Broadcasting in Ad Hoc Multiple Access Channels*

Lakshmi Anantharamu and Bogdan S. Chlebus

Department of Computer Science and Engineering, University of Colorado Denver, Denver, Colorado, USA

Abstract. We study dynamic broadcasting in multiple access channels in adversarial settings. There is an unbounded supply of anonymous stations attached to the channel. There is an adversary who injects packets into stations to be broadcast on the channel. The adversary is restricted by the injection rate, burstiness, and by how many passive stations can be simultaneously activated by injecting packets into their empty queues. We consider deterministic distributed broadcast algorithms, which are further categorized by their properties. We investigate for which injection rates can algorithms attain bounded packet latency, when adversaries are restricted to be able to activate at most one station per round. The rates of algorithms we present make the increasing sequence $\frac{1}{3}$, $\frac{3}{8}$ and $\frac{1}{2}$, reflecting the additional features of algorithms. We show that no injection rate greater than $\frac{3}{4}$ can be handled with bounded packet latency.

Keywords: multiple access channel, adversarial queuing, distributed broadcast, deterministic algorithm, stability, packet latency.

1 Introduction

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 the popular implementation of local area networks by the Ethernet suite of algorithms [18]. 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.

Considering deterministic algorithms and their worst-case performance requires a 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

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-319-03758-9_29

* The full version of this paper is available at `http://arxiv.org/abs/1306.6109`. This work was supported by the NSF Grant 1016847.

T. Moscibroda and A.A. Rescigno (Eds.): SIROCCO 2013, LNCS 8179, pp. 237–248, 2013.

in a specification of a broadcast system is how much knowledge about the system can communicating agents use in their codes of algorithms. 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 such an ad hoc model, the number of stations is not set in any way, as stations come and go similarly as packets do; see [12] for the initial work on this model, and [6] 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 [22], while in the queue-free model the binary exponential backoff is unstable for any arrival rate [1].

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 [2,4,3,9,10]. 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.

In this paper, 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. The stations that want to use the channel to communicate join broadcasting activity. This needs to be coordinated with the other currently active stations by an algorithm, which could be associated with 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 a 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.

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. To make an anonymous system able to break symmetry in a deterministic manner, we restrict adversaries by allowing them to activate at most one station per round. This is shown to be sufficient for *deterministic* distributed broadcast algorithms to exist. We categorize algorithms into acknowledgment based, activation based and full sensing. Independently from that, we differentiate algorithms by the property 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 assess injection rates they can handle with bounded packet latency. Our non-adaptive activation-based algorithm can handle injections rates smaller than $\frac{1}{3}$ on channels with collision detection, the non-adaptive full-sensing algorithm can handle injection rate $\frac{3}{8}$ on channels with collision detection, and the adaptive activation-based algorithm can handle injection rate $\frac{1}{2}$ on channels without collision detection. We show that no algorithm can provide bounded packet latency when injection rates are greater than $\frac{3}{4}$.

Related work. The adversarial queuing methodology was introduced by Borodin et al. [8] and Andrews et al. [5], who used it to study the stability of store-and-forward routing in wired networks. Adversarial queueing on multiple access channels was first studied by Bender et al. [6], who considered randomized algorithms for the queue-free model. A deterministic distributed broadcasting on multiple access channels with queues in adversarial settings was investigated by Chlebus et al. [9,10] and by Anantharamu et al. [2,4,3]. That work on deterministic distributed algorithms was about systems with a known number of stations attached to the channel and with stations using individual names. Acknowledgment-based algorithms include the first randomized algorithms 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 algorithm 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 [21]. Goldberg et al. [13] gave related bounds for backoff, acknowledgment-based and full-sensing algorithms. Håstad et al. [16] compared polynomial and exponential backoff algorithms in the queuing model with respect to bounds on their throughput. For early work on full-sensing algorithms in channels with collision detection in the queue-free model see the survey by Gallager [12]. Randomized algorithms of bounded packet latency were given by Raghavan and Upfal [19] in the queuing model and by Goldberg et al. [14] in the queue-free model. Upper bounds on packet latency in adversarial networks was studied by Anantharamu et al. [2,4] in the case of multiple access channels with injection rate less than 1 and by Rosén and Tsirkin [20] for general networks and adversaries of rate 1. Deterministic algorithms for collision resolution in static algorithmic problems on multiple access channels were first considered by Greenberg and Winograd [15] and Komlós and Greenberg [17]. Algorithmic problems of distributed-computing flavor in systems in which multiple access channels provide the underlying communication infrastructure were considered by Bieńkowski et al. [7] and Czyżowicz et al. [11].

2 Technical Preliminaries

A multiple-access channel consists of a shared communication medium and stations attached to it. We consider dynamic broadcasting, in which packets are injected into stations continually and the goal is to have them successfully transmitted on the channel. A *message* transmitted by a station includes at most one

packet and some control bits, if any. Every station receives a transmitted message successfully, including the transmitting station, when the transmission of this message does not overlap with transmissions by other stations of their messages; in such a case we say that the message is *heard* on the channel. We consider synchronous channels which operate in rounds. Rounds and messages are calibrated such that transmitting one message takes the duration of one round. A message transmitted in a round is delivered to every 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.

Ad hoc channels. A station is said to be *active*, at a point in time, when it has pending packets that have not been heard on the channel yet. A station is *passive*, at a point in time, if either it has never had any packets to broadcast or all the packets it has ever received to broadcast have already been heard on the channel. These "points in time" are understood as real-number time coordinates, which are finer than the discrete partitioning of time into rounds. This is needed to avoid ambiguity in a situation when a station begins a round with just one pending packet, this packet is heard on the channel in this round, and new packets are injected into this station in this very round. We assume that there is an unbounded supply of passive stations. A passive station is said to get *activated* when a packet or multiple packets are injected into it. We impose quantitative restrictions on how passive stations may be activated in a round, which results in finitely many stations being active in any round. There is no upper bound on the number of active stations in a round in an infinite execution, since there is an unbounded supply of passive stations. Stations are *anonymous* when there are no individual names assigned to them. We consider channels that are *ad hoc* which means that (1) every station is anonymous, (2) an execution starts with every station initialized as passive, and (3) there is an unbounded supply of passive stations.

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

Broadcast algorithms. We consider deterministic distributed broadcast algorithms. In the context of communication algorithms, the "knowledge" of properties of a system means using such properties as a part of code of an algorithm. The algorithms we consider do not know the names of stations and the number of stations in the system. This is in contrast with previous work on deterministic distribute algorithms, see [2,4,3,9,10], where the names of stations and the number of stations could be used in a code. No information about adversaries is reflected in the code executed by stations. Every station has a private memory to store data relevant to executing a communication algorithm. This memory is considered to be unbounded, in the sense that it may store an arbitrary amount of data. The part of a private memory of a station used to store packets pending transmission is organized as a queue operating in a first-in-first-out manner. Successfully broadcast packets are removed from their queues and discarded. Packets are never dropped unless just after a successful broadcast. The *state* of a station is determined by the values of its private variables, with the exception of the queue to store packets, which is not a part of a state. One state is distinguished as *initial*. An execution begins with every station in the initial state and with empty queue. The algorithms we consider are distributed in the sense that they 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. An event is structured as the following sequence of actions occurring in a round in the order given: (i) transmitting a packet, (ii) receiving feedback from the channel, (iii) having new packets injected, (iv) making a state transition. Some among the actions (i) and (iii) may be void in a station in a round. A state transition depends on the current state, the feedback from the channel, and on whether new packets were injected in the round. In particular, the following actions occur during a state transition. If a packet has just been successfully transmitted then it is dequeued and discarded. If new packets have just been injected then they are all enqueued. If a message is to be transmitted in the next round, possibly subject to packet availability, then a message to be transmitted is prepared. Such a message may include the packet from the top of the queue, when the queue is nonempty, but a message may consist only of some control bits. A station that begins a round as active becomes passive when it successfully transmits its only pending packet. More precisely, such a station becomes passive at the point in time when this station receives the transmitted message as the feedback from the channel. When new packets are injected into this station in this very round, then it means that this passive station gets activated again. A station's status, of active versus passive, is dynamic in the course of an execution. In particular, an active station may 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 algorithms. We define subclasses of algorithms by specifying what can be sent in messages and how state transitions occur. We begin with the categorizations into full-sensing, activation-based and acknowledgment-based algorithms.

General algorithms are called *full sensing*. This means that stations may have state transitions occur in each round, according to the state-transition rules represented by the code. This term "full sensing" is to indicate that every station is sensing the channel in every round. This encompasses passive stations, which means that when a full-sensing algorithm is executed, then passive stations undergo state transitions from the beginning of the execution. Algorithms such that every 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 its last pending packet is heard on the channel, are called *activation based*. These algorithms have stations ignore the feedback from the channel when they do not have any packets to broadcast. Finally, algorithms such that a station stays in the initial state while passive and it resets itself to the initial state in a round in which a packet that it transmitted was heard on the channel are called *acknowledgment based*. This definition is correct due to the stipulation that the contents of queues do not belong to what constitutes a state; in particular, a station may be in the initial state when its queue is nonempty. A station executing a full-sensing algorithm 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, which is not the case in our considerations. An active station executing an activation-based algorithm may remember the history of the feedback from the channel since the activation. An active station executing an acknowledgment-based algorithm may remember the history of the feedback from the channel since the latest successful transmission or the latest activation, whichever occurred later. We understood these categorizations so that an acknowledgment-based algorithm is activation based, and an activation-based algorithm is full sensing. This is because a station executing an activation-based algorithm could be considered as receiving feedback from the channel but idling in the initial state when not having pending packets. When control bits are used in messages then we say that a algorithm is *adaptive*, otherwise the algorithm is *non-adaptive*. The categorization of adaptive versus non-adaptive is independent of the other three categorizations, into full sensing and activation based and acknowledgment based, so we have six categories of algorithms overall. This categorization of algorithms holds independently for channels with and without collision detection. The strongest algorithms are full sensing adaptive for channels with collision detection, while the weakest ones are acknowledgment-based non-adaptive for channels without collision detection.

The quality of broadcasting. An execution of an algorithm is said to be *fair* when each packet injected into a station is eventually heard on the channel. An algorithm is *fair* against an adversary when each of its executions is fair when packets are injected subject to the constrains of the type of the adversary. An execution of an algorithm has *at most Q packets queued* when in each round the number of packets stored in the queues of the active stations is at most Q. We say that an algorithm has *at most Q packets queued*, against an adversary of a given type, when at most Q packets queued in each execution of the algorithm against such an adversary. An algorithm is *stable*, against an adversary of a given type, when there exist an integer Q such that at most Q packets are queued in

any execution against this adversary. When an algorithm is unstable then the queues may grow unbounded in some executions, but no packet is ever dropped unless heard on the channel. The semantics of multiple access channels allows at most one packet to be heard on the channel in a round. This means that when injection rate of an adversary is greater than 1 then for any algorithm some of its executions produce unbounded queues. In this paper, we consider only injection rates that are at most 1. An execution of an algorithm has *packet latency* t when each packet spends at most t rounds in the queue before it is heard on the channel. We say that an algorithm has packet latency t against an adversary of a given type when each execution of the algorithm against such an adversary has packet latency t.

3 Limitations on Deterministic 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*.*

In the light of Proposition 1, we will 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 get activated. So when we refer to the *station* v, for an integer $v \geq 0$, then 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.

Proposition 2. *No acknowledgment-based algorithm is fair against a* 1*-activated adversary of type* (ρ, b) *such that* $2\rho + b \geq 3$*.*

Theorem 1. *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*.*

Theorem 1 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. In that latter model, a bounded packet latency can be attained for any injection rate less than 1, see [2,4], and a mere stability can be obtained even for the injection rate 1, as it was demonstrated in [9].

4 A Non-adaptive Activation Based Algorithm

We propose a non-adaptive activation-based algorithm 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. A passive or newly activated station has the counter equal to zero. The station at the top of the stack has the counter equal to one. The algorithm applies the rule that if a collision of two concurrent transmissions occurs then the station activated earlier gives up temporarily, which is understood as giving up the position at the top of the stack, while the station activated later persists in transmissions, which is interpreted as claiming the top position on the stack. Every station has a private integer-valued variable `backoff_counter`, which is set to zero when the station is passive. The private instantiations 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 it `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 which changes this variable from zero to one. A station that gets activated initially keeps 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, in which case the station becomes passive without ever modifying its `backoff_counter`. 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 transient names for the stations that are otherwise nameless. This prevents conflicts for access among the stations that are already on the stack.

Theorem 2. *When algorithm* Counting-Backoff *is executed against an adversary of type* (ρ, b)*, where* $\rho < \frac{1}{3}$ *and* $b > 1$*, then the packet latency is at most* $\frac{3b-3}{1-3\rho}$ *and there are at most* $\frac{3b-5}{2}$ *packets queued in any round.*

The bound on packet latency of algorithm Counting-Backoff given in Theorem 2 is tight. It follows that packet latency grows unbounded when the injection rate ρ approaches $\frac{1}{3}$. On the other hand, the bound on queue size given in Theorem 2 depends only on the burstiness of the adversary. The upper bound $\frac{3b-5}{2}$ on queues holds also when injection rate equals $\frac{1}{3}$, so algorithm Counting-Backoff is stable but not fair when injection rate equals $\frac{1}{3}$.

5 A Non-adaptive Full Sensing Algorithm

Stations executing a full-sensing algorithm can listen to the channel at all times and so they may have a sense of time by maintaining common references to the past rounds. This makes it possible to process consecutive past rounds to give

stations activated in them an opportunity to transmit. This, just by itself, may result in unbounded packet latency, if we spend at least one round to examine any past round for a possible activation in it, because the repeated case of active stations with multiple packets would accrue unbounded delays. To prevent this from occurring, one may consider groups of rounds and have stations activated in these rounds transmit simultaneously. If at most one station got activated in a group then we save at least one round of examination, which compensates for delay due to some stations holding more than one packet and for occasional collisions. To implement this approach, a channel needs to be with collision detection, which is assumed in this section. We refer to active stations by the respective rounds of their activation. A round gets *verified* when either all the packets of the station activated in this round have been heard or when it becomes certain that no station got activated in this round.

We present a non-adaptive full-sensing algorithm which we call Quadruple-Round. The rounds of an execution of the algorithm are partitioned into disjoint groups of four consecutive rounds, each called a *segment*. The first and second rounds of a segment make its *left pair*, while the third and fourth rounds make the *right pair* of the segment. The rounds of execution spent on processing the rounds in a segment make the *phase* corresponding to this segment. The purpose of a phase is to verify the stations in the corresponding segment.

A phase is organized as a loop, which repeats actions that we collectively refer to as an *iteration* of the loop. It takes at most four rounds to perform an iteration. An iteration is executed as follows. All the stations activated in the rounds of the phase's segment, if there are any, transmit together in the first round of an iteration. A station, that is scheduled to transmit, transmits a packet from its private queue, unless the queue is empty. This results in either a silence or a message heard or a collision, as a feedback from the channel. This creates the three corresponding cases which we consider next.

When the first round of an iteration is silent, then this ends the iteration and also the loop. This is because such a silence confirms that there are no outstanding packets in the active stations in the segment. When a message is heard in the first round of an iteration, then this ends the iteration but not the loop. The reason of continuing the loop is that the station, which transmitted the packet heard on the channel, may have more packets. If a collision occurs in the first round of an iteration, then the stations of the left pair transmit together in the second round. This leads to the three sub-cases presented next.

The first sub-case is of silence in the second round, which means that no station in the left pair is active. As the first round produced a collision, this means that each station in the right pair holds a pending packet. In this sub-case, the third and fourth rounds of the iteration are spend by the third and fourth stations of the segment transmitting one packet each in order, which concludes the iteration but not the loop. The second sub-case is of a message heard in the second round, which concludes the iteration but not the loop. The third case occurs when there is a collision in the second round of the iteration, which means that each station in the left pair of the segment holds an outstanding packet. In this case, the

third and fourth rounds are spend by the first and second stations of the segment transmitting one packet each in order, which concludes the iteration but not the loop.

Theorem 3. *When algorithm* Quadruple-Round *is executed against an adversary of type* $(\frac{3}{8}, b)$*, then packet latency is at most* $2b+4$ *and there are at most* $b + \mathcal{O}(1)$ *packets queued in any round.*

6 An Adaptive Activation Based Algorithm

Adaptive algorithms may use control bits in messages. 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 so 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. 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. 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 transmits a message when its `queue_position` equals either zero or one. 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` $\leftarrow -1$. 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` $\leftarrow K$ and `queue_position` $\leftarrow K-($`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` $\leftarrow 1$ and `queue_position` $\leftarrow 1$; 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. This is because the station that has

transmitted multiple times, with no other station successfully intervening, has its `queue_position` equal to one, while the other option is to have this variable equal to zero, which is only possible when this value is inherited from the state when still being a passive station.

Theorem 4. *When algorithm* Queue-Backoff *is executed against an 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* $4b - 4$*.*

Algorithm Queue-Backoff was presented as implemented for channels with collision detection. When the global queue is nonempty then each round contributes either a collision or a message heard on the channel. This means that when the channel is without collision detection, then 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 minor modifications in code only and with the same performance bounds.

7 Conclusion

We 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 $\frac{1}{3}$, $\frac{3}{8}$ and $\frac{1}{2}$. To improve beyond the rate $\frac{1}{3}$ attained by an activation-based non-adaptive algorithm, we designed a full sensing algorithm that handles injection rate $\frac{3}{8}$ and an adaptive one that handles the injection rate $\frac{1}{2}$. The optimality of these algorithms, in terms of the magnitude of the injection rate that the algorithms in the respective class of algorithms can handle with bounded packet latency against 1-activated adversaries, is open. We showed that no algorithm can handle the injection rate higher that $\frac{3}{4}$. It is an open question if any injection rate in the interval $(\frac{1}{2}, \frac{3}{4})$ can be handled with bounded packet latency by deterministic distributed algorithms against 1-activated adversaries.

References

1. Aldous, D.J.: Ultimate instability of exponential back-off protocol for acknowledgment-based transmission control of random access communication channels. IEEE Transactions on Information Theory 33(2), 219–223 (1987)
2. Anantharamu, L., Chlebus, B.S., Kowalski, D.R., Rokicki, M.A.: Deterministic broadcast on multiple access channels. In: Proceedings of the 29th IEEE International Conference on Computer Communications (INFOCOM), pp. 1–5 (2010)
3. Anantharamu, L., Chlebus, B.S., Rokicki, M.A.: Adversarial multiple access channel with individual injection rates. In: Abdelzaher, T., Raynal, M., Santoro, N. (eds.) Principles of Distributed Systems. LNCS, vol. 5923, pp. 174–188. Springer, Heidelberg (2009)

4. Anantharamu, L., Chlebus, B.S., Kowalski, D.R., Rokicki, M.A.: Medium access control for adversarial channels with jamming. In: Kosowski, A., Yamashita, M. (eds.) SIROCCO 2011. LNCS, vol. 6796, pp. 89–100. Springer, Heidelberg (2011)
5. Andrews, M., Awerbuch, B., Fernández, A., Leighton, F.T., Liu, Z., Kleinberg, J.M.: Universal-stability results and performance bounds for greedy contention-resolution protocols. Journal of the ACM 48(1), 39–69 (2001)
6. Bender, M.A., Farach-Colton, M., He, S., Kuszmaul, B.C., Leiserson, C.E.: Adversarial contention resolution for simple channels. In: Proceedings of the 17th Annual ACM Symposium on Parallel Algorithms (SPAA), pp. 325–332 (2005)
7. Bieńkowski, M., Klonowski, M., Korzeniowski, M., Kowalski, D.R.: Dynamic sharing of a multiple access channel. In: Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science (STACS), Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. Leibniz International Proceedings in Informatics, vol. 5, pp. 83–94 (2010)
8. Borodin, A., Kleinberg, J.M., Raghavan, P., Sudan, M., Williamson, D.P.: Adversarial queuing theory. Journal of the ACM 48(1), 13–38 (2001)
9. Chlebus, B.S., Kowalski, D.R., Rokicki, M.A.: Maximum throughput of multiple access channels in adversarial environments. Distributed Computing 22(2), 93–116 (2009)
10. Chlebus, B.S., Kowalski, D.R., Rokicki, M.A.: Adversarial queuing on the multiple access channel. ACM Transactions on Algorithms 8(1), 5:1–5:31 (2012)
11. Czyżowicz, J., Gąsieniec, L., Kowalski, D.R., Pelc, A.: Consensus and mutual exclusion in a multiple access channel. EEE Transaction on Parallel and Distributed Systems 22(7), 1092–1104 (2011)
12. Gallager, R.G.: A perspective on multiaccess channels. IEEE Transactions on Information Theory 31(2), 124–142 (1985)
13. Goldberg, L.A., Jerrum, M., Kannan, S., Paterson, M.: A bound on the capacity of backoff and acknowledgment-based protocols. SIAM Journal on Computing 33(2), 313–331 (2004)
14. Goldberg, L.A., MacKenzie, P.D., Paterson, M., Srinivasan, A.: Contention resolution with constant expected delay. Journal of the ACM 47(6), 1048–1096 (2000)
15. Greenberg, A.G., Winograd, S.: A lower bound on the time needed in the worst case to resolve conflicts deterministically in multiple access channels. Journal of the ACM 32(3), 589–596 (1985)
16. Håstad, J., Leighton, F.T., Rogoff, B.: Analysis of backoff protocols for multiple access channels. SIAM Journal on Computing 25(4), 740–774 (1996)
17. Komlós, J., Greenberg, A.G.: An asymptotically fast nonadaptive algorithm for conflict resolution in multiple-access channels. IEEE Transactions on Information Theory 31(2), 302–306 (1985)
18. Metcalfe, R.M., Boggs, D.R.: Ethernet: Distributed packet switching for local computer networks. Communications of the ACM 19(7), 395–404 (1976)
19. Raghavan, P., Upfal, E.: Stochastic contention resolution with short delays. SIAM Journal on Computing 28(2), 709–719 (1998)
20. Rosén, A., Tsirkin, M.S.: On delivery times in packet networks under adversarial traffic. Theory of Computing Systems 39(6), 805–827 (2006)
21. Tsybakov, B.S., Likhanov, N.B.: Upper bound on the capacity of a random multiple-access system. Problemy Peredachi Informatsii 23(3), 64–78 (1987)
22. Tsybakov, B.S., Mikhailov, V.A.: Ergodicity of a slotted ALOHA system. Problemy Peredachi Informatsii 15(4), 301–312 (1979)