its transmission in round $t+1$ heard. The simulation of Cyclic-Update-SILENCE proceeds similarly as of the main thread in Binary-Search-COLLISION.

**Theorem 13** Protocol UPGRADE-SILENCE provides $O(\min(n+w,w\log n))$ packet latency for a channel without collision detection against an adversary of window $w$ in a system of $n$ stations.

**Proof:** The simulation we employ results in a constant overhead per each round that contributes to packet latency of the simulated protocol. Therefore packet latency is of the order of a bound for the simulated protocol, which is as in Theorem 12. □
7.5 A full sensing protocol without collision detection

In this section we consider the channel without collision detection. We develop a non-adaptive full-sensing protocol of bounded packet latency for injection rate 1. The protocol is called COLORFUL-STATIONS. The protocol uses a list of discovered stations, in a way similar as before, in that a newly discovered active station is immediately appended to the list, and there is a pointer associated with the list that is advanced in a cyclic manner.

An execution of the protocol is structured to begin with a stage we call preparation, which is followed by iterated phases. A *phase* consists of three stages organized as follows. A pure stage occurs first, it is followed by an update, and finally a makeup concludes the phase. It may happen that, in some phase, the update and makeup stages are missing, so that the initial pure stage of the phase comprises the remaining part of the execution: this may occur only when a packet is heard in every round starting from a certain point in time in this pure stage.

The following are the intuitions that have guided the design of stages. The purpose of the preparation stage is merely to discover at least one active station; it is because of this goal why this stage occurs only once. Pure stages are for transmissions by stations already discovered. The amount of time allotted for such transmissions correspond to the rate of each station, as estimated so far. It is during these stages that most of the work of broadcasting is expected to be accomplished. An update stage serves the purpose to give the stations an opportunity to announce that some among them are underestimated. Such announcements result in the respective entries of the array $C$ getting incremented in order to improve the estimates of the shares. Stations that are not underestimated pause during these rounds of the update, so these rounds could be considered wasted for these stations. Eventually no station could be underestimated and so updates, if any, would consist of silent rounds only. This creates a potential for the queues to grow unbounded at stations that keep obtaining their maximum load of packets. To prevent instability, we do not rush into an update. We wait patiently until there are $n$ silent rounds accrued during a pure stage. Each such a silent round indicates that the corresponding station has no
packets. When $n$ silent rounds have occurred, the stations are conceptually partitioned into those that have been detected to have no packets during the pure stage and the remaining ‘busy’ ones that have not. The purpose of a makeup stage is to allow the busy stations to compensate for the rounds wasted during the forced silent rounds of the preceding update and so keep queues suitably bounded.

During a pure stage, a token is generated each time a silent round occurs. Creating and manipulating tokens is a conceptual method to assign extra rounds for transmissions to potentially heavily loaded stations during the following makeup. The problem we need to handle is that the adversary may start neglecting some stations while injecting at full capacity at the remaining ones. A busy station $i$ that is being injected at with the average of $C[i] = s_i$ packets during $\gamma$ consecutive rounds needs $C[i]$ rounds to transmit over a contiguous segment of $\gamma$ rounds on the average, which is provided by the design of a pure stage. Here $\gamma$ denotes the current estimate of the aggregate burstiness, while $\delta$ is the true aggregate burstiness as determined by the adversary’s type. Alas, silences incurred by the stations neglected by the adversary keep triggering updates. These stages are there because we do not know if the silences mean $\delta < w$, and so the injection rate is less than 1, or rather that the estimates of the shares in array $C$ are still underestimates while the injection rate is 1. Each such a possible update of $n$ rounds is spent unproductively in the case when no station is actually underestimated. In particular, no station is ever underestimated once the entries of the array $C$ have their final steady values in the execution.

Tokens come with two colors: green and red. Red tokens are used to mark the potentially heavily loaded stations to allow them next to make up for the rounds lost on silences. Green tokens are to indicate that their holders have missed a transmission, which indicates that such stations have a light load. Next we describe the four kinds of stages in detail.

The preparation stage is organized such that every station has one round to transmit at a time, assigned in a round robin manner. A station with a packet available transmits one when the station’s turn comes up, otherwise the station pauses during its time slot. The preparation terminates when the station that transmitted first is scheduled to transmit
again. Each station \( i \) whose packets have been heard during the preparation becomes discovered, which is recorded by setting \( C[i] \leftarrow 1 \).

During a pure stage, the discovered stations proceed through a sequence of transmissions, starting from the current station. A station \( i \) has a segment of consecutive \( C[i] > 0 \) rounds allotted for exclusive transmissions. During this segment of rounds, the station \( i \) keeps transmitting as long as it has packets, otherwise the station \( i \) pauses. The pointer is advanced, and the next station takes over, when either the current station \( i \) has used up the whole segment of \( C[i] \) assigned rounds or just after station \( i \) did not transmit while scheduled to.

Next we explain how tokens are created and assigned to discovered stations during a pure stage. Every station keeps a list of the tokens and their assignments in its private memory and performs operations on tokens in exactly the same way as other stations. All operations on tokens are triggered by silences. When a station \( i \) holds a token, then the color green of the token indicates that the token was generated when \( i \) was silent during a round in a segment of \( C[i] \) rounds allotted for \( i \) to transmit. The red color of a token held by station \( i \) indicates that it was some other station \( j \), for \( i \neq j \), that was silent during a round in a segment of \( C[j] \) rounds allotted to \( j \) to transmit and which generated the token.

A discovered station may hold either no tokens or a green one or a red one in a round. No station holds a token in the beginning of a pure stage. Let a station \( i \) be silent in a round assigned to \( i \) to transmit, which generates a token. If \( i \) does not hold a token yet, then the new token is colored green and it is assigned to \( i \). If \( i \) already holds a green token, then the new token is colored red and it is assigned to the first available discovered station that does not hold a token yet, in the order of their names. If \( i \) holds a red token, then the new token is colored green, it is assigned to \( i \), while the original red token held by \( i \) is reassigned to a discovered station that does not hold a token, if any. A discovered station is considered colored by the same color as the token it holds when an update begins. We need tokens only to assign colors, so when every discovered station gains a token and hence a color, then we refer only to the colors, which remain assigned to the stations for the duration of
the phase. After every discovered station has gained a color, a pure stage terminates.

An update stage is structured to give every station one opportunity to transmit exclusively for a contiguous segment of rounds, including candidate stations. A station $i$ that is underestimated by the amount $x$ transmits $x$ times which is followed by silence. This results in an immediate increment $C[i] \leftarrow C[i] + x$ at all stations. In particular, when a new station $k$ becomes discovered, then this results in setting $C[k]$ to a positive value. When a station simply pauses in the first round assigned to it, then the corresponding entry in the array $C$ is not modified; for instance, a candidate station maintains its status. It might happen that an underestimated station does not have sufficiently many packets ready to announce by how much it is underestimated in an update stage, which is not an issue as this indicates that so far the station has had enough room to handle its packets.

A makeup stage follows next. The red stations spend this stage working to unload their packets while the green stations pause. Red stations transmit in the cyclic order inherited from the list, starting from the current station if it is red or otherwise the next red one following the current station on the list. A red station $i$ has a segment of consecutive $C[i]$ rounds allotted for exclusive transmissions. A silent round by a red station results in changing the color of the station to green immediately and advancing the pointer to the next red station, if any. A makeup stage concludes as soon as there are no red stations anymore. To control the duration of a makeup stage, we impose additional restrictions. Consider the beginning of a makeup stage: at this point all the discovered stations are colored. Let $G$ and $R$ denote the sets of green and red stations, respectively. Let $g$ be the sum $\sum_{i \in G} C[i] = g$ of the entries of the array $C$ over the green stations, and $r$ the similar sum $\sum_{i \in R} C[i] = r$ of the entries of the array $C$ over the red stations. We have $g + r = \gamma$. We additionally require that makeup stage terminates after $3nr/g$ rounds.

This concludes the specification of all the possible kinds of stages, and hence of the protocol.

When a packet is heard in a round, then the packet is dequeued while simultaneously at most one packet on the average could be inserted. It follows that such rounds contribute
only to fluctuations of the queues by $O(w)$ packets, as $w$ is an upper bound on the burstiness, so we may restrict our attention to packets injected during silent rounds only when evaluating the size of queues.

At most $n$ silent rounds occur during the last $n$ rounds of the preparation stage and during every following stage. For accounting purposes, we partition silent rounds into blocks defined as follows. The first block comprises the silences incurred during the last $n$ rounds of the preparation, during the first pure stage and the first update stage. The next block consists of at most $3n$ silences incurred during the immediately following makeup, pure and update stages. This continues throughout the execution, a block comprising silences in consecutive makeup, pure and update stages. The idea is to show that a makeup stage takes care of the preceding block of silent rounds, in terms of compensating some stations for the time lost by not being allowed to transmit.

**Lemma 10** A makeup stage in protocol COLORFUL-STATIONS takes $O(nw)$ rounds.

**Proof:** The makeup stage is to neutralize the increase of the number of packets parked during the preceding block of silent rounds in red stations. Simultaneously it takes care of all packets newly injected into red stations with a conceptual rate being at most the rate of the preceding block. We need to estimate the time by which all red stations go green. By Lemma 6, the sets $G$ of green stations and $R$ of red stations have had packets injected into them with the cumulative rates of at most $g/\gamma$ and $r/\gamma$, respectively, during the current phase.

Let us call *old* the packets injected into red stations during all silent update rounds of the preceding block. There are less than $3n \cdot \frac{r}{\gamma}$ old packets in the red stations when a makeup stage begins. Let us call *new* the packets injected into red stations during makeup rounds. When the red stations are busy transmitting during makeup rounds, these rounds can be partitioned into disjoint segments of $\gamma$ rounds each. We employ the following accounting method. During such a segment of $\gamma$ rounds, the first $r$ rounds may be considered as devoted to unloading new packets injected into red stations during this segment, while
the remaining \( g \) rounds could be treated as accounting for unloading the old packets injected into red stations during the preceding block of silent rounds.

There are less than \( 3n \cdot \frac{\gamma}{g} \) old packets. Handling them by our accounting method requires \( 3n \cdot \frac{\gamma}{g} \cdot \frac{1}{g} \) segments of length \( g \) each. Every such a segment is taken out from a contiguous segment of \( g \) makeup rounds. This is possible when at least

\[
3n \cdot \frac{r}{\gamma} \cdot \frac{1}{g} \cdot g = 3n \cdot \frac{r}{g} \tag{7.1}
\]

makeup rounds are partitioned into such segments of length \( \gamma \) each. This bound is a part of code of the makeup stage. We obtain that bound (7.1) is at most \( 3nw \), as \( r \leq \gamma \leq w \) and \( g \geq 1 \), so the bound is \( O(nw) \). \( \square \)

**Theorem 14** Queues are \( O(n+w) \) and packet latency is \( O(nw) \) when protocol COLORFUL-STATIONS is executed against adversaries of window \( w \) in systems of \( n \) stations.

**Proof:** A packet injected during a phase is heard by the end of the next phase. This observation does not lead to a bound on packet delay, as there is no general upper bound on the duration of a pure stage, but it can be used directly to estimate the total size of the queues.

For each station, its queue becomes empty at some point during each phase. This is because contributing a silence converts the station green. When all stations become such then a phase is over. There are less than \( 3n \) silent rounds during a phase. The fluctuation of the size of the queues due to these rounds is \( O(n+w) \). This means that queues are \( O(n+w) \).

To estimate latency, let us examine each kind of stages. Take pure stages first. The inequality \( \gamma \leq w \) holds by Lemma 6. We consider two cases. When \( \gamma < w \), then a pure stage takes \( O(n+w) \) rounds. Otherwise, when \( \gamma = w \), then there is no general upper bound on the duration of a pure stage. Observe however that a packet is delayed by \( O(n+w) \) rounds during such a stage, as \( O(n+w) \) is an upper bound on the number of packets parked while the throughput is equal to the input rate. An update stage takes \( O(n+w) \) rounds by
its design. A delay due to the makeup stage is $O(nw)$ by Lemma 10. The latter bound determines the latency by being the biggest among the contributions to packet delays by all kinds of stages.

Next we consider the question if the bound in Theorem 14 is tight. Observe that the protocol is conflict free, so it follows from Theorem 11 that packet latency has to be $\Omega(nw)$ for suitable adversaries.

### 7.6 Conclusion

We introduced a model of adversarial queuing on multiple access channels in which individual injection rates are associated with stations. We developed a number of protocols, which have the following two properties. First, the partitioning of the aggregate rate among the stations constrains the adversary but is unknown to the stations. Second, the bounds on queue size and packet latency of our protocols are not expressed in terms of the distribution of the aggregate injection rate among the stations as their individual rates, but are given only in terms of the number $n$ of stations and the burstiness, which equals the window $w$ for the aggregate rate 1.

The purpose of this work was to compare the adversaries determined by individual injection rates with the adversaries constrained only by global injection rates. The comparison of the adversarial models was to be in terms of the attainable quality of service for the maximum throughput of 1. The main discovered difference between the globally-restrained and individually-restrained adversaries is that bounded packet latency is achievable when a separate injection rate is associated with each station by an adversary, even by non-adaptive full sensing protocols that do not know anything about the adversary, which is impossible for the globally-restrained adversaries.

We developed protocols for a window-type adversary with packet latency close to asymptotically optimal in the following two cases. One is when the protocols are adaptive and channels are without collision detection. Another is when protocols are non-adaptive full sensing but channels are with collision detection.
Packet latency of non-adaptive full sensing protocols for channels without collision
detection turned out to be more challenging to restrict. The protocol we developed
achieves $O(nw)$ bound on packet latency. This protocol avoids conflicts for access to
the channel. As we showed, packet latency has to be $\Omega(nw)$ for such conflict-avoiding
protocols. This means that the developed protocol is best possible in this class in terms of
asymptotic packet latency.

The question if a non-adaptive full sensing protocol can achieve packet latency that
is asymptotically less than $nw$ for channels without collision detection for window-type
adversaries of individual injection rates remains open.

The adversarial model considered in this work is of the window type. It is a natural
question how to extend it to the general leaky-bucket case, and how would such an adver-
sarial model affect protocols’ performance. In the case of globally-constrained adversaries
with injection rate 1, it was shown in [30] that the two models of window and leaky-bucket
adversaries make a difference for suitably small systems.
8. Experiments

8.1 Simulations without jamming

Some of the results given in this were presented in a preliminary form as conference papers Anantharamu et al. [11, 12].

![Figure 8.1](chart.png)

**Figure 8.1:** Observed values of maximum packet latency for varying injection rates. Parameter values: $R = 100,000$; $n = 16$; $b = 10$; $\alpha = 0.5$; and $\beta = 0.01$. The vertical scale is logarithmic.

We simulated broadcast protocols on a multiple access channel with packet injection patterns related to leaky bucket adversaries. The following are the main parameters that define a setting of a simulation: a number $n$ of stations, an adversary of some type $(\rho, b)$, and a specific broadcast protocol $\mathcal{P}$. The problem with simulating an adversary is that adversarial models are used to capture worst-case performance of protocols, which is challenging to model via simulations as using randomness in simulations is natural but it has an averaging effect. We worked with injection patterns defined by the same type $(\rho, b)$ as a given adversary, but the patterns being merely consistent with the constraints of this type. Ideally they would represent the worst-case behavior but what they produce is affected by random choices. We attempt to model phenomena that originally prompted the use of randomness to arbitrate for access to the channel, as employed in broadcast protocols like
the Ethernet. Deterministic protocols are of interest to us and we want deterministic protocols to compete against randomized ones; to this end we need a playground that is fair. So we want to model a scenario when there are always a number of stations into which the adversary is injecting packets so they are busy transmitting their packets while other stations do not receive packets temporarily so they stay idle. Such a status of a station as being busy or idle keeps changing. One could argue that this captures the real-world applications and protocols relying on randomness would fare well in such scenarios. To represent such situations in simulations, we provided two independent mechanisms. One controlled the set of stations into which packets were injected and the other governed the number of packets generated in a round.

![Graph](image)

**Figure 8.2:** Observed values of maximum packet latency for varying injection rates. Parameter values: $R = 100,000; n = 16; b = 10; \alpha = 0.5; and \beta = 0.1$. The vertical scale is logarithmic.

### 8.1.1 Machinery of packet injection:

We want to have a behavior that is more bursty than random, and to facilitate it we introduce three additional parameters apart from burstiness and injection rate, denoted by the symbols $\alpha, \beta, \gamma$. We begin with a model that captures the change of status of stations as either busy or idle. Each station goes through periods of activity interleaved with periods
of passivity. There are two parameters: the activity denoted by $\alpha$ and the volatility denoted by $\beta$. These numbers satisfy the inequalities $0 < \alpha \leq 1/2$ and $0 \leq \beta \leq 1$. The motivation was that there are $\alpha n$ stations active in receiving packets while the number $\beta$ determined the propensity to change status. The active stations were eligible to receive packets while the remaining passive stations were transmitting only their old packets. The volatility is to be understood as a rate of change of status of stations from active to passive, in the sense that it is a fraction of the set of all active stations per round that become passive.

The implementation of activity and volatility are as follows. Let $k = \alpha n$ be the number of active stations; initially we pick some $k$ stations as active. What we do depends on whether $\beta k$ is at least 1 or not. If $\beta k \geq 1$, then at each round we choose randomly $\lceil \beta k \rceil$ active stations and $\lceil \beta k \rceil$ passive stations: these stations change their status from active to passive and vice versa. If $\beta k < 1$ then we partition an execution into time segments of length $\lceil 1/(\beta k) \rceil$. At the beginning of such a segment we choose a random active station and a random passive station and let them switch their status. Additionally, the numbers of packets generated at consecutive rounds is consistent with the given adversarial type $(\rho, b)$.

We maintain a variable $m$ called the potential which is to denote an upper bound on the number of packets that can be injected in a round; $m$ is a rational number rather than an integer. This potential $m$ is initialized to $m \leftarrow b$ before the first round. After a round in which $x$ packets were generated and injected we modify the potential $m$ as follows: $m \leftarrow \min(m - x + \rho, b)$, which is consistent with type $(\rho, b)$ of an adversary. The specific number of packets generated during a round is determined by parameter $\gamma$. The parameter $\gamma$ is set to be equal to the injection rate $\rho$. Consider a round and let $m$ be the current value of the potential. We first choose an integer $j$ with a Poisson probability distribution with expected value $\gamma$. If the value of the integer $j$ is in the segment $[0, m]$ then the adversary generates $j$ packets otherwise the adversary generates no packets in this round.
Figure 8.3: Observed values of maximum packet latency for varying injection rate. Parameter values: $R = 100,000; n = 16; b = 10; \alpha = 0.5; \text{ and } \beta = 0.001$. The vertical scale is logarithmic.

Experiments proper: The goal of the simulation was to look into the worst case behavior of broadcast protocols for various parameters. The parameters of an experiment included the following. The number of simulation rounds $R$, the number of stations $n$, injection rate $\rho$, burstiness $b$, and $\alpha$ and $\beta$. Only the packets injected in the first $R$ rounds were used to measure maximum latency. To that end, executions of protocols were continued until all the packets injected within the rounds up to $R$ had been transmitted. We simulated the five deterministic protocols along with the binary exponential backoff and the quadratic backoff. The backoff protocols were implemented as the windowed versions where the round of a transmission is drawn from a window uniformly at random. For exponential backoff (EB), the $i$th window size was determined as $2^{\min(10,i)}$, for $0 \leq i \leq 10$. For quadratic polynomial backoff (PB), the $i$th window size was defined to be $(\min(i,32))^2$, for $1 \leq i \leq 32$. The constant $c$ was chosen to be 10 for binary exponential backoff similar to what is used in the Ethernet. The maximum size of a window was the same in PB and EB. Note that none of the packets were dropped for both deterministic and backoff protocols.
Experiments were run under varying injection rates and numbers of stations; their results are depicted in Figures 8.1 through 8.4. In Figure 8.1 we compare the performance of deterministic and backoff protocols: it can be seen how the deterministic protocols outperform the backoff ones as the injection rates grow. Amongst the deterministic protocols MBTF fared worse. This is what Corollary 4 indicates. Another thing to notice in Figure 1 is the performance of quadratic and exponential backoff protocols. For very high injection rates we were expecting the quadratic backoff to outperform the exponential backoff but this was not achieved in our experiments. This needs to be further investigated. In Figure 8.2 we compare the performance of deterministic protocols. The regular protocols out performed the old-first ones and MBTF had the worse performance.

Figure 8.4 shows the effect of the number of stations on the maximum packet latency for some deterministic and backoff protocols: deterministic protocols again outperformed the backoff ones.

We also looked into the effect of the old-first paradigm to see if it makes a difference in experiments. In Figure 8.3 we compare regular protocols with old-first ones. The
performance of OFSRR and OFRRW appears to be different from what we have in Theorem 6 and Corollary 1. Regular protocols out performed the old-first ones. So what we can see is that when there is some randomization involved in experiments the old first protocols do not necessarily out perform regular protocols. This experimental result with some randomization in packet injection is in contrast to the theoretical results with no randomization.

8.2 Simulations with jamming

We performed experiments to compare packet latency of deterministic broadcast protocols among themselves and also with randomized backoff protocols in an adversarial setting with jamming.

We build on the mechanism proposed in [11] and extend it to incorporate jamming. In an attempt to capture the worst case behavior and burstiness of broadcast demands, where typically some stations are actively receiving packets while others stay idle, two parameters $\alpha$ and $\beta$ defined as before were used.

An adversary controls both injection and jamming and injects only in active stations in a round. The type of an adversary determines injections by the parameters $(\rho, b)$ and jamming by the parameters $(\lambda, b)$; the rates $\rho$ and $\lambda$ are rational numbers in experiments. The freedom in how many packets to possibly inject in a round or if to jam a round depends on the history of the adversary’s activity. We represent such history by potential $P$ understood to mean either an upper bound on the number of packets possible to inject in a given round or the number of consecutive rounds that could be jammed starting from the current round, depending on which aspect of adversary’s activity is to be decided upon, respectively. We maintain two potentials, packet-injection potential $P_i$ and jamming potential $P_j$ to represent the maximum number of packets that could be injected in the current round and the consecutive number of packets that could be jammed, respectively. More precisely, the number of packets generated to be injected is taken from the interval $[0, P_i]$ and the number of consecutive rounds to jam starting from the current round is taken from the interval $[0, P_j]$. The potentials are initialized to $P_i \leftarrow \rho + b$ and $P_j \leftarrow \lambda + b$. After
a round in which \( x \) packets were generated and injected we modify the packet-injection potential by 

\[ P_i \leftarrow \min(P_i - x + \rho, b) \]

which is consistent with the \((\rho, b)\) component of the adversary’s type. After a round in which \( y \) consecutive rounds were decided to be jammed we modify the jamming potential by 

\[ P_j \leftarrow \min(P_j - y + \lambda, b) \]

which is consistent with the \((\lambda, b)\) component of the adversary’s type. In each round of a block of rounds to be jammed as decided in advance, the numbers \( y \) are interpreted to mean 0 until a decision about jamming needs to be taken again. Given a potential \( P \), the decision to select a number in the interval \([0, P]\) for a round is determined by an additional parameter \( \gamma \); this is implemented as follows. Parameter \( \gamma \) is set to be equal to injection rate \( \rho \) for packet injection and to jamming rate \( \lambda \) for jamming. First choose an integer \( k \) with a Poisson probability distribution with mean \( \gamma \). If an integer \( k \) happens to be in the segment \([0, P]\) then \( k \) is the selected number, otherwise it is 0 that is selected. Stations attached to the channel receive two possible kinds of feedback in a round: if a round is jammed then each station perceives the round as silent, otherwise what is heard depends on the actions of the stations in terms of packet transmissions, as it is specified for a multiple-access channel without collision detection.

We implemented the five deterministic protocols specified in Sections 5.1.1 and 5.1.2; these are two full-sensing protocols JRRW(J) and OF-JRRW(J) and three adaptive protocols C-RRW, OF-C-RRW, and MBTF. For the two full sensing protocols the jamming burstiness \( J \) was set to 

\[ J = \min\left(\left\lfloor \frac{b}{(1 - \lambda)} \right\rfloor + 1, 3\right) \]

Note that while the jamming rate changes in the experiments, the jamming burstiness also changes accordingly. In addition to deterministic protocols, we implemented two back-off protocols: the exponential back-off, denoted EXPBACKOFF, and the quadratic polynomial back-off, denoted POLYBACKOFF. The implementation for back off protocols was exactly similar to the ones used in non jamming experiments.

Simulations were run to see how injection rates, jamming rates effected maximum latency, respectively. The numbers of rounds for injections in simulations were fixed in advance. The maximum latency of packets injected up to the maximum injection round
was measured by continuing the simulation until all the packets injected up to the fixed final injection round had been transmitted. We ran experiments to compare maximum latency of deterministic protocols with back-off protocols.

Maximum latency of deterministic protocols were compared to back-off protocols as shown in Figure 8.5, for varying injection and jamming rates. Three deterministic protocols MBTF, C-RRW, and JRRRW(J) were compared among themselves for maximum latency for varying injection and jamming rates; these are the protocols considered that do not have the old-first mechanism. The outcomes of these experiments are shown in Figures 8.6. The outcomes can be interpreted as follows.

Figure 8.5 contains a chart that presents a comparison of all the seven protocols, including the back-off ones. A logarithmic scale is used for packet latency. The horizontal axis of rates in a figure represents the sum $\rho + \lambda$, which is considered with increments 0.05. The vertical axis represents the maximum packet latency recorded for the corresponding values of the sum $\rho + \lambda$. The injection rate $\rho$ is set to $10 \times \lambda$. Amongst the deterministic protocols, the order of best performance for maximum packet latency was C-RRW, OF-
Figure 8.6: Observed values of maximum packet latency for varying injection and jamming rates. Parameter values: $R = 100,000; n = 16; \rho = \lambda; b = 10; \alpha = 0.5; \text{ and } \beta = 0.001$. The vertical scale is linear.

C-RRW, MBTF, JRRW($J$) and OF-JRRW($J$). Deterministic protocols outperformed the two back-off protocols for higher injection and jamming rates, although the back-off protocols did better than some deterministic protocols for smaller injection and jamming rates. We can observe a jump in packet latency of back-off protocols, by two or three orders of magnitude, around the combined rates of 0.4. Again regular protocols performed better than the Old-first protocols, which is in contrast with the corresponding theorems of Sections 5.1.1 and 5.1.2. The adaptive regular and old first protocols, JRRW($J$) and OF-JRRW($J$), were very close in terms of the maximum packet latency, with the regular protocol performing slightly better. For the full sensing protocols, regular protocol C-RRW outperformed old-first protocol. This results of regular and old first protocols indicate that randomization in experiments make a difference.

Figure 8.6 is a comparison of all the regular protocols. The vertical scale used is linear. The order of performance is C-RRW, MBTF and JRRW($J$).

8.3 Conclusion

Deterministic protocols outperformed the backoff ones in the experiments for higher injection and jamming rates, although the backoff protocols performed better than at least
some deterministic ones for smaller injection and jamming rates. Adaptive protocols typically fared better than full sensing ones. Adaptive protocol MBTF typically had greatest packet latency among adaptive deterministic protocols. Between the regular and old-first protocols the regular protocols performed better than the old-first ones. This is in contrast with theoretical results.
9. Adhoc multiple access channels

Multiple access channels model shared-medium networks in which simultaneous broadcast to all users is provided. They are an abstraction of the networking technology of popular implementation of local area networks by the Ethernet suite of algorithms [56].

In a multiple access channel, transmissions by multiple users that overlap in time result in interference so that none can be successfully received. This makes it necessary either to avoid conflict for access to the channel altogether or to have a mechanism to resolve conflict when it occurs.

We consider broadcasting in multiple-access channels in a dynamic scenario when there are many stations attached to the channel, but only a few of them are active at any time and the stations’ status of active versus passive keeps changing. This corresponds to a realistic situation when most stations are idle for most of the time while a few stations occasionally want to use the broadcast functionality of the channel. Moreover, it is normally impossible to predict in advance which stations will need to access the channel at what times, as bursts of activity among stations do not exhibit any regular patterns. The unpredictability of access to the channel among the stations can be handled by using randomization in resolving conflict for access, as implemented in the carrier-sense multiple access, see [46].

Considering deterministic algorithms and their worst-case performance requires methodological setting specifying worst-case bounds on how much traffic a network would need to handle. This can be accomplished formally through suitable adversarial models of demands on network traffic.

Another component in a specification of a broadcast system is how much of what the communicating agents know can be used in codes of algorithms. There are various approaches to model multiple access channels in terms of what is known to the stations attached to the communication medium. Historically, the first approach was to use the queue-free model, in which each injected packet is treated as if handled by an independent station without any name and no private memory for a queue. In this model, the number
of stations is not set in any way, as stations come and go similarly as packets do; see [35] for the initial work on this model, and [20] for more recent one.

An alternative approach to ad hoc channels is to have a system with a fixed number $n$ of stations, each equipped with a private memory to store packets in a queue. An attractive feature of such fixed-size systems is that even simple randomized protocols like Aloha are stable under suitable traffic load [65], while in the queue-free model the binary exponential backoff is unstable for any arrival rate [4].

The popular assumptions used in the literature addressing distributed deterministic broadcasting stipulate that there are some $n$ stations attached to a channel and that each station is identified by its name in the interval $[0,n-1]$, with each station knowing the number $n$ and its own name; see [11, 12, 13, 30, 31]. Our goal is to explore deterministic broadcasting on multiple-access channels when there are many stations attached to a channel but only a few stations use it at a time. In such a situation, using names permanently assigned to stations by deterministic distributed algorithms may create an unnecessarily large overhead measured as packet latency and queue size. This is because the packet latency is expected to be a function of the total number of stations, which is assumed to be very large; see [11, 12] for such approaches.

Here we consider distributed deterministic broadcasting but we depart from the assumption about a fixed known size of the system. Instead, we view the system as consisting of a very large set of stations that are not individually identified in any way. As some stations need to use the channel to communicate, they join the broadcasting activity, which needs to be coordinated with the other currently active stations by the medium-access control layer.

The process of activating stations is modeled by a suitable adversarial model that we propose. This adversarial model is designed to represent a flexible system in which we relax the assumption that there is a finite fixed set of stations attached to the channel, and that their number is known to each participating station, and that each station has assigned unique name which it knows. We call such channels *ad hoc* to emphasize the volatility of
the system and the relative lack of knowledge of individual stations about themselves and
the environment. Ad hoc channels are a crossover between the queue-free model, with
which they share the property of an unbounded supply of anonymous stations activated
by injected packets, and the model of finitely many stations in a system, with which they
share the property that stations use their private memories to implement queues to store
pending packets.

We measure the performance of broadcast algorithms by packet latency and queue
sizes. These metrics reflect the suitable properties of the adversarial model that define
constraints on packet injection. Such constraints include packet injection rate, understood
as the average number of packets injected in a large time interval, and burstiness, which
means the maximum number of packets that can be injected simultaneously. Adversarial
models of traffic allow to study the worst-case performance of deterministic communica-
tion algorithms in the suitable metrics.

Our results. We propose an adversarial model of traffic demands for ad hoc multiple
access channels, which represents dynamic environments in which stations freely join and
leave broadcasting activity. As anonymous systems cannot break symmetry in a deter-
ministic manner, we restrict adversaries by allowing them to activate at most one station
per round. This is shown to be sufficient for efficient deterministic distributed broad-
cast algorithms to exist. We categorize algorithms into acknowledgment based, activation
based and full sensing. Independently from that, we differentiate algorithms by the prop-
erty if they use control bits in messages or not, calling them adaptive and non-adaptive,
respectively. We give a number of algorithms, for channels with and without collision
detection, for which we asses injection rates they can handle with bounded packet latency.
More specifically, our non-adaptive activation-based algorithm can handle injections rates
smaller than $\frac{1}{3}$ on channels with collision detection, the adaptive activation-based algo-
rithm can handle injection rate $\frac{1}{2}$ on channels without collision detection, the non-adaptive
full-sensing algorithm can handle injection rates less than $\frac{2}{3}$ on channels with collision
detection, and the adaptive full-sensing algorithm can handle injection rate $\frac{3}{4}$ on channels
with collision detection. We also show that the latter algorithm is optimal, in terms of injection rate $\frac{3}{4}$ that can be handled with bounded packet latency, as we prove that no algorithm can provide bounded packet latency when injection rates are greater than $\frac{3}{4}$.

**Related work.** Adversarial queuing methodology was introduced by Andrews et al. [14] and Borodin et al. [26], where it was used for store-and-forward routing in wired networks. Adversarial queuing on multiple access channels was first studied by Bender et al. [20], in the case of the queue-free model and randomized protocols. Deterministic distributed broadcasting on multiple access channels with queues in adversarial settings was investigated first by Chlebus et al. [30, 31] and later studied by Anantharamu et al. [11, 12, 13]. That work on deterministic distributed protocols was about systems with a known fixed number of stations attached to the channel and with stations using individual names. Deterministic protocols for collision resolution in static algorithmic problems on multiple access channels were first considered by Greenberg and Winograd [40] and Komlós and Greenberg [47]. Acknowledgment-based protocols include the first randomized protocols studied on dynamic channels, as Aloha and binary exponential backoff fall into this category. The throughput of multiple access channels, understood as the maximum injection rate with Poisson traffic that can be handled by a randomized protocol and make the system stable (ergodic), has been intensively studied in the literature. It was shown to be as low as 0.568 by Tsybakov and Likhanov [64]. Goldberg et al. [38] gave such bounds for backoff, acknowledgment-based and full-sensing protocols. Håstad et al. [42] compared polynomial and exponential backoff protocols with respect to bounds on their capacity in the queuing model. For early work on full-sensing algorithms in channels with collision detection in the queue-free model see the survey by Gallager [35]. Randomized protocols of bounded packet latency were given by Raghavan and Upfal [59] in the queuing model and by Goldberg et al. [39] in the queue-free model. Upper bounds on packet latency in adversarial networks was studied by Anantharamu et al. [11, 12] in the case of multiple access channels with injection rate less than 1 and Rosén and Tsirkin [61] for general net-
works and adversaries of rate 1. Algorithmic problems of distributed-computing flavor in systems in which multiple access channels provide the underlying communication infrastructure were considered by Bieńkowski et al. [25], Chlebus et al. [28, 29], and Czyżowicz et al. [32].

9.1 Technical preliminaries

A multiple-access channel is a model of communication for local area networks. It consists of a shared communication medium and stations attached to it. These general properties can be made specific in various ways, our specifications are summarized next.

**Multiple access channels.** There is an unbounded supply of stations attached to the channel. Stations are *anonymous* in that they do not have individual names assigned to them. The contents a station broadcasts on the channel is called a *message*. Packets are injected into stations and the goal is to broadcast them on the channel. A message may include at most one packet and possibly a number of control bits attached to it. Each station has private memory, which is used to store private variables. A station stores pending packets in a private queue implemented in its private memory; such a queue is operated in a first-in-first-out manner. When a station transmits a message then the message reaches every station. A station receives the message successfully when the reception of this message does not overlap with any reception of other messages; in this instance we say that the message is *heard* on the channel. When a message is heard on the channel, then all stations receive it successfully, including the transmitting one. We consider synchronous channels which operate in rounds. Rounds and messages are calibrated such that transmitting one message takes the duration of a round. A message sent in a round is delivered to each station in the same round. When at least two messages are transmitted in the same round then this creates a *collision*, which prevents any station from hearing any of the transmitted messages. When no station transmits in a round, then the round is called *silent*. A channel is said to be *with collision detection* when the feedback from the channel in a collision round is different from the feedback received during a silent round, otherwise the channel is *without collision detection*. For a channel
without collision detection, a collision round and a silent one are perceived the same. A round is *void* when no station hears a message; such a round is either silent or a collision one. A station is said to be *active* if it has packets pending to be transmitted on the channel, otherwise it is *passive*. A station with an empty queue is said to be *activated* when at least one packet is injected into it. A station is initialized as passive. We consider channels that are *ad hoc* which means that stations are anonymous and there is no *a priori* bound on the number of active stations in a round. We impose restrictions on how passive stations may be activated.

**Adversarial model of packet injection.** Packets are injected by adversaries. Adversaries are constrained by how many stations they can activate in a round. An adversary is *k-constrained*, for an integer \( k > 0 \), if at most \( k \) stations may be activated in a round. We consider \( 1 \)-constrained adversaries, unless explicitly stated otherwise. The adversarial model of this paper is that of general leaky-bucket adversaries. For a number \( 0 < \rho \leq 1 \) and integer \( b > 0 \), a (leaky-bucket) *adversary of packet injection type* \((\rho, b)\) may inject at most \( \rho |\tau| + b \) packets in any time interval \( \tau \) of \( |\tau| \) rounds. In this context, the number \( \rho \) is the *rate of injection*. The maximum number of packets that an adversary may inject in one round is the *burstiness* of this adversary. The burstiness is \( \lfloor \rho + b \rfloor \) for the adversary of type \((\rho, b)\).

**Broadcast protocols.** We consider deterministic distributed broadcast protocols. Distributed protocols are event driven. An event in which a station participates consists of everything that happens to the station in a round, including what the station receives as feedback from the channel and how many packets are injected into it. The protocols we consider are organized such that pending packets at a station are stored in a queue in the private memory of the station and successfully broadcast packets are removed from their queues. This gives the property that a station is active if and only if its queue is nonempty.

Protocols do not “know” anything about the broadcast system in which they are
executed, where “knowledge” of properties of a system means using them as a part of code. This is in contrast with previous work on deterministic distribute protocols, see [11, 12, 13, 30, 31], where the names of stations and the number of stations were known to the stations. No information about adversaries is reflected in the code executed by stations. The state of a station is determined by the values of its private variables; this may or may not include the private queue used to store pending packets, depending on whether we want to have a station in the initial state while its queue is non-empty. Each station is initialized to the same initial state, by the same values stored in the private memory locations, and with an empty queue. Packets are treated as abstract tokens and their individual properties do not affect state transitions.

A round of an execution of a protocol is structured such that stations first perform their actions and next the adversary injects packets into some stations, if any. More precisely, a round consists of the following actions: first transmitting packets, then receiving feedback from the channel, next making state transitions, and finally having new packets injected; some of these actions may be void in a station in a round. A packet that was successfully transmitted by a station is removed from this station’s queue at the time of its state transition. A station becomes passive in the round in which it successfully broadcasts the only remaining packet in its queue, which makes the queue empty after dequeuing the packet. At this moment the station becomes passive, so when a packet is injected into this station in this very round, it is an injection into a passive station. A station’s status, of active versus passive, is dynamic in the course of an execution: an active station may either eventually be relegated to passive and stay such forever, or it may stay active forever, or it may change its status between active and passive any number of times.

Classes of protocols. We define subclasses of protocols as follows. General protocols may have stations attach control bits to packets to broadcast as messages. When such bits are used then we say that the protocol is adaptive, otherwise the protocol is non-adaptive. Protocols such that a station stays in the initial state while passive and resets itself to the
initial state in a round in which a packet it transmitted was heard on the channel are called 
acknowledgment based; for this to be meaningful, the private queues at stations are not 
considered as part of state, which is the only such situation we consider. Protocols such 
that a station stays in the initial state while passive and it resets itself to the initial state 
when it becomes passive again, that is, in a round in which the last packet from its queue 
is heard on the channel, are called activation based. Protocols such that stations may 
have state transitions in each round, whether a station is active or passive, effected by the 
feedback from the channel according to the state-transition rules represented by the code, 
are called full sensing. The categorization of adaptive versus non-adaptive is independent 
of the other three categorizations, so overall we have six categories of protocols. This 
categorization of protocols holds independently for channels with and without collision 
detection.

A station executing a full-sensing protocol may (in principle) remember the whole 
history of the feedback from the channel, unless the size of its private memory restricts 
it in this respect. An active station executing an activation-based protocol may remem-
ber the history of the feedback from the channel since the activation. An active station 
executing an acknowledgment-based protocol may remember the history of the feedback 
from the channel since the latest successful transmission or activation, whichever occurred 
later. Acknowledgment-based protocols are a subclass of activation-based protocols. Sim-
ilarly, activation-based protocols are a subclass of full sensing ones, as a station executing 
an activation-based protocol could be considered as receiving feedback from the chan-
nel but idling in the initial state when without pending packets. The terminology about 
acknowledgment-based and full-sensing protocols is consistent with that used in the lit-
erature on randomized protocols in the queue-free model, see [35], while it is different 
from the terminology used in the recent literature on deterministic distributed protocols in 
adversarial settings, see [11, 12, 13, 30, 31].
Quality of broadcasting. A protocol is *fair* when each packet injected into some station eventually gets heard on the channel. A protocol is *stable* when the queues stay bounded throughout any execution. A protocol has *packet latency* $t$ when each packet spends at most $t$ rounds in the queue before it is heard on the channel. These quality features normally hold depending on an adversary, that is, how $k$-constrained it is and what is its type.

9.2 Limitations on broadcasting

In this section we consider what limitations on deterministic distributed broadcasting are inherent in the properties of ad-hoc multiple access channels and the considered classes of algorithms.

**Proposition 1** No deterministic distributed algorithm is fair against a $2$-activated adversary of burstiness at least $2$.

**Proof:** We want to maintain an invariant that there are two stations that proceed through the same states in the course of an execution. Let an execution begin with the adversary injecting packets simultaneously into two passive stations, one packet per station. These two stations execute the same deterministic algorithm, so their actions are the same until one station experiences what the other does not. The adversary does not need to inject any other packets. It follows, by induction on round numbers, that when one of these two stations transmits a packet then the other one transmits as well, and when one station pauses then the other station pauses as well. In this execution, each transmission attempt results in a collision. This means that the two packets never get heard on the channel. □

In the light of Proposition 1, we restrict our attention to $1$-activated adversaries in what follows. For $1$-activated adversaries, we may refer to stations participating in an execution by the round numbers in which they are activated. So when we refer to *station* $v$, for an integer $v \geq 0$, we mean the station that got activated in the round $v$. If no station got activated in a round $v$, then a station bearing the number $v$ does not exist. To avoid multiple identities of a station, we assume that once a station is activated and later becomes passive, then it never gets activated again; this does not make a difference from the perspective of
the adversary, as we assume that there is an unbounded supply of passive stations.

**Proposition 2** No acknowledgment-based algorithm is fair against a 1-activated adversary of a burstiness that is at least three.

**Proof:** It is sufficient to demonstrate the existence of an execution in which two stations simultaneously start working to broadcast a new packet each. This is because the stations start from initial states, they are anonymous, and they execute the code of the same deterministic algorithm. We may assume that a passive station immediately attempts to transmit a packet when activated, as otherwise a delay results in a suitably earlier activation.

Let the adversary inject two packets into a passive station in round one and next one packet into another passive station in round two. The station 1 transmits its first packet successfully in the second round. The two stations 1 and 2 start working on a new packet each from the third round, which results in a collision and an execution whose existence we wanted to demonstrate.

For such a scenario to be possible, the adversary needs to be able to inject three packets in two consecutive rounds, which requires burstiness to be three when injection rate is less than \( \frac{1}{2} \).

Because of Proposition 2, we consider only activation-based and full-sensing algorithms in what follows. We consider the question what is the maximum injection rate for which bounded queues or packet latency can be attained. The answer may depend on the restrictions on the algorithms in a class, like the classes of activation based and general full-sensing algorithms, and whether the channel has collision detection or not.

Before embarking on optimizing adversaries in non-existence of bounded-latency algorithms, we may observe that no deterministic distributed algorithm can provide bounded packet latency against an adversary of injection rate equal to 1 and with burstiness at least 2. To see this, consider an arbitrary algorithm executed against such an adversary.

We build an execution by defining it through the prefixes of a sequence of auxiliary executions. Let an execution \( \mathcal{E}_1 \) be obtained by activating a station per each round, by
way of injecting one packet into a passive station. There are two cases. First, suppose
that there exists an active station \( v_1 \) which alone transmits a packet. The transmission
by \( v_1 \) is successful in \( E_1 \) and a packet is heard. Let us modify the execution \( E_1 \) to another
execution \( E_2 \) such that the station \( v_1 \) is not activated at all, which results in a silent round.
Instead, after the silent round in \( E_2 \), we activate a station by injecting two packets into it.
The target execution has its prefix determined until and including the injection of these two
packets simultaneously in \( E_2 \). In the second case, there exist two active stations \( v_2 \) and \( v_3 \)
such that they transmit together in \( E_1 \) in a round of the first transmission in this execution.
This creates a collision, which contributes to a packet delay. The target execution has its
prefix determined until and including this collision in \( E_1 \). Which of the two cases holds
depends on the algorithm considered. This construction is continued by building prefixes
of growing lengths. Each time we consider the execution with its prefix determining the
target execution, we next examine the suffix after this prefix for one of the two possible
cases as above. The obtained final execution has the property that there are infinitely many
void rounds in which no packet is heard, while simultaneously the adversary injects with
the rate of one packet per round on the average. This concludes the argument that the
injection rate 1 is too much to provide bounded packet latency. This observation can be
strengthened to smaller injection rates with a more involved argument, as we show next in
Theorem 15.

If an execution is to be of a bounded packet latency, an active station needs to have
sufficiently many opportunities to transmit its packets. In particular, if a certain round
is not checked for the possibility of a station being activated in this round and given an
opportunity to transmit at least one packet, then there exists an execution in which a station
is activated in this round indeed and its packets are never heard on the channel.

This is represented formally as follows, for an execution of a broadcast algorithm
on an ad hoc channel. We say that round \( v \) is verified in round \( s \) of the execution if
either the station that got activated in the round \( v \) transmits for the first time in round \( s \)
in the execution or no station got activated in the round \( v \) but such a station would have
transmitted for the first time in round $s$ if it were activated in the round $v$ in the execution. We use the phrase “station $v$ is verified” interchangeably with “round $v$ is verified” as we will consider only 1-activated adversaries so that at most one station gets activated in any round. Intuitively, a station gets verified in a round if it is the first round in which the station gets an opportunity to transmit, unless no station got activated in the round that identifies the station. We will always assume that if a station $v$ is verified in round $s$ then $s \geq v$, as otherwise there is no station to be verified.

We say that a verification of station $v$ gets completed in round $w$ when it is the first round in which the interaction of $v$ with the channel, or lack thereof, certifies that $v$ does not have pending packets. There are two ways in which such a certification could occur. One is when a station identified by the number $v$ is to become verified in the round $w$ and no station has been activated in round $v$ so $v$ does not transmit at all. Another way is when $v$ is still active in round $w$ and transmits its last packet in this round to immediately become passive.

If a station $v$ is verified in a round $s$ then the number $s - v$ is called the delay of verification of $v$. Observe that if an algorithm has packet latency at most $t$ in any execution against some adversary, then the rounds in any execution against this adversary are verified with the delay at most $t$. This is because if some round $r$ gets verified later than at the round $r + t$ in some execution $\mathcal{E}_1$ then consider an execution $\mathcal{E}_2$ in which a station is activated in round $r$ and its packet needs to wait beyond the round $r + t$ to be heard, which violates the bound on packet latency. One may argue about unbounded packet latency by specifying an execution in which delays of verification grow unbounded. This is how the next fact is proved.

**Theorem 15** No deterministic distributed algorithm can provide bounded packet latency against a 1-activated adversary of injection rate greater than $\frac{3}{4}$ and with burstiness at least 2.
Proof: Let us consider any deterministic distributed algorithm $A$ and a 1-activated adversary of a type $(\rho, b)$, where the injection rate $\rho$ satisfies $\frac{3}{4} < \rho \leq 1$ and $b \geq 2$. We determine executions of this algorithm by foreseeing the actions of the stations as directed by the algorithm $A$ and have the adversary act accordingly.

We make the following assumptions about algorithm $A$ to simplify the exposition of the arguments. We may assume that a successful transmission by a station is followed by other transmissions of this station, as long as the station has pending packets, until they are exhausted. We may assume, without loss of generality, that the stations are verified in the order of their activation, as this is most restrictive for the adversary. This means that no station $v$ is verified in a round $w$ when there exists a number $k < v$ such that the round $k$ has not been verified by the round $w$.

We build a specific execution $E$ by determining contiguous segments of up to four rounds, which we call portions. At any stage of the construction, the portions make a prefix of the execution $E$, initially it is the empty prefix. We consider a possible extension of a given prefix, and then specify what the adversary’s does. The adversary injects packets into a station only once at the time of its initialization.

Let $P$ be the already determined prefix of $E$. We consider a portion in $E$ that immediately follows $P$, which we denote by $S$. We will categorize these portions by the cases given below. The portions falling under one case are called similar. When such a portion $S$ has the properties that $S$ consists of four rounds and the adversary may cause exactly four stations to be verified in $S$ when allowed only to use the injection rate $\frac{3}{4}$ but if the injection rate $\rho$ is such that $\rho > \frac{3}{4}$ then the adversary can make infinitely many similar portions result in fewer than four stations verified, assuming that there will be infinitely many such similar portions, then such a portion is called critical. We consider only the stations scheduled to be verified in such a portion $S$, usually four of them. If stations with greater numbers than the four scheduled to be verified in $S$ want to be verified in $S$, then the adversary could activate the first of them. This would result in a collision, which can be shown by extending the argument along the lines of what we describe next when just
four consecutive stations are considered. There are the following cases to categorize the portions $S$.

The first case occurs when in the round just after the prefix no verification is scheduled by the algorithm. Then we extend $E$ by adding this round to the prefix.

The second case occurs when there are up to three consecutive verifications scheduled, subject to the property that if the first verification is completed in one round by a lack of transmission then the second similar happens, and so on up to three, but at most the fourth one would involve multiple verifications. The adversary does not activate any of the verified stations and the portion of up to three rounds is added to the prefix.

The third case occurs when there would be four consecutive verifications of single rounds just after the prefix $P$ if the adversary did not activate any of these stations. Let the adversary do not activate the first of these stations $v$ and activate the second $v+1$ with two packets. This station $v+1$ transmits successfully in the second round of the portion and next time in the third round. If the station $v+2$ is to be verified concurrently in the third round, then the adversary activates $v+2$ with one packet, which results in a collision in the third round of $S$. Now the best scenario for the algorithm is to have the packet of $v+1$ heard in the fourth round of $S$. Such $S$ results in a delay because only two stations have been verified in it. If this is how things happen then we extend the prefix by this $S$. If the station $v+2$ is not to be verified concurrently in the third round, so that only $v+1$ transmits, then this transmission is successful. Now there are sub-cases. If no station or only $v+2$ is to be verified in the fourth round, then the adversary does not activate $v+2$ at all, and just three rounds are verified in $S$ and we add $S$ after the current prefix. If both $v+2$ and $v+3$ are to be verified in the fourth round, then the adversary does not activate any of them, because it could be at most one such a station with the injection rate $\frac{3}{4}$. This is a critical portion at this point because an activation of both $v+2$ and $v+3$ would result in a collision and so a delay of verification. We add this portion $S$ after the current prefix.

The fourth case occurs when at least two stations are scheduled to be verified in the same round of the portion $S$ of four rounds immediately following the prefix $P$, which is
broken into three sub cases. More precisely, we mean that either (a) at least two stations are scheduled to be verified in the first round of \( S \), or (b) at most one station \( v \) in the first round of \( S \) and if there is no transmission then at least two stations are scheduled to be verified in the second round, or finally (c) at most one station \( v \) in the first round of \( S \) and if there is no transmission then also at most one station in the second round and with at least two stations scheduled to be verified in the third round. In the sub case (a), when \( v \) and \( v + 1 \) are scheduled to be verified in the first round of \( S \) then the adversary activates each of these stations so there is a collision in the first round of \( S \). Now the best case for the algorithm is for \( v \) and \( v + 1 \) to complete their verification by transmissions in the next two rounds of \( S \), and for \( v + 2 \) and \( v + 3 \) to transmit together in one round of \( S \). If this is what occurs then the adversary does not activate neither \( v + 2 \) nor \( v + 3 \), and this becomes a critical portion, because with the possibility of two activations the adversary could create a collision and so a delay in the fourth round. If such a best case does not occur then this means that either \( v + 2 \) or \( v + 3 \) is scheduled to be verified in a round in which one of \( v \) and \( v + 1 \) transmits solo, and then the adversary activates this station to be verified which results in a collision and so a delay. We extend the prefix of the execution by \( S \) in each of these cases. In the sub case (b), when at most one station \( v \) is verified in the first round of \( S \) and if there is no transmission then at least two stations are scheduled to be verified in the second round, then the adversary does not activate \( v \) but activates \( v + 1 \) and \( v + 2 \). This results in silence in the first round and a collision in the second. Even when \( v + 1 \) and \( v + 2 \) complete their verifications in the rounds three and four of the portion \( S \), then this results in a delay of verification, so we add the portion \( S \) as is. In the sub case (c), when the first two rounds could be silent, then the adversary does not activate \( v \) and \( v + 1 \) but activates \( v + 2 \) and \( v + 3 \). This results in the first two silent rounds followed by a collision in the third round. Now whatever happens in the third round, and the best case for the algorithm is when either \( v + 2 \) or \( v + 3 \) completes verification, still one of \( v + 2 \) and \( v + 3 \) cannot complete its verification in \( S \) so there is a delay. We add this portion \( S \) as just specified.
The execution $E$ is obtained with the adversary’s behavior consistent with injection rate $\frac{3}{4}$. This execution has the property that either there are infinitely many portions with delay or infinitely many critical portions. If the former is the case then we are done, since the obtained execution has unbounded packet delays. Otherwise, we construct a new execution in which the adversary’s behavior is consistent with the injection rate $\rho > \frac{3}{4}$. The adversary’s behavior is as if the injection rate were $\frac{3}{4}$ until it can inject one more extra packet to create a delay in a portion that otherwise would be categorized as critical. The adversary can create an execution with an unbounded number of delays because either there are infinitely many delays or the opportunity to convert a critical portion to one with delay will occur infinitely times, due to the inequality $\rho > \frac{3}{4}$.

Theorem 15 demonstrates a difference between the adversarial model of ad-hoc channels with the model of channels in which stations know the fixed number of stations attached to the channel and their names, as in that model bounded packet latency can be attained for any injection rate less than 1, see [11, 12], and mere stability can be obtained even for the injection rate 1, as it was demonstrated in [30].

9.3 Activation based protocols

We propose two activation-based algorithms, which handle different ranges of injection rates, depending on whether they are adaptive or not. The algorithms apply a paradigm to implement a global queue of stations to facilitate coordination among stations in their attempts to transmit. One of these queues uses the last-in-first-out queuing discipline and the other the first-in-first-out queuing discipline.

9.3.1 A non-adaptive protocol

We develop a non-adaptive activation-based algorithm which we call COUNTING-BACKOFF. It is designed for channels with collision detection. The underlying paradigm of algorithm COUNTING-BACKOFF is that active stations maintain a global virtual stack, that is, a last-in-first-out queue. Each station needs to remember its position on the stack, which is maintained as a counter with the operations of incrementing and decrementing
by one. The station at the top of the stack has the counter equal either to zero or one. The algorithm applies the rule that if a collision of two concurrent transmissions occurs then the station activated earlier gives up temporarily, understood as giving up the position at the top of the stack, while the station activated later persists in transmissions, understood as claiming the top position on the stack.

```c
/* if this is the first round after activation then backoff_counter = 0 */
if backoff_counter ≤ 1 then transmit a message

feedback ← feedback from the channel

if feedback = collision then backoff_counter ← backoff_counter + 1
else if feedback = silence then backoff_counter ← backoff_counter − 1
else if feedback = own message then
  if still active then backoff_counter ← 1 else
    backoff_counter ← 0
```

Figure 9.1: Algorithm COUNTING-BACKOFF code for one round of a station active in the beginning of a round.

The pseudocode of Protocol COUNTING-BACKOFF is presented in the Figure 9.1. Every station has a private integer-valued variable backoff_counter, which is set to zero when the station is passive. The copies of the variable backoff_counter are manipulated by the active stations according to the following general rules. An active station transmits a packet in a round when its backoff_counter is at most one. When a collision occurs, then each active station increments its backoff_counter by one. When a silent round occurs, then each active station decrements its backoff_counter by one. When a message is heard then the counters backoff_counter are not modified, with the possible exception of a station activated in the previous round.
A station that gets activated initially preserves its `backoff_counter` equal to zero, so the station transmits in the round just after the activation. Such a station increments its `backoff_counter` in the next round, unless its only packet got heard on the channel, in which case the station becomes passive. A station that transmits and its packet is heard withholds the channel and keeps transmitting in the following rounds, unless it does not have any other pending packets or a collision occurs. The variables `backoff_counter` are manipulated such that they implement positions on a stack, and thereby serve as dynamic temporary names for the stations that are otherwise nameless. This prevents conflicts for access among the stations that are already in the stack. Next, we make these intuitions precise.

We use the convention to refer to a station activated in round $t$ as the station $t$ and to its private variable `backoff_counter` as $\text{backoff\textunderscore counter}_t$.

**Lemma 11** When an active station executing COUNTING-BACKOFF has its `backoff_counter` positive at the end of a round, then this value may be interpreted as this station’s position on a global stack of stations, with the active station whose `backoff_counter` = 1 placed at the top.

**Proof:** We argue that the following stronger invariant is maintained in an execution of algorithm COUNTING-BACKOFF: if there are some $k > 0$ stations active in the beginning of a round then, at the end of this round, each such a station that remains active has a different value of its `backoff_counter` from the interval $[1,k]$, assigned in the inverse order of their activation. This is shown by induction on the round number. The first round is silent so the base of induction holds. Consider an arbitrary round $t + 1 > 1$ and assume that the invariant holds prior to this round. In round $t + 1$, either a packet is heard, or the round is silent, or there is a collision in it. Next we consider each of these three cases.

When a packet is heard in round $t + 1$, then there are two sub cases depending on whether some station was activated in round $t$ or not. The first sub-case occurs when a station $t$ got activated in round $t$. At this point $\text{backoff\textunderscore counter}_t$, equals zero, which
results in \( t \) transmitting its packet. By the inductive assumption, the stack is empty at round \( t \), because otherwise the station at its top would have transmitted in round \( t + 1 \) and created a collision. If station \( t \) is still active after the transmission, then \( t \) sets \( \text{backoff_counter} \leftarrow 1 \) and becomes the first on the stack, otherwise the stack remains empty. The second sub-case occurs when no station got activated in round \( t \). Then, by the inductive assumption, the station at the top of the stack transmitted in round \( t + 1 \). If after the transmission this station is still active, then nothing changes in the arrangement of the stations in the stack, and if the transmitting station becomes passive, then it resets \( \text{backoff_counter} \) back to zero, which is interpreted as this station leaving the stack.

The next case occurs when round \( t + 1 \) is silent. This means that no station on the stack has \( \text{backoff_counter} \) equal to one, so that either the stack is empty or there is at least one station on the stack with its \( \text{backoff_counter} \) equal to two. In the latter case, each station on the stack decrements its \( \text{backoff_counter} \) by one, making its value the true position on the stack.

The final case is of a collision in round \( t + 1 \). This means that some station has its \( \text{backoff_counter} \) equal to one and so is at the top of the stack, while another one has it equal to zero, which means it is a station newly activated in round \( t \). Now each active station increments its \( \text{backoff_counter} \) by one, which results in inserting the station \( t \) at the top of the stack.

\[ \square \]

**Theorem 16** When algorithm COUNTING-BACKOFF is executed against a 1-activation adversary of type \( (\rho, b) \), where \( \rho < \frac{1}{3} \), then there are at most \( \frac{3}{2} b \) packets queued in any round and the packet latency is at most \( \frac{6b}{1 - 3\rho} \).

**Proof:** We consider strategies by the adversary to delay packets. By Lemma 11, it is the packet at the bottom of the stack that is delayed longest, so a bound on time it takes to start with making an empty stack nonempty to having it become empty again is a bound on packet latency. The adversary may choose a strategy to fluctuate the size of the stack, but by rearranging the actions, within the constraints imposed by the adversary’s specification,
we may reduce each strategy to yielding at least the same bound on packet latency and such that it is in two parts: in the first part, the adversary works to keep the stack growing as much as possible, and in the second part, the adversary works to keep the stack decreasing in size as slowly as possible. We also consider a packet injected into the first activated station and estimate its delay, as then the adversary is not additionally constrained by previous injections. Observe that it is advantageous, with the goal to increase packet delay, to keep adding stations to the stack with just one packet to transmit rather than with multiple packets. We assume throughout that $b > 1$, as otherwise the adversary can apply only very restricted strategies.

The strategy to maximize packet latency is as follows. In the first round, inject two packets into station 1. In the second round, station 1 transmits and sets \( \text{backoff\_counter} \leftarrow 1 \), while one packet is injected into station 2. In the third round, stations 1 and 2 transmit simultaneously, station 1 sets its \( \text{backoff\_counter} \leftarrow 2 \) and station 2 sets \( \text{backoff\_counter} \leftarrow 1 \), while one packet is injected into station 3. This continues for a maximum possible number \( L \) of rounds, where \( L \) satisfies the equality \( L = b - 1 + \rho L \) so that \( L = \frac{b - 1}{1 - \rho} \). The number of packets after these \( L \) rounds is \( p_1 = L - 1 \) and the bottom packet has waited \( p_1 + 1 = L \) rounds. The number \( L - 1 \) is the upper bound on the number of packets queued. We have \( L = \frac{b - 1}{1 - \rho} < \frac{3}{2}b \), because \( 1/(1 - \rho) < \frac{3}{2} \) when the inequality \( \rho < \frac{1}{3} \) holds. At this point, the adversary cannot continue with a new activation per round, immediately for at least one round, which results in occasional successful transmissions, while at the same time activating new stations by injecting single packets into them as often as possible. What the adversary wants to obtain is to spend three rounds for one station: collision, hearing, silence.

Next we estimate the time it takes for the global virtual stack to become empty. To transmit \( p_1 \) packets requires at most \( 3p_1 \) rounds. During these rounds, up to \( 3\rho p_1 = p_2 \) new packets are injected. Observe that \( p_2 < p_1 \) as \( \rho < \frac{1}{3} \). It takes at most \( 3p_2 \) rounds to transmit \( p_2 \) packets. During these rounds, up to \( 3\rho p_2 = (3\rho)^2 p_1 = p_3 \) new packets are injected. This pattern is iterated to determine the numbers \( p_i \) so that it takes at most \( 3p_i \)
rounds to transmit $p_i$ packets. During $3p_i$ rounds up to

$$3p_i = (3\rho)^i p_1 = p_{i+1}$$

new packets injected. The time for this to occur is at most:

$$L + 3p_1 + 3p_2 + 3p_3 \ldots = L + 3p_1(1 + 3\rho + (3\rho)^2 + \ldots) \leq L + \frac{3L}{1 - 3\rho}, \quad (9.1)$$

where we used the inequality $p_1 < L$. The right-hand side of (9.1) can be upper bounded by

$$\frac{3b}{2} \left(1 + \frac{3}{1 - 3\rho}\right) \leq \frac{3b}{2} \left(\frac{4}{1 - 3\rho}\right) \leq \frac{6b}{1 - 3\rho},$$

because of the inequality $L = \frac{b - 1}{1 - \rho} < \frac{3}{2}b$. \quad \square

We can observe that algorithm COUNTING-BACKOFF on channels with collision detection is not fair when injection rate is $\frac{1}{3}$ and $b > 1$. To see this, consider the following execution. Let the adversary activate $b - 1$ stations in $b$ contiguous rounds, the first station activated with two packets and the following stations activated with one packet per station. After that, let the adversary keep activating a new station once in every contiguous segment of three rounds by injecting a single packet into it. This results in a collision in every third round, in a transmission in every third round, and in silence in every third round, to the effect that the stack never gets empty and the packet at its bottom is never heard.

9.3.2 An adaptive algorithm

We present an adaptive activation-based algorithm which we call QUEUE-BACKOFF. The underlying paradigm is that active stations maintain a global virtual first-in-first-out queue. This approach is implemented by the stipulation that if a collision occurs, caused by two concurrent transmissions, then the station activated earlier persists in transmitting while the station activated later gives up temporarily. This is a dual alternative to the rule used in algorithm COUNTING-BACKOFF.

Assume first that the channel is with collision detection. The pseudocode of algorithm QUEUE-BACKOFF is in Figure 9.2.
/* in the round of activation : queue_size = queue_position =
collision_count = 0 */

if \(0 \leq \text{queue}_{\text{position}} \leq 1\) then transmit a message

feedback ← feedback from the channel

if feedback = collision then

    if queue_size > 0 then queue_size ← queue_size + 1

else if queue_size = 0 then

    \text{queue}_{\text{position}} ← -1 ; \text{collision}_{\text{count}} ← \text{collision}_{\text{count}} + 1

else if feedback = a foreign message with \(K > 0\) and \(\text{queue}_{\text{position}} = -1\) then

    \text{queue}_{\text{size}} ← K ; \text{queue}_{\text{position}} ← K - \text{collision}_{\text{count}}

else if feedback = a foreign message with the “over” bit attached then

    \text{queue}_{\text{size}} ← \text{queue}_{\text{size}} - 1 ; \text{queue}_{\text{position}} ← \text{queue}_{\text{position}} - 1

else if feedback = own message and queue_size = 0 and still active then

    \text{queue}_{\text{size}} ← 1 ; \text{queue}_{\text{position}} ← 1

Figure 9.2: Algorithm Queue-Backoff for one round of an active station when the channel is with collision detection.
Every station has three private integer-valued variables: queue_size, queue_position, and collision_count, which are all set to zero in a passive station. The values of these variables represent a station’s knowledge about the global distributed virtual queue of stations, as captured by Lemma 12.

A message transmitted on the channel includes a packet and the value of the sender’s variable queue_size; if this is the last packet from the sender’s queue then a marker bit “over” is also set on in the message. In a round, an active station whose queue_position equals either zero or one transmits a message. The private variables are manipulated according to the following rules. When a collision occurs, then each active station with a positive value of queue_size increments its queue_size by one while an active station with queue_size = 0 increments its collision_count by one and sets queue_position ← −1. When a message with some value $K > 0$ of queue_size is heard and an active station has queue_position = −1, then the station sets queue_size ← $K$ and queue_position ← $K$ − (collision_count − 1). When a message with the “over” bit is heard, then each active station decrements its variables queue_position and queue_size by one. When a station is still active, it has just heard its own message and its queue_size equals zero, then the station sets its variable queue_size ← 1 and queue_position ← 1; this occurs when the global virtual queue is empty.

Some of the underlying ideas of this algorithm are similar to those used in the design of algorithm COUNTING-BACKOFF, they are as follows. A station that becomes activated transmits in the next round after activation, as then its queue_position is still zero. A station that transmits and the transmitted message is heard withholds the channel by transmitting in the following rounds, subject to packet availability. This works because the first transmission is with queue_position equal to either zero or one and the following ones with queue_position equal to one. A collision in a round means that some new station got activated in the previous round, because the station that has transmitted multiple times, with no other station successfully intervening, has its queue_position equal to one, while the only other possibility is to have this variable equal to zero, which is only
possible when inherited from the state when still a passive station. A difference with algorithm COUNTING-BACKOFF is that an active station cannot receive silence as feedback from the channel. This is because QUEUE-BACKOFF is adaptive and the “over” bit in messages eliminates silent rounds when there are some active stations.

**Lemma 12** When the queue_position of an active station executing QUEUE-BACKOFF is positive then this value may be interpreted as the number of this station’s position on a global first-in-first-out queue of stations, with the active station for which queue_position = 1 being at the front.

**Proof:** There are two invariants that hold true in any execution of algorithm QUEUE-BACKOFF.

The first invariant: the stations whose variable queue_size is positive store in this variable the number of active stations.

The proof of this invariant is by induction on the round number. When the first active station stays active for at least two rounds, then this sets its variable queue_size to one. The inductive step is by the rules of manipulation of the instance of this variable, namely, the “over” bit decreases the value and a collision increases it by one at each station.

The second invariant: at the end of a round, each station has a different value of the number defined as either queue_position, in the case queue_position > 0, or the number of active stations activated prior to the current round decremented by the value of the variable collision_count, in the case position_in_queue = 0, and these numbers fill the interval [1,k], where k is the number of active stations.

The proof of this invariant is by induction on the consecutive round numbers. We consider cases depending on what is the feedback from the channel. Silent rounds occur when there are no active stations activated prior to the current round. Once the feedback from
the channel is different from silence, it is either hearing a message or a collision. A message brings the number of active stations, and it allows to update \(\text{queue\_position}\), if it comes after a series of collisions, by the specification of the algorithm and the first invariant that \(\text{queue\_size}\) represents the number of active stations if heard in a message. A collision results in the stations that do not know the number of active stations record their offset from the next successful transmission by counting collisions. These values are all different as there is at most one station activated in a round.

The second invariant implies that once a message is heard and there are \(k\) active stations then, at the end of a round, each such station has different value of \(\text{queue\_position}\), all these values filling the interval \([1, k]\) and assigned in the order of activation. □

**Theorem 17** When algorithm \(\text{QUEUE-BACKOFF}\) is executed against a 1-activation adversary of type \((\frac{1}{2}, b)\) then there are at most \(2b - 3\) packets queued in any round and packet latency is at most \(4b - 6\).

**Proof:** The adversary may choose a strategy to fluctuate the size of the queue according to any pattern, but by rearranging the actions, subject to the constraints imposed by the adversary’s specification, we may reduce the pattern of fluctuations without compromising the worst possible bound on packet latency. Such a reduced strategy is in two parts. In the first part, the adversary works to keep the queue growing as much as possible. In the second part, the adversary works to keep the size of the queue decreasing as slowly as possible.

The active stations may be interpreted as being stored in a global first-in-first-out queue, by Lemma 12. The best strategy of the adversary is to maximize packet latency is to first build a queue of maximum size and then slow down its evolution when stations cannot be stopped to move towards the front. The maximum packet latency is the time spent in the queue by the packet injected at the moment when the global queue attains its maximum size. This strategy is implemented as follows by the adversary.
In the first round, the adversary injects two packets into the first station. In the subsequent rounds, the adversary activates a station per round by injecting one packet into it. This continues for a maximum possible number $y$ of rounds, where $y$ satisfies the equality $y = b - 1 + \frac{1}{2}y$ so that $y = 2(b - 1)$. The number of packets after these $y$ rounds is $y - 1 = 2b - 3$. The number $y - 1$ is the upper bound on the number of packets queued at any time. Next, the adversary injects as often as possible, which means at every other round. This results in alternating collisions and messages heard on the channel. Each packet in the global queue needs two rounds to move one position closer to the front. This means that a packet waits the number of rounds that is at most twice the maximum size of the global queue, which is $4b - 6$. □

Protocol \textsc{Queue-Backoff} was discussed as implemented for channels with collision detection. We may observe that an execution has the property that when the global queue is nonempty then each round contributes either a collision or a message heard on the channel. This means that collisions can be detected as void rounds by any involved active station, while passive stations do not participate anyway. It follows that this algorithm can be executed on channels without collision detection with small modifications only and with no change in its performance.

\subsection*{9.4 Full sensing protocols}

Stations running full-sensing algorithms can listen to the channel at all times and so they may have a sense of time by maintaining common references to past rounds. An idea could be to proceed through consecutive past rounds to give stations activated in them an opportunity to transmit, and then withhold the channel if needed to unload all their packets. This, just by itself, may result in unbounded packet latency, if we spend at least one round to verify any past round, for a possible activation in it, because recurring withholding of the channel would accrue unbounded delays. To prevent this phenomenon from occurring, we may consider groups of consecutive rounds and have stations activated in these rounds transmit in the same round. The relevant effect is that if at most one station got activated
in a group then we save at least one round of verification, which compensates for a delay due to withholding the channel. On the level of implementation, it is not necessary to maintain a counter of examined rounds, as it would grow unbounded, instead, one could count the number of rounds since the latest round examined for a station activated in it. These paradigms are employed in the algorithms presented in this section.

Channels are assumed to be with collision detection throughout the whole section. We will refer to the active stations by the respective rounds of their activation.

9.4.1 A non-adaptive algorithm

We propose a non-adaptive full-sensing algorithm which we call TRIPLED-ROUNDS. The algorithm operates by having the rounds of an execution partitioned into groups of consecutive three rounds per group; these groups are called segments. Simultaneously, an execution is partitioned into phases, the purpose of a phase is to verify all the rounds of the respective segment. A phase begins by the stations activated in rounds that belong to a segment transmitting together. A station that has its packet transmitted withholding the channel to unload all its packets. A silent round indicates that withholding the channel by a station is over.

The following is the underlying idea what can be exploited for sufficiently small injection rates. When an injection rate is smaller than \( \frac{2}{3} \), then less than two stations get activated in a segment on the average. In the case when one station gets activated in a segment, then just one round is spent to hear one packet and another one round to close its transmissions. This makes two rounds together and is smaller than the length of the segment whose rounds we want to verify.
if this is the first round of a phase and last_examined ≥ 3 and
last_examined − 2 ≤ waiting_time ≤ last_examined
then

transmit a message

feedback ← feedback from the channel

if feedback = silence then this round ends the phase
else if feedback = collision and v is first in segment then v becomes current
else if feedback = message then the phase becomes a one-station-segment
else

if v is current then transmit a message

feedback ← feedback from the channel

if feedback = silence then

if the phase is one-station-segment or current is third station in the segment

then this round ends the phase
else if v is next after the current station then v becomes current

last_examined ← last_examined + 1
if this round ends the phase then last_examined ← last_examined − 3
if v is still active then waiting_time ← waiting_time + 1

Figure 9.3: ALGORITHM TRIPLED-ROUNDS for one round of an active station v.
Protocol TRIPLED-ROUNDS has its pseudocode summarized in Figure 9.3. Each station has two private variables waiting_time and last_examined, each initialized to zero. Their interpretation is such that variable waiting_time denotes the number of rounds since the activation of the station, and last_examined is the number of rounds that have passed from the last round verified, which means examined for existence of a station activated in it, to the current round. The variable last_examined is manipulated by each station, including passive ones, in each round. This variable is treated as a counter of rounds, in that it gets incremented by one with each passing round. This is justified because the distance to that round grows by one with each passing round. Once a station gets activated, then its variable waiting_time is also treated as a counter of rounds, that is, it gets incremented by one with each passing round, until the station gets passive again and sets this variable equal to zero. The increments by one of these variables, when applicable, occur independently of any other operations on the variables, including decrementing, as discussed next. This means that a variable can be incremented and decremented in the same round.

An execution is partitioned into phases specified as follows. If a variable last_examined is at most two then a phase consists of a single round in which no station transmits while the variable last_examined gets incremented by one. Otherwise, when last_examined is at least three, then a phase with possible transmissions occurs, which takes care of packets in at most three active stations that have waited longest since their activations; all such three stations, whichever of them exist, make together the segment of the phase. These stations are identified by the following properties: the first station in the segment is determined by waiting_time = last_examined, if it exists, and the second station in the segment is determined by waiting_time = last_examined − 1, if it exists, and the third station in the segment has waiting_time = last_examined − 2, if it exists. When a station succeeds in transmitting in a round, then it is called the current one. A current station withholds the channel to unload its packets. When a phase begins, the stations in its segment transmit together. If this round results in silence then this concludes the
phase, as this means that the segment is empty. If a message is heard, then the phase is called a one-station-segment. The station that transmitted the heard message withholds the channel to unload all its packets, and when this is over then a silent round concludes the phase. If there is a collision, then, first, the first station in the segment unloads its packets one by one, followed by a silent round, possibly just one silent round occurs. Next, the second station in the segment unloads its packets one by one, concluded by a silent round. Finally, the third station in the segment unloads its packets, concluded by a silent round, which ends the phase. When a phase is over, then in its last round each station, including passive ones, decrements its variable last_examined by three and a new phase starts from the next round.

The variables last_examined and waiting_time have complementary roles, in that the former provides the identity of the last-examined round, represented as the distance from the current round counted backwards in time, while the latter is equal to the number of rounds a station has been active. This interpretation is made valid by the next two facts.

**Lemma 13** The variable last_examined has the same value at each station at the end of a round.

**Proof:** The variable last_examined is initialized to zero in the beginning of an execution. Next it is updated by each station in exactly the same way. Namely, it is incremented by one in each round, and its decremented by 3 at the end of a phase with possible transmissions, which is determined by the feedback from the channel, the same for each station.

□

**Lemma 14** The variable waiting_time has a unique value at each active station at the end of a round.

**Proof:** We examine how the instances of this variable are manipulated. When a station gets activated, then it is unique among the active stations in having waiting_time still equal to zero, as it is the only station activated in the round. All the other active stations
have their values of the variable different and greater than zero, as each of them had an opportunity to increment it by one at least once. The stations active at this moment increment their waiting_time values by one in unison, so they all stay distinct. □

It follows, from Lemmas 13 and 14, that the implementation of algorithm TRIPLED-ROUNDS by way of using variables last_examined and waiting_time is correct, as these variables determine the segments and each station’s position in one of them.

**Theorem 18** When algorithm TRIPLED-ROUNDS is executed against a 1-activation adversary of type \((\rho, b)\), where \(\rho < \frac{2}{3}\), then there are at most \(b \frac{3b}{2-3\rho}\) packets queued in any round and packet latency is at most \(\frac{3b}{2-3\rho}\).

**Proof:** We begin by computing the length of each kind of phases. We restrict our attention to the case when a station is activated with a single packet, as this is most advantageous to the adversary. A phase of a silent round takes one round. A phase that starts with hearing a message lasts two rounds. A phase with two stations activated in it, say they were activated in the first two rounds of the segment, takes six rounds, because it consists of collision, message, silence, message, silence, and the final silence. A phase with three stations activated in it takes seven rounds, because it consists of collision, message, silence, message, silence, message, and the final silence. In terms of delay per injected packet, activating two stations per segment is most advantageous to the adversary, as each injected packet contributes three rounds to the corresponding phase.

It follows that the strategy of the adversary to maximize the number of queued packets and their delays is to create the longest possible contiguous interval of segments such that each has precisely two stations activated in it. Let \(L\) be the length of such an interval of rounds. It satisfies the equation \(\frac{2}{3}L = b + \rho L\), which determines \(L = \frac{3b}{2-3\rho}\); we assume, to simplify notation that \(L\) is an integer divisible by three.

When the last segment in the initial interval of \(L\) rounds is completed, half of it has already been verified, because the rate of verification is half of the rate of time’s flow. What is available for verification at this point is an interval of \(L/2\) rounds. There are
\begin{equation}
\frac{2}{3} \cdot \frac{1}{2} L = \frac{b}{2 - 3p}
\end{equation}
packets in queues at this moment, which is their peak number. This segment is verified in time $2 \cdot \frac{1}{2} L = \frac{3b}{2 - 3p}$, which is an upper bound on packet latency. \hfill \square

### 9.4.2 An adaptive algorithm

We develop an adaptive full-sensing algorithm which we call PAIRED-ROUNDS. It is based on the idea to have groups of two consecutive rounds of an execution paired together. The stations in the same group are verified for activation in any of the rounds of a group by transmitting together. This resembles the approach of algorithm TRIPLLED-ROUNDS, the difference is in having smaller groups of two stations only and in using control bits in messages. Most of the terminology we introduced for algorithm TRIPLLED-ROUNDS is applicable, after replacing references to three stations by the corresponding ones to two stations.

An execution of the algorithm is partitioned into groups of two consecutive rounds, each called a segment, similarly as in Section 9.4.1. The consecutive series of rounds spent on the verification of the rounds in a segment is the phase corresponding to this segment. When the first message transmitted by a station is heard in any round, then the station withholds the channel to unload all its packets. The last packet transmitted by a station has the “over” bit attached to the message to indicate that the station becomes passive. A phase begins with the two stations activated in a round of the phase’s segment, if any, transmitting together. A silence means an immediate end of the phase. If, instead, a message is heard, then the transmitting station unloads its packets, with a message with the “over” bit concluding the phase. If a collision occurs then the first station of the segments begins its transmissions, which is followed by the second station’s transmissions. In the case of each station activated with a single packet, it takes three rounds to verify a segment of two rounds: collision and hearing messages twice, each with the “over” bit. This delay occurs sufficiently seldom to provide stability if injection rate is at most $\frac{3}{4}$. To see this, consider two consecutive segments, comprising four consecutive rounds in total. On the average, at most three stations get activated in them, so one of the segments is verified.
for the cost of only one round, for a saving of one round, and the total of four rounds
to verify the two segments of four rounds. This may seem as no progress with respect
of the method to have groups of singleton rounds and a phase to consist of verifying one
station. If so then this would be troubling, as the naive approach to verify one station per
round is susceptible of a cumulative delay growing unbounded, due to burstiness available
to the adversary who can make some stations withhold the channel for more than one
round. This phenomenon does not occur in an execution of algorithm TRIPLED-ROUNDS,
because the adversary needs to withhold activations through some rounds to be able to
inject more than one packet per station later, with this withholding resulting in a speedup,
of one round spent to verify a segment of two rounds.
if this is the first round of a phase and last_examined ≥ 2 and

last_examined − 1 ≤ waiting_time ≤ last_examined then

transmit a message

feedback ← feedback from the channel

if feedback = silence then this round ends the phase

else if feedback = collision and v is the first in segment then v becomes current

else if feedback = a message then the phase becomes a one-station-segment

if feedback = a message with the “over” bit attached then this round ends the phase

else

if v is current then transmit a message

feedback ← feedback from the channel

if feedback = a message with the “over” bit attached then

if the phase is a one-station-segment or the current station is second in segment

then this round ends the phase

else if v is next after the current station then v becomes current

last_examined ← last_examined + 1

if this round ends the phase then last_examined ← last_examined − 2

if v is still active then waiting_time ← waiting_time + 1

Figure 9.4: Algorithm Paired-Rounds code for one round of an active station v.
Protocol PAIRED-ROUNDS has its pseudocode presented in Figure 9.4. Each station has two private integer-valued variables \texttt{waiting\_time} and \texttt{last\_examined}, each initialized to zero. The role of these variables as counters at each station, including passive ones, is similar to how they are used by algorithm TRIPLED-ROUNDS. An instance of the variable \texttt{last\_examined} gets incremented by one in each round in every station, including passive ones. Once a station gets activated, then its variable \texttt{waiting\_time} also gets incremented by one in each round, until the station gets passive, when it is reset to zero. The increments by one of these variables occur independently of other operations on the variables, including decrementing, which are discussed next.

Now we explain how phases are implemented. If \texttt{last\_examined} is at most one then a phase consists of a single round in which no station transmits; in this round the variable \texttt{last\_examined} gets incremented by one. Otherwise, when \texttt{last\_examined} is at least two, then a phase with possible transmissions occurs, which takes care of the packets in the active stations that have waited longest since their activations. These stations are identified by the properties \texttt{waiting\_time} = \texttt{last\_examined} and \texttt{waiting\_time} = \texttt{last\_examined} − 1; the former is the first in the segment and the other one the second in the segment. When a phase begins, the stations with one of these properties transmit together. If a silence occurs, then this concludes the phase. If a message is heard, then the station that transmitted it keeps unloading its packets, the last packet with the “over” bit, which concludes the phase. If there is a collision, then, the first station in the pair unloads its packets first, the last packet with the “over” bit, and the second station in the pair unloads its packets next, the last packet with the “over” bit, which ends the phase. When such a phase is over, then in its last round each station decrements its variable \texttt{last\_examined} by 2 and a new phase starts from the next round.

The facts analogous to Lemmas 13 and 14 hold true for algorithm PAIRED-ROUNDS, so that the implementation of algorithm PAIRED-ROUNDS by way of using variables \texttt{last\_examined} and \texttt{waiting\_time} is correct, by analogy with algorithm TRIPLED-ROUNDS.
Theorem 19 When algorithm PAIRED-ROUNDS is executed against a 1-activation adversary of type $(\frac{3}{4}, b)$, then there are at most $\frac{4}{3}b$ packets queued in any round and packet latency is at most $2b$.

Proof: There are three kinds of phases: of no active station, of one active station, and of two active stations. A phase of the former two kinds last one round each, assuming that an active station holds just one packet, and a phase of the latter kind takes three rounds. Therefore it is advantageous for the adversary aiming to maximize the number of queued packets and packet latency to create as long an interval of troubling segments, requiring three rounds to verify a segment, as possible. Let us consider an interval of $L$ rounds in which each round contributes an active station by the adversary injecting one packet into it. Then $L$ satisfies the equation $L = b + \frac{3}{2}L$, so that $L = 4b$. By the time the last of these stations is activated, $\frac{2}{3}L$ stations activated first get verified. What still remains to be verified are $\frac{1}{3}L = \frac{4}{3}b$ stations, each holding one packet, which means that $\frac{4}{3}b$ is an upper bound on the number of queued packets. It takes $\frac{3}{2} \cdot \frac{4}{3}b = 2b$ rounds to verify these stations, which is an upper bound on packet latency. □
9.5 Conclusion

We have introduced ad hoc multiple access channels along with an adversarial model of packet injection in which deterministic distributed algorithms can handle non-trivial injection rates. These rates make the increasing sequence of $\frac{1}{3}$, $\frac{1}{2}$, $\frac{2}{3}$, and $\frac{3}{4}$, corresponding to the increasing power of algorithms. The highest injection rate that we can handle with bounded packet latency is $\frac{3}{4}$ and it is shown to the best possible, which means that algorithm PAIRED-ROUNDS is optimal. The optimality of the other three protocols in their respective classes is open. On the lowest end of this spectrum, injection rates arbitrarily close to $\frac{1}{3}$ but smaller than this number were shown to be handled by a non-adaptive activation-based algorithm on channels with collision detection with a bounded packet latency. It is an open question if a non-adaptive activation-based algorithm can provide bounded packet latency for sufficiently small positive injection rates on channels without collision detection.
10. Open problems and future work

Here we give some of the open problems and future work. Lower bounds for the deterministic protocols is still open and needs to be studied. The model with individual injection rates can be further extended to leaky bucket adversaries.

The experiments use randomization in packet injection. Theoretical work that is closer to the simulation environment that uses randomization is an open problem. We have only studied the worst case queues and latency. One could investigate the average queues and latency and possibly work on developing new protocols that are optimal for an average case rather than a worst case. Little’s formula is a fundamental result that relates average queues, average latency and average rate of arrival of packets in a queuing system. It is as follows. "The long-term average queues in a stable system $L$ is equal to the long-term average effective arrival rate, $\lambda$, multiplied by the average time a packet spends in the system, $W$; or $L = \lambda W$". If Little’s formula holds for an adversarial model of packet injection is also an open problem.
REFERENCES


118


119


