Version of Record: https://www.sciencedirect.com/science/article/pii/S0022000023000685
Manuscript_3a48e2a43d54bf02b1e9448a7ad6468a

Restrained medium access control on adversarial shared channels[⋆,⋆⋆]

Elijah Hradovich[a], Marek Klonowski[a], Dariusz R. Kowalski[b]

[a] *Faculty of Fundamental Problems of Technology, Wrocław University of Science and Technology, Wrocław, Poland*
[b] *School of Computer and Cyber Sciences, Augusta University, Augusta, Georgia, USA*

Abstract

We study the fundamental problem of utilization of the channel shared by stations competing to transmit packets on the channel. In their turn, packets are continuously injected into stations' queues by an adversary at a rate at most ρ packets per round. The aim of the distributed medium access control algorithms is to successfully transmit packets and maintain system stability (bounded queues). We further restrain algorithms by introducing an upper bound k on the allowed number of active stations in any given round. We construct adaptive and full sensing protocols with optimal throughput 1 and almost optimal throughput $1-\epsilon$ (for any positive ϵ), respectively, in a constant-restrained channel. On the opposite side, we show that restricted protocols based on schedules known in advance suffer from a substantial drop of throughput, at most $\min\{\frac{k}{n}, \frac{1}{\log n}\}$. We compare our algorithms experimentally with well-known backoff protocols.

Keywords: multiple-access channel, restrained channel, contention resolution, adversarial queueing, throughput, stability

1. Introduction

Problem. Networks of different kinds and purposes commonly contain areas of contention between different actors over the access to common medium.

⋆Preliminary version containing some results presented in this paper occurred as [1]

⋆⋆Supported by Polish National Science Center (NCN) - grant UMO-2015/17/B/ST6/01897 (first and second author), and by the Polish National Science Center (NCN) grant UMO-2017/25/B/ST6/02553 and the National Science Foundation Grant No. 2131538 (third author).

The medium constrains a system by collisions or denial of service, when more than one device attempts to use it simultaneously. In our study we model actors by stations and a common resource by a shared channel. This model is also known as a multiple-access channel (MAC). In this model, there are n stations, each with a unique Id, connected to the same communication medium. Number n and the space of Ids is known by each station. Stations attempt to transmit packets via the shared communication channel in discrete intervals of time, called rounds. Due to the constraints of the channel, at most one successful transmission can happen at any round. While the original model allowed all of the stations to observe the channel every round, we introduce channel restraint k, limiting the number of stations able to be active (i.e., transmit or listen) in any round by k. The restraint accounts for the cost of non-transmitting stations' activities (e.g. , power required for listening to the channel's feedback), which in many settings is comparable to the cost of transmission attempts.
We focus on dynamic scenario when an adversary injects (in an arbitrary way) at most $0 < \rho \leq 1$ packets per round, on average, to stations' buffers. The primary goal is to design algorithms that guarantee *stability*, that is, a property that the sizes of queues in buffers stay bounded, for the highest possible injection rate ρ, which we will be calling *throughput*. Another important aim is to minimize channel restraint to achieve the maximum possible (i.e., in non-restrained classical channel) throughput.

Our contribution. In this work we investigate how the channel restraint k, understood as the upper bound on the number of active stations per round, influences the throughput on that channel for different classes of algorithms. We construct optimal or nearly-optimal solutions for different classes of protocols studied in the literature (see [2]): achieving throughput 1 for adaptive protocols (i.e., algorithms that can use channel history and additional control bits attached to messages), throughput $1 - \epsilon$ (for any positive ϵ) for full-sensing protocols (i.e., algorithms that can use channel history but not control bits), and throughput $\Theta(\frac{k}{n \log^2 n})$ for non-adaptive protocols (i.e., algorithms that follow a fixed schedule of transmissions starting they got the first packet). The latter result is complemented by the upper bound for this class $\min\{\frac{k}{n}, \frac{1}{\log n}\}$. All the performance bounds of algorithms are presented in Table 1.
The main conclusion from our results is that for some classes, i.e., adaptive and full sensing protocols, we are able to construct strongly restrained algo-

Algorithm	Sec.	Injection	Queues	Restraint	Class
12 O'CLOCK-AD	3	$\rho = 1$	$\mathcal{O}(n^2 + \beta)$	2	Adaptive
12 O'CLOCK-FS	4	$\rho < 1$	$\mathcal{O}(n^2 + \beta)$	3	Full-sensing with CD
K-LIGHT IS	5	$\rho < \Theta\left(\frac{k}{n \log^2 n}\right)$	UNKNOWN	k	Non-adaptive
Impossibility	5	$\rho > \min\{\frac{k}{n}, \frac{1}{\log n}\}$	Stable	k	Non-adaptive

Table 1: A summary of the performance bounds of algorithms and impossibility results, broken into four main sub-topics. The adversary is of type (ρ, β), where ρ is the injection rate and β is the burstiness coefficient. The abbreviations used to specify properties or algorithms are as follows: IS = interleaved selectors, CD - collision detection. 12 O'CLOCK-AD and 12 O'CLOCK-FS stand for 12 O'CLOCK adaptive and 12 O'CLOCK full-sensing respectively.

rithms without decreasing throughput of the system (i.e., comparing to the respective families of protocols without channel restraint described e.g. in [3]). Note that for the adaptive class of algorithms we were able to achieve the maximum possible throughput for the smallest possible channel restraint. Surprisingly, in some other classes, e.g. , non adaptive protocols, restraining the channel may substantially limit the throughput of efficient solutions. Another consequence is that the amortized number of transmissions/listenings per packet is constant for our adaptive and full sensing algorithms, while it is $O(n \log^2 n)$ for the non adaptive one. Let us stress that our non adaptive algorithm uses a newly introduced combinatorial structure, called *k-light selector*, which we thoroughly study for its own independent interest.

Previous and related work. To the best of our knowledge, adversarial packet injections on multiple-access channel (*without restricting energy*) were considered for the first time in the context of contention resolution by Bender et al. in [4] and Chlebus et al. [5]. The authors of the former paper considered maximal possible throughput of randomized backoff protocols in queue-free model, while in the latter work deterministic distributed broadcast algorithms in the model of stations with queues were studied. Further results in this line considering the maximum rate for which stability of queues is achievable include Bender et al. [6], Chlebus et al. [3] and Anantharamu et al. [7],[8], wherein the authors considered a wide spectrum of models with respect to adversary's limitations and capabilities of stations and the channel (e.g. , distinguishing collisions from the silence on the channel). Algorithms with a partial knowledge of adversary strategy with attention to packet la-

tency were studied by Bienkowski et al. [9]. Chlebus et al. in [10] studied limitations of hearing on MAC in relation to universal stability.
This work can be seen as a continuation of the research line from Chlebus et al. [3] paper (see also references therein) for a new model *with* limiting the number of active stations (i.e., the restraint). In particular we use the same taxonomy of the models of communication channels (w.r.t. stations capabilities) and classes of protocols as in [3]. Let us stress, however, that limiting the number of transmissions requires completely different algorithmic approach as well as using different analytic techniques.
On the other hand, there are some papers dealing with energy constraints in parallel environment, however for substantially different models/problems. For instance, Chlebus et al. [11] consider energy restricted parallel routing. Randomized *queue-free* constant throughput-based model for contention resolution with only bounds on transmission energy being known was studied by De Marco et al. [12]. Energy-efficient leader election, size approximation and census in single-hop networks were studied by Chang et al. in [13]. Earlier research on energy complexity of leader election and related problems includes Jurdzinski et al. [14, 15] and Nakano et al. [16].
Explicit construction of selectors and selective families was studied by Indyk [17] and Chlebus et al. in [18]. Universally strong selectors in relation to packet-oblivious routing on multi-hop networks were studied by Cholvi et al. in [19].
Algorithms managing energy usage were discussed in the surveys by Albers [20] and Irani and Pruhs [21]. Routing subject to energy constraints have been by Chabarek et al. [22] and Andrews et al. [23, 24]. Reducing network energy consumption via sleeping and rate-adaptation was addressed by Nedevschi et al. [25]. Distributed power control improving the energy efficiency of routing algorithms in ad hoc networks has been proposed by Bergamo et al. [26].
Ethernet local area data networks are typically under-utilized; energy conservation schemes for shutting down network interfaces when using the Ethernet were proposed by Gupta and Singh [27]. Gunaratne et al. [28] investigated policies to dynamically modify the link data rate based on the demand imposed on the link rate as a means to reduce the energy consumption in Ethernet installations.
Hardware-related challenges were studied by Ogierman et al. [29], with a focus on adversarial jamming limited by the energy budget in MAC protocols. Physical layer effects on a single hop fading channel were also studied

by Fineman et al. [30], with particular attention to the spectrum reuse enabled by fading.

Motivation. The fundamental problem of accessing a single medium by multiple devices is faced in many distributed systems, including processor transactional memory, radio networks communication medium, access to a shared resource on machines or data-centers. When more than one device attempts to access it simultaneously, a collision or denial of service occurs. However, the number of simultaneous attempts also matters, both in practice and, as we will show in this work, in theory. In practice, channel access could be constrained by physical factors, such as power, energy or availability. For instance, the "energy" spent by devices during such access attempts could be capped, in order to minimize its waste (because at most one of the attempting stations may succeed). Another motivating example comes from hardware systems, which often are designed with a spike (maximal) power use in mind to prevent meltdown or blackout. The above examples show potential applications of our results, obtained in the context of contention resolution problem, to other related problems on restrained channels.

Organization of the paper. In Section 2 we present a formal model. Section 3 is devoted mainly to adaptive (strongest), algorithm, wherein stations can adapt their behaviour to the communication channel and add some information to the transmitted packages. We construct an algorithm that needs only a constant number of stations being switched on in each round, which guarantees stability for an adversary even for $\rho = 1$. In Section 4 we discuss weaker full-sensing protocols. We construct an algorithm with a collision-detection mechanism that is stable for an adversary with any $\rho \leq 1 - 1/n$. The weakest type of algorithms (non adaptive), wherein all actions are set before the execution of the algorithm, are discussed in Section 5. In Section 6 we present experimental results for the constructed algorithms as well as a comprehensive comparison with commonly used (also in real-life systems) back-off protocols.

2. Model

System. We follow the classical model of a shared channel, e.g. , [3, 31]. In this model we distinguish n stations attached to a transmission medium, called a *multiple-access channel (MAC)*, working according to the following

	1	2	3	4	5	6	7	8	9	10	11
A						X			X		
B	X		X			X		X		X	
C			X		X						
D				X		X		X			X
	B->	-	-	D->	C->	-	-	-	A->	B->	D->

Figure 1: Example channel states by stations activity: top row represents rounds from 1 to 11; leftmost column stands for station names A, B, C, D; "X" stands for the transmission of the corresponding station in the corresponding round; bottom row stands for the resulting state of the channel in the end of the round with the arrow representing successful transmission by the referred station; "-" stands for silence channel feedback. Rounds $1, 4, 5, 9, 10, 11$ resulted in successful transmissions; rounds $3, 6, 8$ resulted in collisions; rounds $2, 7$ were silent. Note that the channel feedback for rounds with collision and silent rounds is the same.

rules: (1) a packet transmitted by a station reaches all the stations instantaneously; (2) a packet is successfully received if its transmission does not overlap with any other transmissions.

We restrict attention to synchronous 'slotted' model, in which stations use local clocks ticking at the same rate and indicating the same round numbers. It is assumed that each station knows n. Global round numbering is available to the stations.

Each round consists of phases: transmission, listening and data processing. The stations, according to their programs, attempt either to transmit in the first phase or to listen to the channel in the second phase. The duration of a round and the size of a packet are mutually scaled such that it takes a round to transmit one packet.

We say that a station *hears* a transmitted packet when the station receives the transmitted packet successfully as feedback from the channel.

If exactly one station transmits a packet in a round then all the stations that are switched-on[1] in this round hear the packet, including the transmitting station.

When at least two stations transmit their packets in the same round then

[1]In the basic model we assume that all stations are switch-on all the time. We deviate from this assumption in extensions of the model when only a limited number of stations can be switched-on in a single round.

no station can hear any packet in this round, including the transmitting stations. We call this situation to be a *collision* on the channel. A round during which no packet is transmitted is called *silent*. An example of channel states relation to stations activity can be seen in Figure 1.

	1	2	3	4	5	6	7	8	9	10	11
A						X			X		
B	X		X			X		X		X	
C			X		X						
D				X		X		X			X
	B->	-	|	D->	C->	|	-	|	A->	B->	D->

Figure 2: Example channel states by stations activity for channels with collision detection: top row represents rounds from 1 to 11; leftmost column stands for station names A, B, C, D; "X" stands for the transmission of the corresponding station in the corresponding round; bottom row stands for the resulting state of the channel in the end of the round with the arrow representing successful transmission by the referred station; "-" stands for silence channel feedback; "—" stands for collision channel feedback. Rounds $1, 4, 5, 9, 10, 11$ resulted in successful transmissions; rounds $3, 6, 8$ resulted in collisions; rounds $2, 7$ were silent.

Collision detection. Optionally, algorithms can rely on the capability to distinguish silence from collision on the channel. This capability is known as *collision detection mechanism* or simply *collision detection*. Collision detection enhances channel feedback to three possible output states: successful transmission (with station name, if there is one in the packet), silence and collision. It is assumed that names cannot be recovered from packets participating in collision, thus only the fact of the collision occurrence can be recorded. An example of feedback of the channel with collision detection can be seen in Figure 2.

Packet arrival. We assume that packets are kept in individual queues by each station, till they are successfully transmitted. Packets arrival to stations' queues is called *injection*. We assume that an *adversary* can inject packets to stations of his choice according to limitations characteristic for a given adversary. Those limitations include *injection rate* ρ and *burstiness* β, where ρ and β are numbers such that $0 < \rho \leq 1$ and $\beta \geq 1$. The adversary (ρ, β) is defined as follows: in each continuous time interval of length t, the

adversary can inject at most $\rho \cdot t + \beta$ packets, in total; in any single round, the maximum number of packets that the adversary can inject is $\lfloor \beta + \rho \rfloor$. This adversarial model of packet injection is called *leaky bucket*; it was used before to model traffic in shared channels, in particular in [2, 3].

Channel restraint. In our paper [1] we introduced a concept of *channel restraint* each station can be at one of two states – *switched on* (on-mode) or *switched off* (off-mode). Only a switched-on station in a given round can transmit a packet or listen to the channel. In a round in which a station is switched on, the station can set its *timer* to any positive integer c, which results in the station spending the next c rounds in the off-mode and returning to the on-mode immediately afterwards. The following is assumed: (1) it costs one unit to keep a station switched on in a round, and (2) it costs a negligible amount to keep a station switched off in a round. When representing the whole system's channel expenditure in a given round, we make it equal to the number of stations that spend this round switched on. The upper bound on the number of stations that can be switched on simultaneously in a round is the *channel restraint* of the system. A multiple-access channel system is determined by the total number of available stations and the channel restraint. We assume that the adversary can inject packets into the station packet queue independently from the station mode. Therefore, the adversary can inject packets to stations in off-mode.

Protocol quality measures. We distinguish the following quality and performance measures of algorithms:
Stability – queues of all stations stay bounded by some function on model parameters n, ρ, β at any round;
Maximal latency – the maximal number of rounds spent by a packet in station's queues;
Channel restraint – upper bound on the number of online stations in one round (we also say that the channel is k-restrained);
Throughput – the injection rate ρ for which all executions of the algorithm are stable. Usually we are looking for maximal possible throughput[2].
The *queue size* measure of an execution of an algorithm is defined as the maximum number of packets queued in all stations in a round of this execu-

[2]Note that the maximal throughput may not exists. In such a case we try to find a the limes superior of throughputs.

tion. The *latency* measure of an execution of an algorithm is defined as the longest span of rounds a single packet have spent in stations queues. Both the queue size and latency are natural performance metrics of algorithms and are represented as functions of the size of the system n and the type of an adversary ρ, b. If the latency of an algorithm is bounded then queues are bounded as well, since a queue's size at a station is always a lower bound on the delay of some packet handled by this station. We say that an algorithm is *stable*, against a class of adversaries, if the queue size is bounded, for a given number of stations and an adversary in this class.
An algorithm has a *universally bounded latency* when latency is bounded against each adversary of injection rate less than 1; we refer to such algorithms as *universal*.

Knowledge. We say that a property of a system is *known* when it can be used in codes of algorithms. It is assumed throughout that the size of the system n and the channel restraint $k < n$ are known, but the adversary parameters ρ, β are not. Algorithms may have their correctness and performance bounds depend on the magnitudes of the unknown parameters of the communication environment. For example, an algorithm may be stable or have bounded latency for sufficiently small injection rates.

Algorithm correctness. We say that a protocol with channel restraint k and injection rates ρ is correct, when queues of all stations stay bounded at all times independently from adversary strategy and the number of switched-on stations is at most k.

Classes of protocols. Current classification in the literature, c.f., [2, 8], considers the following classes of distributed protocols, depending on their computational power in a round:
Adaptive protocols — each station may access the history of transmissions and each packet contains the unique name of the sender. Moreover the sender can add a constant number of bits to each packet, which other stations can read and make future decisions based on this information.
Full-sensing protocols — each station may access the history of transmissions and each packet contains the unique name of the sender. However, no extra bits can be added to a packet.
Non-adaptive protocols — each station computes some function, with its unique name and round number as an input, determining whether a transmission attempt should be made. Note that in the model without a global

clock, the round number is local (the number of rounds that passed from the one when the first packet is injected into a given station). Whenever a station has a packet to be transmitted, the station broadcasts it as long that the function indicates that an attempt needs to be done.
Backoff protocol can be described as a randomized non-adaptive algorithm with a fixed initial sequence of probabilities assigned to subsequent rounds. The probability of transmitting is a function of the station's name and the round number.
Note that these algorithm classes assume that each station has a unique name (also referred to as ID).

3. Adaptive protocol 12-O'CLOCK

3.1. Protocol description

The 12-O'CLOCK(n) algorithm, where n is the number of stations in the system, schedules exactly two stations to be switched on in a single round — one in the *transmitting* role and another in the *listening* role. Since only one of those stations has the right to transmit, collision never occurs and the channel restraint is 2. The algorithm allows for any adversary burstiness value β.

High level description. We call a group of n consecutive rounds a *cycle* if the last round r of the group satisfies $r = 0 \mod n$. End-of-cycle (or 12-O'clock) rounds play an important role in coordination and decision making during the execution; they also motivate the name of the algorithm.
Every station keeps an ordered list of all the stations. These lists are the same in every station at the beginning of a cycle; at such a moment they represent one list, which we call *the list*. Initially, the list consists of all the stations ordered by their names.
Stations take the transmitting role in their order on the list. The process of assigning transmitting stations to rounds can be visualized as passing a virtual token from station to station, such that a station holding the token is in the transmitting role. Station spends one round in the listening role before taking the token, in order to learn the status of the channel. When a cycle ends, then the token is typically passed on to the next station on the list. The order determined by the list is understood in a cyclic sense, in that the first station assumes the transmitting role after the last one in the list has concluded its assignment. An exception for this process occurs when the

transmitting station is moved to become the head of the list while keeping the token.
The exception is handled as follows: a transmitting station B holding the token has the right to keep it when it has at least $3n$ packets in its queue. In such a case the station considers itself Big and informs other stations about its status, by suitably setting a toggle bit in packets. All of the stations while in the listening role, learn from this bit that they have no right to take the token.
Station B has the right to keep the token until the first end-of-cycle (12-O) round with queue size not greater than $3n$ — once this condition is fulfilled, the station considers itself to be Last-Big, has the right to hold the token for one more full cycle and informs other stations by setting another toggle bit in its packets. By the end of this last cycle, all stations move B to the head of the list. Starting with the next cycle stations follow their routine, with station B being the head of the list and B holding the token to transmit in the first round of the cycle. This mechanism allows transmitting stations to stretch cycles, possibly indefinitely, should the adversary inject packets in a certain way, e.g. , into one station only.

Technical description. Station can be at one of the five states: Idle, Listening, Transmitting, Big or Last-Big. The last three states are given the right to transmit; they could be encoded by two bits when attached to the packet by the transmitting station. The Listening state is dedicated to listening, while in the Idle state the station is switched off. Finite state machine for the relationship between those states can be found on Fig. 3.

Pseudo-code. We assume that each station has its internal information saved in the local object called s. The internal information includes the list of stations, the state of the station (i.e. Idle, Listening, Transmitting, Big or Last-Big), as well as methods allowing to modify this information. The pseudo-code of the algorithm is presented as two procedures:
`transmissionPhase` (Algorithm 1) – executed in the transmission phase of the round;
`listeningPhase` (Algorithm 2) – executed in the listening phase of the round.
Both procedures are executed by switched-on stations.

Methods. Procedures rely on the following methods:

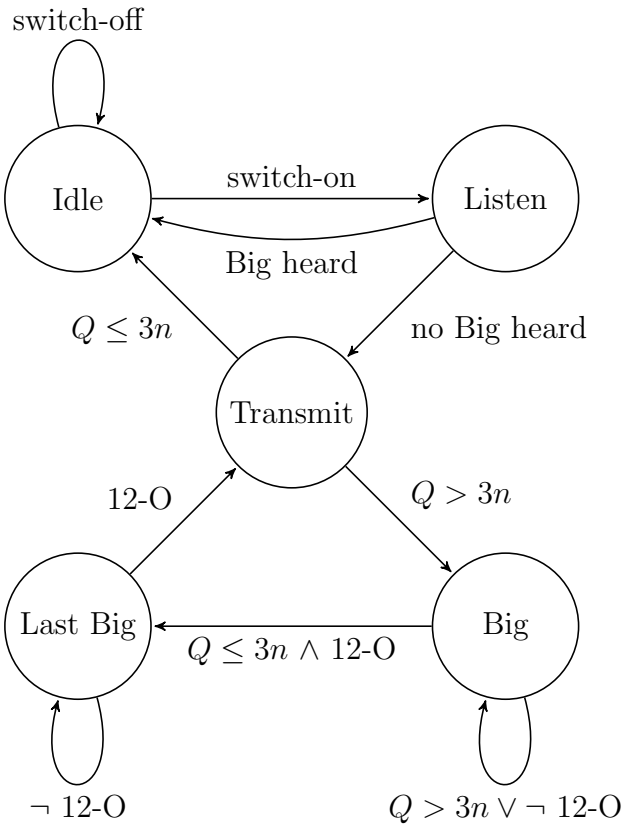


Figure 3: Finite state machine for a station in 12-O'clock adaptive algorithm executed independently on each station. Circle nodes represent states. Arrows are actions associated with state transitions, they happen with a tick of clock. Note that from the system perspective there are two stations with starting state being Transmitting and Listening respectively, otherwise the starting state is Idle as shown. Descriptions on arrows represent extra checks performed by the algorithm: Q stays for station queue size; n for system size; *12-O* for the end-of-cycle round; channel state heard by a station is described by: *Big heard* for packets with big or last-big bit being toggled. Arrows *switch-on* and *switch-off* represent a global clock-based decision for choosing on-mode and off-mode respectively.

```
Procedure transmissionPhase(round, station)
    switch station.state do
        case Transmitting do
            if station.queue > 3n then
                station.state := Big;
                station.transmit();
            else
                if station.queue > 0 then
                    station.transmit();
                station.state := Idle;
        case Big do
            if round = 0 mod n AND station.queue ≤ 3n then
                station.state := Last-Big;
                station.moveBigToFront(station.id);
            station.transmit();
        case Last-Big do
            station.transmit();
            if round = n - 1 mod n then
                station.state := Transmitting;
```

Algorithm 1: 12 O'clock adaptive algorithm, transmission phase.

`moveBigToFront(station ID)` — moves station of the input ID to the front of the (local) station list;
`transmit()` — transmits a packet from the station queue, attaches ID and state information to it;
`shouldWakeUp()` — checks the idle timeout of the station, that is, the number of rounds left until its predecessor could be in the Transmitting state. It starts from $n-1$ when the station drops the Listening state, and decreases by 1 each round. Upon becoming 0, the procedure outputs "true" and the station switches to Listening state.

Initialization. In the beginning all but the first two stations are in the Idle state, while the one with the smallest ID is in Transmitting state and its successor is in Listening state.

Idle state. In this state the station does not access the channel, it only keeps updating its idling time until the next wake-up — each round decreases by 1. The starting number of idling rounds is either n or $n-1$ or $n-2$, depending on the state from which the station switches to Idle and the

```
Procedure listeningPhase(station, channel)
    switch station.state do
        case Idle do
            if shouldWakeUp() then
                station.state = Listening;
        case Listening do
            switch channel.state do
                case Silence do
                    station.state = Transmitting;
                case Normal do
                    station.state = Transmitting;
                case Big do
                    station.state = Idle;
                case Last-Big do
                    station.state = Idle;
                    station.moveBigToFront(channel.transmitterId);
```

Algorithm 2: 12 O'clock adaptive algorithm, listening phase.

packet on the channel, see the description of Listening and Transmitting states below. When the idling time decreases to zero, the state switches to Listening.

Listening state. Station in the Listening state updates the local station list when the Last-Big transmission occurs on a channel. It changes its state to Transmitting upon receiving a packet from a station in the Transmitting state or upon no packet received. Otherwise, it becomes idle for the next $n-1$ or n rounds until wake-up. The latter idling time is caused by move of Last-Big station from behind of the Listening station location on the list of stations, to the front, therefore increasing the Listening station location on the list by 1.

Transmitting state. The Transmitting state is taken (from Listening state) by a station once per cycle in the round corresponding to its current position on the list of stations, unless there is a Big or Last-Big station in this round. Station in the Transmitting state changes its state to Big and transmits if its queue size is bigger than $3n$. Otherwise, it transmits being in the Transmitting state, provided it has a packet in its queue, and changes its state to Idle (in order to awake in its listening turn during the next cycle, after $n-2$ rounds).

Big state. At the end of each cycle, each Big station checks whether its queue size is still bigger than $3n$; if not, it changes its state to Last-Big. In any prior round, the Big station transmits a packet and remains in the same state. The following property can be easily deducted: once a station changes its state to Big (which happens when being in its regular Transmitting state), it stays there till the end of the cycle; it may then continue throughout whole next cycles, until it changes to Last-Big state at the end of one of them.

Last-Big state. Station in the Last-Big state transmits until the end of the cycle. It changes its state to the Transmitting in the end of the cycle after the last transmission happens. Note that, due to the condition of switching from Big to Last-Big state, a station remains in the Last-Big state during this whole cycle, from the beginning when it switched from the state Big to the end when it switches to Transmitting.

3.2. Protocol correctness and performance analysis

Consider the total size of the queues in the beginning of the cycle. If the size is greater than $\ell = n(3n-1)+1$ we say that the cycle belongs to a *dense* interval, otherwise it belongs to a *sparse* interval (here we consider intervals of time). This way the execution of the algorithm consists of interleaved dense and sparse intervals, each containing a number of whole cycles.

In relation to a fixed interval, we consider the following terminology: station is *pre-big* in a given round of the interval if it has not been in the Big state during this interval before that round, and it is *post-big* if it has been at least once in the Last-Big state during the interval by that round. Station is *potentially-big* if its queue size is bigger than $3n$ (i.e., the size allows the station to become Big eventually) or it is in a Big or Last-Big state. Note that this newly added terminology serves for the purposes of analysis only. We use the lower case writing convention to distinguish the newly added terms from the station states.

Observe that each station is pre-big in some prefix of the interval and post-big in some (disjoint) suffix of the interval; each of these periods could be empty or the whole interval. In-between of being pre-big and post-big, a station is continuously in a Big state.

We define the following types of cycles depending on availability of Big and Last-Big stations:

Type-1 cycle: without any Big or Last-Big station. Token is being passed in the Round-Robin way, by adapting Listening and Transmitting states.

This means that at any single round there is one station in the Transmitting state and one in the Listening state.
Type-2 cycle: with a station S starting to transmit as Big in some round of the cycle. Here, the token is being passed in the Round-Robin way by applying sequence of Listening and Transmitting states to each station on the list, until S transmits. Since S becomes Big, it keeps the token afterwards till the end of the cycle. Note that stations at Big and Transmitting states cannot occur simultaneously in the same round, because once there is a Big station all Listening stations immediately switch to Idle state instead of switching to Transmitting state.
Type-3 cycle: with a Big station S holding the token for the whole cycle. Upon waking-up in Listening state, a station will learn about the state of S and become idle until their scheduled wake-up round in the next cycle.
Type-4 cycle: with a Last-Big station S keeping the token for the whole cycle. Station can be in the Last-Big state only for a one cycle and after being in the Big state (at the end of the previous cycle). All stations after switching from Idle to the Listening state will learn about the Last-Big state of S and become idle until their scheduled wake-up round in the next cycle. The local lists of stations stay synchronized in the beginning of cycles; in fact, only the type-4 cycle changes the order of stations, and the whole cycle is needed to do it consistently in all stations (when they act as listeners) so that they all apply the move of the Last-Big station to the beginning of their local lists by the end of the cycle.

Lemma 3.1. *Each cycle is of one of the above four types.*

Proof. Algorithm's initialization conditions (Subsection 3.1) enforce that the first cycle type is either type-1 or type-2, as there is no Big or Last-Big station in the beginning. Type-1 can be followed only by the type-1 — if there is no potentially-big station during the cycle, or by the type-2 cycle otherwise. In the type-2 cycle the Big station is chosen during the cycle, and thus the cycle can be followed by the type-3 cycle — if the Big station queue size is above $3n$ at the end of the cycle, or by type-4 otherwise. The case of type-3 cycles is the same as the ones of type-2 described above, as in both types there is a Big station at the end (which determines conditions for the next cycle); they can be followed only by a cycle of type-3 or type-4. The type-4 cycle can be followed by type-1 — if there is no potentially-big station, or by type-2 cycle otherwise. Using an inductive argument over cycles, it can be concluded that each cycle is of one of the four defined types. □

Lemma 3.2. *In any dense interval, a station can cause a silent round (i.e., is in state Transmitting but has an empty queue) at most $n-1$ times while being pre-big.*

PROOF. Silent rounds occur when some station holds the token but has no packets in its queue. Note that it is only possible for stations in Transmitting state, as stations in any of Big states have more than n packets in their queues.
Assume that station S has no packets in its queue. Within a dense interval, in each round there is a potentially-big station. For any cycle, if potentially-big station is before S in the list, then S would receive no token or receive it and decrease its position in the list. The position of S cannot decrease more than $n-1$ times, as there can be no potentially-big station after S if it is last in the list. When S is the last on the list it either never has a possibility to transmit or becomes potentially-big. Pre-big station life-cycle terminates once the station is in the Big state by definition. □

Lemma 3.3. *In any dense interval, post-big or in a Big state station causes no silent round.*

PROOF. By definition of Big state, a station must have had more than $3n$ packets in its queue in the beginning of the current cycle or in the round of the cycle when it turned into the Big state. Therefore, in each round of the cycle it has packets and causes no silent round.
A post-big station S can be in any of the states. In the case of Listening and Idle states, the station does not attempt to transmit, thus it cannot cause a silent round. The case of Big state was already analyzed. If the station enters Last-Big state, it switches to this state from the Big state having more than $2n$ packets in its queue, thus in each round of the cycle it has packets and causes no silent round.
It remains to analyze the case when S is in the Transmitting state. Upon leaving the Last-Big state for the last time, it had at least n packets in its queues and was placed in the beginning of the list of stations, by the algorithm construction. Then, observe that S has had an opportunity to transmit only at some type-2 cycle, when there is no potentially-big station before it on the list or when S is potentially-big at the time it switches from Listening to Transmitting state. In the latter case, instead of staying in Transmitting state it immediately switches to Big state, in which case we already analyzed in the beginning of the proof.

In the former case, either potentially-big station after S becomes Big, which implies that in some of the next cycles, it switches to Last-Big state and the position of S on the list decreases without causing any more silent rounds, or S receives no token and so it cannot cause a silent round by default. The position cannot decrease more than $n-1$ times, because there can be no potentially-big station after S if S is the last on the list (the argument is similar to the one from the proof of Lemma 3.2). Since S had at least n packets when switching from its Last-Big state, it can transmit and decrease its position at most $n-1$ times or become Big, whatever comes first; in any case, it has at least one packet when transmitting. □

Theorem 3.4. *The 12 O'clock adaptive protocol achieves throughput* 1 *on the channel with restraint* 2 *and the maximum number of packets stored in round is at most* $O(n^2+\beta)$.

Proof. Consider an adversary with injection rate $\rho = 1$ and a burstiness β. Within a sparse interval, there can be no more than $\ell + n + \beta$ packets in the stations at the end of any cycle for dense interval threshold ℓ. Indeed, the biggest possible number of packets that the system can start a cycle with is equal to ℓ, and the adversary can inject no more than $n+\beta$ packets in n consequent rounds of the cycle. Once the queue size becomes greater than ℓ in the beginning of a cycle, the sparse interval terminates and the dense interval begins.
In the remainder, we focus on dense intervals. Note that in the beginning of a dense interval, the number of packets in the system is at most $\ell + n$ plus the burstiness above the injection rate (upper bounded by β); indeed, as in the beginning of the preceding cycle the interval was sparse, the number of packets was not bigger than ℓ, and during that cycle the adversary could inject at most n packets accounted to the injection rate plus the burstiness. Within any dense interval, a station in the Big or Last-Big state is guaranteed to be in each cycle, by the pigeon-hole principle. It makes type-1 cycle impossible to occur. Consider type-3 and type-4 cycles: during those cycles a packet is transmitted in every round, and thus a silent round cannot occur; hence the number of packets does not grow (except of burstiness above the injection rate, but this is upper bounded by β at any round of the interval, by the specification of the adversary). In type-2 cycles, post-big stations cannot cause silent round, by Lemma 3.3, and stations in Big state cannot cause silent rounds as they always have more than $2n$ packets pending. Hence, type-2 cycles may have silent rounds caused only by pre-big

stations. However, there can be no more than $n-1$ pre-big stations in the system in the beginning of the dense interval (because there is at least one potentially-big station). Each pre-big station can cause no more than $n-1$ silent rounds, by Lemma 3.2. Observe that in each cycle with a silent round some potentially-big station will change its state to Big — silent round would not occur if there was a potentially-big station with higher position in the list than any empty station. Hence, there can be no more than $n-1$ cycles with silent rounds caused by same (pre-big) station. To summarize, there are at most $n-1$ cycles with silent rounds for each of at most $n-1$ pre-big stations, resulting in the upper bound of $\ell + (n-1)^2 + n + \beta$ on system queues. Since only one of those stations has the right to transmit, collision never occurs and channel restraint is 2. □

Note that the algorithm requires each station to store the list of stations with some auxiliary data (that is linear w.r. to the system size n). We do not see it as a limitation for most of the cases however, since station queue size is square to the system size in the worst case, as we prove it above.
That is, packets in the station queue should be stored in some sort of memory. To fairly compare different protocols – the queue size needs to be taken into an account together with the size of the state. To the best of our ability, we performed such a comparison by running simulations of algorithms in Section 6. Figure 7b is of particular interest in that context.

4. Full-sensing protocol 12 O'CLOCK

4.1. Protocol overview

The 12-O'CLOCK full-sensing with collision detection protocol is an adaptation of the protocol described in Section 3 to the more restrictive algorithm class. In this class, the protocol has no ability to attach control bits to individual packets. Let us stress however that the protocol maintains the ability to add transmitting stations' identities to individual packets.
Similarly to the original algorithm, each station maintains the copy of the ordered list of stations referred as *the list*. There is a conceptual token permitting a station to transmit a packet to the channel. The token is passed in Round-Robin way following the order of the list. Stations are scheduled to switch on and listen to the channel one round before receiving the token. More precisely, for each round t, algorithm schedules two stations to be switched on — station S holding the token and station S' following S in the

order of the list. Station S transmits a packet from its queue if it has one, and station S' listens to the channel. Station S' claims the token at round $t+1$ if S transmission was successful or there was a silence on the channel. Station S switches off in the end of the round t if there was collision on the channel (we describe how collisions can occur below) or the size of its queue is less than $3n$, where n is the number of stations.

In contrary, when station S discovers at round t that the size of its queue is greater than $3n$, S becomes *big* and withholds the channel starting from round $t+1$. It follows that, there are three switched on stations at round $t+1$: S – as it has claimed the token by withholding the channel, S' – as it has received the token by following the order of the list, and S'' – the station following S' in the order of the list and scheduled to listen to the channel. The token ambiguity at round $t+1$ results in collision if both S and S' have packets in their queues. However we use the fact of the collision to inform both S' and S'' about the claim of station S on withholding the channel. In the case of S' having no packets to transmit, there is no collision at round $t+1$. It follows that only the packet transmitted by S is heard on the channel, hence both S' and S'' recognise that S holds the token by extracting transmitter identity from the packet. For both of these cases, station S'' learns that it cannot take the token at round $t+2$, therefore no more collisions occur.

Withholding the channel by station S lasts until the first round τ satisfying the following conditions: (1) the queue size of S is less than $2n$ and (2) round τ is a *12 O'clock* round – meaning that $\tau \mod n = 0$.

Before round τ and while S is big, stations listen to the channel following the order of the list. Whenever such a listening station S' learns that S is big, S' moves the identity of S to the top of its local copy of the list. Since S can become big only with its queue size counting not less than $3n$ packets, there are at least n rounds with station S transmitting while being big. Therefore all of the stations learn that S is big and local copies of the list are synchronized by the end of round τ. Starting from round τ stations pass the token following the new order of the list and the system returns to the initial configuration.

By distinguishing silence from collision, the algorithm is able to manage edge cases, see the description below.

However, due to collisions the protocol is not universally stable, albeit we will prove its stability against injection rates $\rho \leq \frac{n-1}{n}$.

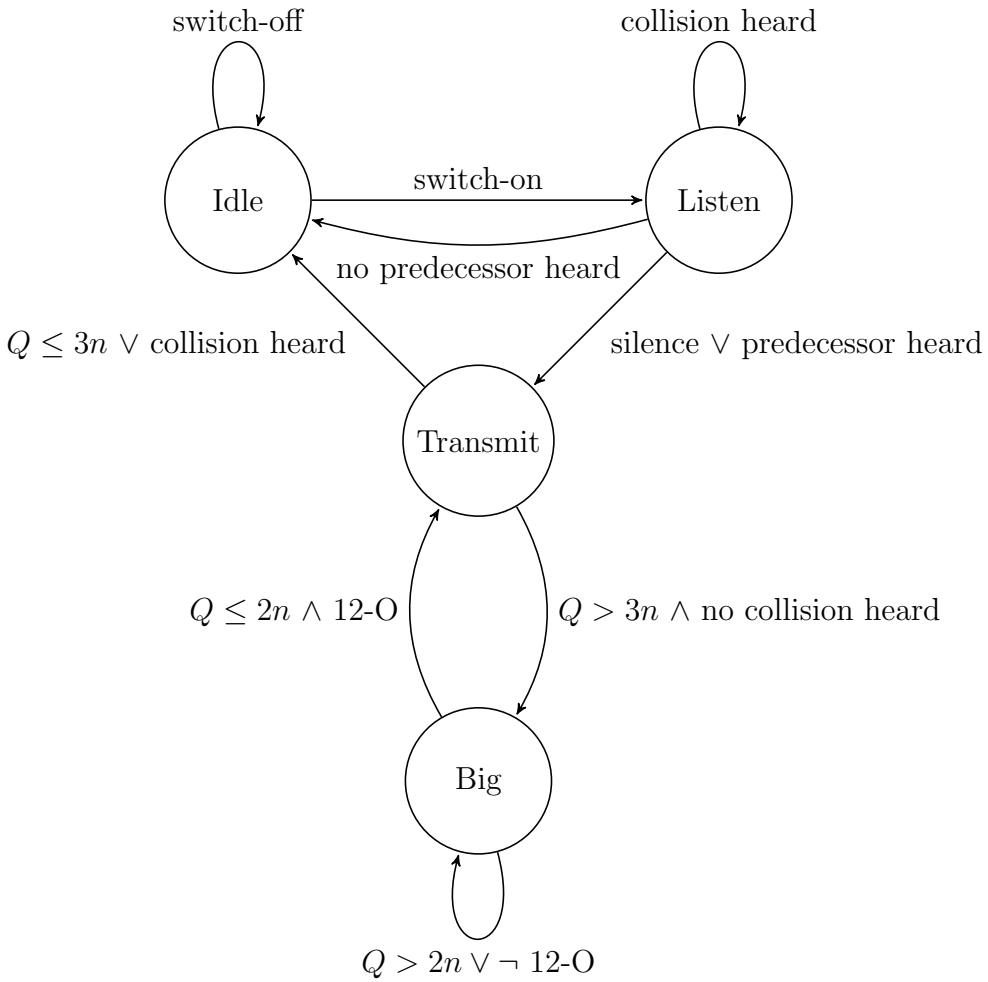


Figure 4: Finite state machine for a station in 12-O'clock full-sensing algorithm algorithm executed independently on each station. Circle nodes represent states. Arrows are actions associated with state transitions, they happen with a tick of clock. Note that from the system perspective there are two stations with starting state being Transmitting and Listening respectively, otherwise the starting state is Idle as shown. Descriptions on arrows represent extra checks performed by the algorithm: Q stays for station queue size; n for system size; *12-O* for the end-of-cycle round. Channel states are described as *silence heard*, *collision heard*, *predecessor heard*, *no predecessor heard* for station detecting silence, collision, predecessor or not predecessor transmissions on the channel respectively. Arrows *switch-on* and *switch-off* represent a global clock-based decision for choosing on-mode and off-mode respectively.

Technical description. We consider three channel states: *Silence* when there is no transmission, *Transmission* when there is single transmission on the channel, *Collision* when there is more than one transmission. Stations can be at one of four states: Idle, Listening, Transmitting or Big. The last two states are given the right to transmit; they are distinguished by the order in the list – only Big station can transmit out of the order of the list; in the only one possible case when Big station transmits within the order, collision occurs and later transmissions clarify the system state. The Listening state is dedicated to listening, while in the Idle state the station neither transmits or listens. We describe these states later in this section. As previously we assume that transmission happens before the listening phase. Simplified finite state machine for the relationship between those states can be seen in *Figure 4*.

Pseudo-code. We assume that each station has its internal information saved in the local object called s. The internal information includes the list of stations, the state of the station (i.e. Idle, Listening, Transmitting or Big), as well as methods allowing to modify this information. We also assume that there is a globally accessible *channel* information — containing the state of the channel (Silence, Collision, Transmission) and the identity of the transmitting station (if there was a successful transmission). The pseudo-code of the algorithm is presented as two procedures:
`transmissionPhase` (Algorithm 3) – executed in the transmission phase of the round;
`listeningPhase` (Algorithm 4) – executed in the listening phase of the round.
Procedures are executed by switched on stations.

Methods. The algorithm relies on the following methods:
`moveBigToFront(station ID)` — moves station of the input ID to the front of the (local) station list;
`transmit()` — transmits a packet from the station queue, with attached ID;
`shouldWakeUp()` — checks the idle timeout of the station, that is, the number of rounds left until its predecessor can be in the Transmitting state. Depending on the moment of when the station switches to Idle state, the starting value is either n or $n-1$ or $n-2$ and decreases by 1 each round. Upon becoming 0, the procedure outputs "true" and the station switches to Listening state.

```
Procedure transmissionPhase(station)
    switch station.state do
        case Transmitting do
            if station.queue > 0 then
                station.transmit();
                station.transmitted = true;
        case Big do
            station.transmit();
```

Algorithm 3: 12 O'clock full-sensing algorithm, transmission phase.

Initialization. In the beginning all but the first two stations are in the Idle state, while the one with the smallest ID is in Transmitting state and its successor is in the Listening state.

Idle state. In this state the station does not access the channel, it only keeps updating its idling time until the next wake-up — it decreases by 1 each round. The starting number of idling rounds is either n or $n-1$ or $n-2$, depending on the state from which the station switches to Idle and the packet on the channel, see the description of Listening and Transmitting states below. After awakening, i.e., when the idling time decreases to zero, the state switches to Listening.

Listening state. A station in the Listening state considers all three channel state cases, in the following way.
Collision on the channel occurs only when a Big station S interrupted its successor. No information is available on the channel, hence the Listening station keeps its state unchanged for one more round in order to hear an ID of the Big station. Note that there will be two stations in the Listening state and one in the Big state next round. Both listening stations would recognize S as Big and update their local station lists accordingly.
Upon hearing a silence, the Listening station knows that it will not interrupt a Big station transmission next round and thus it changes its state to Transmitting.
Finally in case of the transmission on the channel, the Listening station checks transmission ID on the channel and either it takes the token from its successor, or becomes idle and updates the local station list if it was not its predecessor's transmission. It becomes idle for the next $n-2$, $n-1$ or n rounds until subsequent wake-up; more specifically, the first idling time

```
Procedure listeningPhase(round, station, channel)
    switch station.state do
        case Idle do
            if shouldWakeUp() then
                station.state = Listening;
        case Listening do
            switch channel.state do
                case Silence do
                    station.state = Transmitting;
                case Collision do
                    station.state = Listening;
                case Transmission do
                    if channel.transmitterId=predecessor.id then
                        station.state = Transmitting;
                    else
                        station.state = Idle;
                        station.moveBigToFront(channel.transmitterId);
        case Transmitting do
            switch channel.state do
                case Silence do
                    station.state = Idle;
                case Collision do
                    station.state = Idle;
                    station.moveBigToFront(predecessor.id);
                case Transmission do
                    if station.transmitted then
                        if station.queue > 3n then
                            station.state := Big;
                        else
                            station.state = Idle;
                    else
                        station.state = Idle;
                        station.moveBigToFront(predecessor.id);
        case Big do
            if mod(round,n) = n-1 AND station.queue ≤ 2n then
                station.state := Transmitting;
                station.moveBigToFront(station.id);
```

Algorithm 4: 12 O'clock full-sensing algorithm, listening phase.

$n-2$ occurs when station waited additional round after collision on the channel, the second idling time $n-1$ occurs when the station hears a Big station which is currently located after it on the list of stations, and the last idling time n occurs when the Big station was located before it on the list.

Transmitting state. The Transmitting state is taken by a station once per cycle in the round corresponding to its current position on the list of stations, unless there is a Big station in the beginning of that round.
A station in the Transmitting state changes its state to Idle when there is a silence on the channel — it is possible only when it had no packets and there was no Big station in the beginning of this round. In the case of collision, it updates its local station list by moving its predecessor from the list to the front, as its is the only station which transmission on the channel would allow the Transmitting station to change its state from Listening to Transmitting. If a Transmitting station was successfully heard on the channel, there can be no Big station transmission in this round. Additionally, if the station has a queue size exceeding $3n$, it changes its state to Big and keeps transmitting accordingly starting from the next round. Otherwise, it changes its state to Idle, in order to awake in its listening turn during the next cycle, after $n-2$ rounds. If the station has not transmitted but a single transmission occurs on the channel, then this is a transmission from predecessor of Big station (any other Big station would cause the station not to switch to the Transmitting state in the first place, as it would switch directly from the Listening to Idle) which has not caused a collision only because the Transmitting station has had no packets to transmit. In this case the station behaves accordingly — updates the local list of stations and changes its state to Idle.

Big state. At the end of each cycle, a Big station checks whether its queue size is still bigger than $2n$; if not, it changes its state to Transmitting. In any other round, the Big station transmits a packet and remains in the same state. The following property can be easily deducted: once a station changes its state to Big (which happens when being in Transmitting state), it stays there at least till the end of the next cycle; it may then continue throughout the whole next cycles, until it changes to the Transmitting state at the end of some of them.

4.2. Protocol correctness and performance analysis
Similarly to the Adaptive protocol analysis, we consider the sum of the queues' sizes in the beginning of a cycle. If the sum of queues' sizes is greater

than $\ell = n(3n-1)+1$ we say that it belongs to the *dense* interval, otherwise it belongs to the *sparse* interval. This way any execution of the Full-sensing algorithm consists of *dense* and *sparse* intervals. In relation to a fixed interval we consider the following terminology: station is *pre-big* if it had never been in the Big state and it is *post-big* if it was at least once in the Big state, during the considered cycle. Station is *potentially-big* if its queue size allows it to become Big (provided other necessary conditions would hold) or it is in the Big state. Each cycle can be only of one of the three types:

Type-1. Without any Big station. Token is being passed in the Round-Robin way, by adapting Listening and Transmitting states. This means that at any single round there is one station in the Transmitting state and one in the Listening state.

Type-2. With a Big station S starting to transmit as Big in some round of the cycle. Here, the token is being passed in the Round-Robin way by applying the sequence of Listening and Transmitting states to consecutive stations on the list, until S transmits for the second time. The successor of station S cannot recognize S as Big since S is supposed to transmit by the default Round-Robin way of passing the token within the list order. Collision occurs if the successor of S has a packet to transmit. Otherwise, in the case of successful transmission, stations in Listening and Transmitting states active at this round would read the Big station ID from the transmission, both changing their states to Idle afterwards. Otherwise the station in the Transmitting state learns from the collision about the state of S, and then it changes its state to Idle and updates the local station list. The station which was in the Listening state at that time learns about the state of S a round after the collision, since it could not be a successor of any Big station.

Type-3. With a Big station S keeping the token for the whole cycle. All but one station after waking-up in the Listening state will learn about the state of S and become Idle until the next cycle. However, there is a station which would not recognize S as Big, but it will be interrupted by its transmission. Through the collision on the channel it would however learn about the state of S, an then it changes its state to Idle and updates the local station list.

The following two lemmas justify the usage of cycles defined above and provide the limit on the number of collisions. They will be used implicitly in the analysis.

Lemma 4.1. *Each cycle is of one of the three above types.*

PROOF. The initial conditions of the algorithm specified in Subsection 4.1, enforce the system to start in the type-1 cycle. Type-1 cycle can be followed by another type-1 cycle, if there is no potentially big station, or by a type-2 cycle otherwise.
In a type-2 cycle the Big station is chosen, and therefore it can only be followed by a type-3 cycle — this is because the Big station needs to transmit more than n packets in order to start to consider changing its state, which may happen only at the end of some cycle.
A type-3 cycle, with a Big station keeping the token (to transmit) for the whole cycle, can be followed either by the same type of a cycle if the adversary keeps injecting packets into the Big station, or by a type-1 cycle if there is no potentially-big station, or by a type-2 cycle otherwise. □

Lemma 4.2. *No more than one collision per cycle can occur.*

PROOF. Note that in a type-1 cycle collision may not occur, as at any single round there is station in the Transmitting state and another one in the Listening state.
In a type-2 cycle no collision occurs until the second transmission of a station in the Big state, by the same reasoning as for type-1 cycles. If the Big station successor has packets in its queues there is a collision on the channel. The station in the Transmitting state becomes Idle at the end of this round until the next cycle. Stations further down on the list cannot have the Big station as predecessor and would wake up in the Listening state, learn about the Big station from its transmission and change their state directly to Idle, hence there can be no more collisions.
A type-3 cycle with a Big station S holding the token. Consider the case, when type-2 cycle precedes. We divide stations of the system into two groups: group-A consists of stations after the Big station S on the list, which have already learned about the state of S and updated their local lists of stations. Group-B are stations before S on the list, which had no occasion to do so. If group-A is empty, then there is a single succeeding to S station in the group-B. It causes one collision due to assumption of default Round-Robin predecessor, which is S. The rest of the stations in this cycle will switch directly from the Listening to Idle state, thus no more collisions occur. If both group-A and group-B are not empty, then no station in the group-B can have S as predecessor, because S is down in the list for any station in group-B by definition, and its not last on their outdated list version since

group-A is not empty. Due to group-A stations having their lists updated, S is the first station in their lists, which together with nonempty group-B assumption makes it impossible to any station from the group-A to have S as predecessor. It follows that all of the group-A and group-B stations would change state directly from Listening to Idle, thus no collision occurs. If group-B is empty or type-3 cycle precedes the current cycle, then the cyclic order of the list does not change (i.e. each station has the same successor and predecessor in the beginning and the end of the cycle). It follows that there is a single succeeding to S station which causes a single collision due to assumption of default Round-Robin predecessor, which is station S. No more stations can have S as predecessor, thus the rest of the stations would change state directly from Listening to Idle and no more collisions occur. □

We call a round with collision caused by station in the Big state an *assertion* round. In relation to cycles we assume that there is an assertion round in every cycle, since this is the worst possible case – no more than one collision in a cycle can occur by Lemma above. By a *silent* round we understand any non-assertion round with no successful transmission. We say that a station *causes* a silent round if during this round it is in state Transmitting; note that it may occur only if the station has empty queue in this round. Observe also that there cannot be a Big station in a silent round, as stations in Big state have more than n packets in their queues.

Lemma 4.3. *In any dense interval, a station can cause a silent round at most* $n-1$ *times while being pre-big.*

Proof. Silent rounds occur when some station holds the token but has no packets in its queue. Assume that a station S has no packets in its queue. Within dense interval, in each round there is a potentially-big station. For any cycle, if potentially-big station is before S on the list, then S would receive no token or receive it and decrease its position on the list. The position of S cannot decrease more than $n-1$ times, because there can be no potentially-big station after S if it is the last on the list. Since in the dense interval there is always a Big station, S as the last station in the list either has no possibility to cause silent round (when some other station S' before it in the list changes state to Big), or becomes Big itself. Pre-big station life-cycle terminates once station is in the Big state by our definition, hence through the whole pre-big life-cycle station S may cause no more than $n-1$ silent round. □

Lemma 4.4. *In any dense interval, a station causes no silent round while being post-big or in a Big state.*

PROOF. A post-big station S could be in a Big state, Transmitting state or in one of the other two states. In the latter case, it does not attempt to transmit, hence it cannot cause a silent round. If the station enters Big state, it switches from the Transmitting state at some round of the cycle, having more than $3n$ packets in its queue; it'll switch back to the Transmitting state when having less than $2n$ packets in the end of the cycle. Thus, in any round of the cycle the number of packets cannot drop below n, and hence no silent round occurs.
It remains to analyze the case when S is in the Transmitting state. Upon leaving the Big state for the last time, it had at least n packets in its queues and was placed in the beginning of the list of stations, by the algorithm construction. Then, observe that S has had an opportunity to transmit only at some type-2 cycle when there is no potentially-big station before it on the list or when S is potentially-big at the time it switches from Listening to Transmitting state. In the latter case, instead of staying in Transmitting state it immediately switches to Big state, in which case we already analyzed in the beginning of the proof. Otherwise (i.e., in the former case), either some potentially-big station after S becomes Big, which implies that in some of the next cycles, it will switch back to Transmitting state and the position of S on the list decreases without causing any more silent rounds, or S receives no token and so it cannot cause a silent round by default. The position cannot decrease more than $n-1$ times, because there can be no potentially-big station after S if S is the last on the list (the argument is similar to the one from the proof of Lemma 4.3). Since S had at least n packets when switching from its Big state, it can transmit and decrease its position at most $n-1$ times or become Big, whatever comes first; in any case, it has at least one packet when being in Transmitting state. □

Theorem 4.5. *The 12 O'clock full-sensing protocol achieves throughput* $1-\frac{1}{n}$ *on a channel with restraint of* 3 *and the maximum number of packets stored in a round is at most* $\ell + (n-1)^2 + n + \beta = O(n^2+\beta)$.

PROOF. Injection rate stability limit of $1-\frac{1}{n}$ follows from inability to identify a Big station B by B station successor in the list. This results in potential collisions every cycle and in consequence wasting one each of n rounds.

The analysis of bounds on the queue size bases upon sparse and dense intervals defined above. Within a sparse interval, there can be no more than $\ell + n + \beta$ packets in the stations at the end of any cycle. Indeed, the biggest possible number of packets that the system can start a cycle with is equal to ℓ, and the adversary can inject no more than $n + \beta$ packets in n consequent rounds of the cycle. Once the queue size becomes greater than ℓ in the beginning of a cycle, the sparse interval terminates and the dense interval begins.

In the remainder, we focus on dense intervals. Note that in the beginning of a dense interval, the number of packets in the system is at most $\ell + n + \beta$. indeed, as in the beginning of the preceding cycle the interval was sparse, the number of packets was not bigger than ℓ, and during that cycle the adversary could inject at most n packets accounted to the injection rate plus the burstiness.

Within any dense interval, a station in the Big state is guaranteed to exist in each cycle, by the pigeon-hole principle. It makes type-1 cycle impossible to occur. Consider type-3 cycles: during those cycles a packet is transmitted in every round, and thus a silent round cannot occur; hence the number of packets does not grow (except of burstiness above the injection rate, but this is upper bounded by β at any round of the interval, by the definition of the adversary).

In type-2 cycles, by Lemma 4.4 silent rounds cannot be caused by post-big stations and stations in Big state cannot cause silent rounds as they always have more than $2n$ packets pending. Hence, type-2 cycles may have silent rounds caused only by pre-big stations. However, there can be no more than $n - 1$ pre-big stations in the system in the beginning of the dense interval (because there is at least one potentially-big station). Each pre-big station can cause no more than $n - 1$ silent rounds, by Lemma 4.3. Observe that in each cycle with a silent round some potentially-big station will change its state to Big — a silent round would not occur if there was a potentially-big station with higher position on the list. Hence, there can be no more than $n - 1$ cycles with silent rounds caused by same (pre-big) station. To summarize, there are at most $n - 1$ cycles with silent rounds for each of at most $n - 1$ pre-big stations, resulting in the upper bound of $\ell + (n - 1)^2 + n + \beta$ on the sum of the queue sizes in a round. $\square$

4.3. Stability bound improvement

It was proved in [3] that it is not possible to construct a full-sensing stable protocol against an adversary $\rho = 1$ for a system with a number of stations bigger than 3. Below we show how the 12 O'clock full-sensing protocol can be modified to withstand injection rates higher than $1 - \frac{1}{n}$.

Lemma 4.6. *For any given $\rho < 1$, the 12 O'clock full-sensing protocol can be modified to be stable against the adversary with injection rate ρ .*

Proof. Algorithm may handle any injection rate ρ smaller than 1 by following the strategy:
• Transmitting station considers itself Big when it has more then $2n + kn$ packets, where $k \geq \frac{1}{n(1-\rho)}$ is a positive integer;
• Transmitting station remembers of being interrupted by its predecessor, and instead of waking up after the subsequent nearly n rounds, as in the original 12 O'clock full-sensing protocol, it wakes up after kn rounds.
This way interruption may happen only once in kn rounds and the adversary with injection rate of $\rho = 1 - \frac{1}{kn}$ can be handled. We adjust the sparse/dense border value to $\ell' = n((2 + k)n - 1) + 1$, since the Big station definition has changed. Following the logic of the proof of Theorem 4.5, in any dense interval there are at most $k(n - 1)$ cycles for each of at most $(n - 1)$ pre-big stations, resulting in the upper bound of total queue size of $\ell' + k(n - 1)^2 + n + \beta = O(kn^2 + \beta)$. □

5. Non-adaptive protocols

In this section we consider non-adaptive protocols in k-restrained model, specified in Section 5.1.
In Section 5.2 we prove two limitations for this class of protocols. One of these limitations restricts the protocol class to use global-clock mechanism and it is followed as the basic requirement later through this section.
Next, we introduce a new combinatorial construction called *k-light selectors*, c.f., Sections 5.3 and 5.4. This construction is an extension of the well known selectors concept and we believe that it can be of independent interest.
We utilise k-light selectors to design an algorithm that is throughput-optimal up to the multiplicative polylogarithmic factor. The algorithm works in k-restrained channel and achieves throughput $\Theta\left(\frac{k}{n \log^2(n)}\right)$, see Sections 5.5 and 5.6.

5.1. Specification of non-adaptive protocols

In non-adaptive protocols, the decision to attempt to transmit is a function of the round number and the Id of the station. In effect, the schedule of transmission attempts can be represented by an unbounded binary sequence, in which 1 at position i represents an attempt to transmit in round i, where rounds are counted since the packet became pending, and 0 represents pausing. The round number is computed locally in the model without a global clock. That is, the round number is computed as the number of rounds from the injection of the first packet.
In our paper, we operate on non-adaptive protocols adjusted with a global clock mechanism, wherein the round number is common for all stations.
Note that pseudo-random number generators, commonly used by randomized protocols (including Backoff), often use the global time reference as the seed – it could be seen as analogous to our definition of non-adaptive protocols, which, based on the global time, compute 0-1 decisions about transmission attempt in subsequent rounds.
In our design of efficient non-adaptive protocol in Section 5.5, we will use the global clock mainly to synchronize transmission sequences across stations. The transmission sequences will be specified based on a selector family, defined and constructed in Sections 5.3 and 5.4.

5.2. Upper bound on throughput

We first prove that access to global clock is necessary to have non-trivial stability of k-restrained non-adaptive protocols, for any non-trivial $k < n$.

Lemma 5.1. *For any $\rho > 0$ and $k < n$, there is no k-restrained deterministic non-adaptive algorithm, without a global-clock mechanism which is stable for injection rate ρ.*

Proof. Assume, to the contrary, that P is a k-restrained deterministic non-adaptive protocol without a global clock that is stable for injection rate $\rho > 0$ in the system of n stations. Then, for each station S_i, there is a default transmission schedule p_i, where i is the index of the station. Note that p_i does not depend on global time, because we assumed that P does not have access to it. Because P is stable for a positive injection rate, each p_i contains a first occurrence of transmission event, indicated by bit 1 at some position of the sequence; call this position t_i.

Since the system is not equipped with the global clock mechanism, each station S_i will run its corresponding schedule p_i regardless of the round it starts considering a packet. Let us say that s_i is the first round when station S_i receives a packet, and thus it starts executing its transmission schedule. It follows that the first transmission of station S_i occurs at round s_i+t_i. In order to overload the system, the adversary follows the strategy: choose round e as $e = \max\{t_1, \ldots, t_n\}$; inject first packet to station S_i at round $s_i = e - t_i$. Under such adversarial scenario, all n stations transmit at round e, which violates the k-restrained assumption for $k < n$. This contradiction proves the lemma. Note that applying this scenario requires waiting for sufficient number of rounds to be able to inject n packets. This is however eventually possible for any $\rho > 0$. In this construction, we used burstiness equal to n. □

Next, we show instability result for the considered protocols for sufficiently large injection rates, even if the global clock is provided.

Theorem 5.2. *Non-adaptive algorithms in the k-restrained channel with global clock, where $k < n$, cannot achieve throughput higher than* $\min\{\frac{k}{n}, \frac{1}{\log n}\}$.

Proof. Assume first that $\frac{k}{n} \leq \frac{1}{\log n}$. Directly from the definition of stability, we show that the system cannot be stable if $\rho > \frac{k}{n}$. Namely, for any arbitrary threshold T we show that the adversary can inject packets generating a queue of size greater than T during some τ consecutive rounds. Parameter τ is to be fixed later. The channel restraint of k implies that at most k stations can be active in a single round and, therefore, during τ consecutive rounds there could be at most $\tau \cdot k$ station activities in total. In particular, at most $\tau \cdot k$ times, stations will be given an opportunity to transmit a packet and decrease their queues. There are n stations in the system, hence, by pigeonhole principle, there is a station S_j for some $j \in [n]$ that is allowed to transmit at most $\frac{\tau \cdot k}{n}$ packets during the considered τ rounds.
Non-adaptive protocols with a global clock provide the adversary with a power to know stations schedules in advance, as the adversary can calculate values of the protocol function for any round and each station; hence, it can pick the station S_j in advance. Once the station S_j is chosen, the adversary can inject $\lfloor \tau \cdot \rho \rfloor$ packets into the queue of S_j during τ rounds. It follows that station S_j at the end of τ considered rounds has at least $\lfloor \tau \cdot \rho \rfloor - \tau \cdot k/n \geq \tau \cdot (\rho - k/n) - 1$ packets. In order to go beyond the

threshold T it is enough to take τ such that $\tau \cdot (\rho - k/n) - 1 > T$, that is, $\tau > (T+1)/(\rho - k/n)$. Observe that such finite τ exists since, by our initial assumption, $\rho - k/n > 0$.
The second case, when the minimum formula equals to $\frac{1}{\log n}$, follows directly from Theorem 5 in [32]. □

5.3. k-light Selectors

Let us consider a set $N = \{1, \ldots, n\}$ and its subsets $S, X, Y \subset N$. We say that S *hits* X if $|S \cap X| = 1$, and S hits an element x if $x \in S$. We say that S *avoids* Y if $|S \cap Y| = 0$.

Definition 1. We say that a family $\mathcal{S} \subset 2^N$ is a (n, ω)-selector if for any subset $X \subset N$ such that $\omega/2 \leq |X| \leq \omega$ there are $\omega/4$ elements hit by at least one subset from $\mathcal{S}$.

Note that this definition is a special case of a *selective family* [18]. The intuition behind $\mathcal{S}$ is as follows: we can "separate" at least a fraction of elements of any subset X (of appropriate size) using sets that belong to $\mathcal{S}$.

Definition 2. We say that $\mathcal{S} = (n, \omega)$-selector is k-light if any $S \in \mathcal{S}$ satisfies $|S| \leq k$.

Theorem 5.3. *k-light (n, ω)-selector of size $m = O\left((\omega + n/k) \log n\right)$ exists.*

Proof. We divide the proof of the formula

$$m = O\left((\omega + n/k) \log n\right) \tag{1}$$

into two parts. The first part of the formula, $O(\omega \log n)$ for $\omega \geq \frac{n}{k}$, comes directly from Lemma 1 done by Chrobak et al. [33].
To prove the remaining part, let us assume that $\omega > 1$ and ω is divisor of n. Let m be the size of a selector to be fixed later. Let us choose independently m random subsets of $\{1, \ldots, n\}$ of size $l = \frac{n}{\omega}$. That is, $\mathcal{S} = (S_1, \ldots, S_m)$ is a random family. Let us consider any **fixed** sets $X, Y \subset \{1, \ldots, n\}$, such

that $\omega/4 \leq |X| \leq \omega$; $|Y| \leq \omega/4$ and a **random** S_i.

$$\begin{aligned}
&\Pr[S_i \text{ avoids } Y \text{ and hits } X] = \frac{\binom{|X|}{1}\binom{n-|X|-|Y|}{l-1}}{\binom{n}{l}} \\
&= |X| \cdot l \cdot \frac{(n-|X|-|Y|)^{\underline{l-1}}}{n^{\underline{l}}} = \frac{|X| \cdot l}{n-l+1} \prod_{i=0}^{l-2} \frac{n-|X|-|Y|-i}{n-i} \\
&> \frac{\frac{\omega}{4} \cdot l}{n} \prod_{i=0}^{l-2} \frac{n-\frac{5}{4}\omega-i}{n-i} \geq \frac{\frac{\omega}{4} \cdot l}{n} \left(1 - \frac{\frac{5}{4}\omega}{n-l+2}\right)^{l-1} \\
&\geq \frac{\frac{\omega}{4} \cdot \frac{n}{\omega}}{n} \left(1 - \frac{\frac{5}{4}\omega}{n/4}\right)^{l-1} \geq \frac{1}{4}\exp(-5) = c > 0 \ .
\end{aligned} \tag{2}$$

Let us bound the probability that for **any** sets X, Y such that $\omega/4 \leq |X| \leq \omega$ and $|Y| \leq \omega$ there exists an i such that S_i hits X and avoids Y. The probability of complementary event can be roughly bounded as follows:

$$\begin{aligned}
&\sum_{|X|=\frac{\omega}{2}}^{\omega} \binom{n}{|X|} \sum_{y=0}^{\omega} \binom{n}{|Y|} (1-c)^m \leq \omega^2 n^{2w} (1-c)^m \\
&\leq n^{4\omega}(1-c)^m \leq e^{4\omega \ln n - m \ln(1-c)} < 1 \ .
\end{aligned} \tag{3}$$

Note that the last inequality holds for some $m = O(\omega \log n)$ — for such m the random structure $\mathcal{S} = (S_1, S_2, \ldots, S_n)$ with probability greater then zero hits **any** X and avoids **any** Y of an appropriate sizes. Thus such a structure must exist and in consequence we can take $\mathcal{S}$ and use it for the reminder of the proof.
Now we show that $\mathcal{S}$ is a (n, ω)-selector. Let us take any X such that $\omega/2 \leq |X| \leq \omega$ and $Y = \emptyset$. By the property of $\mathcal{S}$ there exists S_{i_1} such that it hits X. Let $\{r_1\} = |S_{i_1} \cap X|$. Now let us construct $X = X \setminus \{r_1\}$ and $Y = Y \cup \{r_1\}$. Since still $\omega/4 \leq |X| < \omega$ and $|Y| \leq \omega/4$ we can find $S_{i_2} \in \mathcal{S}$, such that it hits the truncated X and avoids $Y = \{r_1\}$ thus there exists $r_2 = |S_{i_2} \cap X|$. Then we set $X = X \setminus \{r_2\}$ and $Y = Y \cup \{r_2\}$. We iterate such separation $\omega/4$ times to get $\omega/4$ distinct elements that are chosen from the initial X. Thus we get the first case of the theorem.

To prove the second part of the formula (1) $O\left((n/k)\log n\right)$ for $\frac{n}{k} > \omega$, first we need to construct an $\frac{n}{\omega}$-light selector $\mathcal{S}'$ of size $m = O(\omega \log n)$. Clearly, this is possible using the above construction. Then we need to partition each

$S_i \in \mathcal{S}'$ into $\lceil \frac{n}{k\omega} \rceil$ sets of size at most k to obtain a "diluted" selector. This results in $m = O(\frac{n}{k\omega}\omega \log n) = O(\frac{n}{k} \log n)$ sets of size at most k. □

5.4. Construction of selector in polynomial time

The previous section has provided a proof that k-light selectors exist, but does not specify how it can be constructed. It turns out that the construction is not trivial, therefore in the current section we present a polynomial time construction of k-light selectors. It uses two major components: dispersers and superimposed codes.

Dispersers. A bipartite graph $H = (V, W, E)$, with set V of inputs and set W of outputs and set E of edges, is a $(n, \ell, d, \delta, \varepsilon)$-*disperser* if it has the following properties: $|V| = n$ and $|W| = \ell d/\delta$; each $v \in V$ has d neighbors; for each $A \subseteq V$ such that $|A| \geq \ell$, the set of neighbors of A is of size at least $(1 - \varepsilon)|W|$. Ta-Shma, Umans and Zuckerman [34] showed how to construct, in time polynomial in n, an $(n, \ell, d, \delta, \varepsilon)$-disperser for any $\ell \leq n$, some $\delta = O(\log^3 n)$ and $d = O(\text{polylog } n)$.

Superimposed codes. A set of β binary codewords of length a, represented as columns of an $a \times b$ binary array, is a *d-disjunct superimposed code*, if it satisfies the following property: no boolean sum of columns in any set D of d columns can cover a column not in D. Alternatively, if codewords are representing subsets of $[a]$, then d-disjunctness means that no union of up to d sets in any family of sets D could cover a set outside D. Kautz and Singleton [35] proposed a d-disjunct superimposed codes for $a = O(d^2 \log^2 b)$, which could be constructed in polynomial time.

Polynomial construction of light selectors. We present a construction method for k-light (n, ω)-selectors of length $m = O(\omega \text{ polylog } n)$ for $m = O\left(\frac{n}{k} \text{ polylog } n\right)$ and $k \geq \frac{n}{\omega}$ for $k < \frac{n}{\omega}$, in time polynomial in n. Such setting is equivalent to constructing k-light (n, ω)-selectors of length $m = O((\omega + n/k) \text{ polylog } n)$ in time polynomial in n. The construction combines specific dispersers with superimposed codes. Let $0 < \varepsilon < 1/2$ be a constant. Let $G = (V, W, E)$ be an $(n, \omega/4, d, \delta, \varepsilon)$-disperser for some $\delta = O(\log^3 n)$ and $d = O(\text{polylog } n)$, constructed in time polynomial in n, c.f., [34]. Let $N_G(v)$ stay for the set of neighbors of node $v \in V$ in graph G. Let $\mathcal{M} = \{M_1, \ldots, M_a\}$ be the rows of the $c\delta$-disjunct superimposed code array of n columns, for $a = O((c\delta)^2 \log^2 n)$, constructed in time polynomial in n, c.f., Kautz and Singleton [35]; here δ is the parameter from the disperser G and $c > 0$ is a sufficiently large constant. W.l.o.g. we could

uniquely identify an ith of the n columns of the superimposed code with ith node in V.
For a constant integer c we define a k-light (n,ω)-selector $\mathcal{S}(n,\omega,k,c)$ with length at most $\min\{n, a|W|\alpha\}$, for some α to be defined later, which consists of sets S_i, for $1 \leq i \leq m$. Consider two cases. For the case when $n \leq m|W|\alpha$, we define $S_i = \{i\}$. In the case of $n > a|W|\alpha$, we first define sets F_j as follows: for $j = xa + y \leq a|W|$, where x, y are non-negative integers satisfying $x+y > 0$, F_j contains all the nodes $v \in V$ such that v is a neighbor of the x-th node in W and $v \in M_y$; i.e., $F_{x\cdot a+y} = M_y \cap N_G(x)$. Next, we split every F_j into $\lceil |F_j|/k \rceil$ subsets S of size at most k each, and add them as elements of the selector $\mathcal{S}(n,\omega,k,c)$. Note that each set S_i from $\mathcal{S}(n,\omega,k,c)$ corresponds to some set F_j from which it resulted by the splitting operation; we say that F_j is a parent of S_i and S_i is a child of F_j. In this view, parameter α in the upper bound $m \leq a|W|\alpha$ could be interpreted as an amortized number of children of a set F_j. We will show in the proof of the following theorem that $\alpha \leq \frac{nd\cdot(c\delta)^2\log^2 n}{k} \cdot \frac{1}{a|W|} + 1$.

Theorem 5.4. *$\mathcal{S}(n,\omega,k,c)$ is a k-light (n,ω)-selector of length*

$$m = O\left(\min\{n, (\omega + n/k)\ \textit{polylog}\ n\}\right) \tag{4}$$

for a sufficiently large constant c, and is constructed in time polynomial in n.

Proof. First we show that $\mathcal{S}(n,\omega,k,c)$ is a k-light (n,ω)-selector, for sufficiently large constant $c > 0$. Second, we consider the more complex case of $n > a|W|\alpha$.
Let set $A \subseteq V$ be of size between $\omega/2$ and ω. Suppose, to the contrary, that there are less than $\omega/4$ elements in A hit by some sets in $\mathcal{S}(n,\omega,k,c)$. It implies that there is a subset $B \subseteq A$ of size $\omega/4 + 1$ such that none of the elements in B is hit by sets from $\mathcal{S}(n,\omega,k,c)$.

Claim. Every $w \in N_G(B)$ has more than $c\delta$ neighbors in A, where $c\delta$ is a disjunctness parameter of $\mathcal{M}$. The proof is by contradiction. Assume, for simplicity of notation, that $w \in W$ is the w-th element of set W. Suppose, to the contrary, that there is $w \in N_G(B)$ which has at most $c\delta$ neighbors in A. More precisely, that $|N_G(w) \cap A| \leq c\delta$. By the fact that $\mathcal{M}$ is a $c\delta$-disjunct superimposed code, for $a = O((c\delta)^2\log^2 n)$, we have that, for every $v \in N_G(w) \cap A$, the equalities

$$F_{w\cdot a+y} \cap A = (M_y \cap N_G(w)) \cap A = M_y \cap (N_G(w) \cap A) = \{v\} \tag{5}$$

hold, for some $1 \leq y \leq a$.
This holds in particular for every $v \in B \cap N_G(w) \cap A$. There is at least one such $v \in B \cap N_G(w) \cap A$, because set $B \cap N_G(w) \cap A$ is nonempty due to $w \in N_G(B)$ and $B \subseteq A$. The existence of such v is in contradiction with the choice of set B. More precisely, B contains only elements which are not hit by sets from $\mathcal{S}(n, \omega, k, c)$, but $v \in B \cap N_G(w) \cap A$ is hit by some set $F_{w \cdot a+y}$, thus is also hit by some children $S_j \in \mathcal{S}(n, \omega, k, c)$ of $F_{w \cdot a+y}$. This makes the proof of the Claim complete.
Recall that $|B| = \omega/4+1$. By dispersion, the set $N_G(B)$ is of size larger than $(1-\varepsilon)|W|$, hence, by the Claim above, the total number of edges between the nodes in A and $N_G(B)$ in graph G is larger than

$$(1-\varepsilon)|W| \cdot c\delta = (1-\varepsilon)\Theta((\omega/4+1)d/\delta) \cdot c\delta > \omega d \ , \tag{6}$$

for a sufficiently large constant c. This is a contradiction, since the total number of edges incident to nodes in A is at most $|A| \cdot d = \omega d$. It follows that $\mathcal{S}(n, \omega, k, c)$ is a k-light (n, ω, k)-selector, for a sufficiently large constant c.
Before estimate the length m of this selector, note that the union of all sets F_j in the case $n > a|W|\alpha$ is at most $a \cdot (|V| \cdot d)$, because an element in some F_j corresponds to some edge in the disperser and repeats at most as many times as the number of rows a in the superimposed code $\mathcal{M}$. Hence, the amortized number of children $S \in \mathcal{S}(n, \omega, k, c)$ of a set F_j, parameter α, is at most

$$\frac{a \cdot (|V| \cdot d)}{k} \cdot \frac{1}{a|W|} + 1 \ . \tag{7}$$

The length m of this selector is thus at most

$$\begin{aligned} \min\{n, a|W|\alpha\} = O\left(\min\left\{n, \delta^2 \log^2 n \cdot \omega d/\delta + \frac{nd \cdot (c\delta)^2 \log^2 n}{k}\right\}\right) \\ = O\left(\min\left\{n, (\omega + n/k) \operatorname{polylog} n\right\}\right) \ , \end{aligned} \tag{8}$$

since $d = O(\operatorname{polylog} n)$ and $\delta = O(\log^3 n)$.
The case $n \leq a|W|\alpha$ is clear, since each element i in a set A of size between $\omega/2$ and ω occurs as a singleton in some selector's set, mainly in S_i. □

5.5. Protocol k-light Interleaved Selectors

W.l.o.g., to avoid rounding, assume that n is a power of 2 and therefore $\log n$ is an integer. We consider a sequence of $\mathcal{S}_1, \ldots, \mathcal{S}_{\log n}$, where $\mathcal{S}_i$ is k-light

$(n, 2^i)$-selector of size m_i. Let S_i^j be the j-th set of the i-th selector. That is, $\mathcal{S}_i = \{S_i^1, \ldots, S_i^{m_i}\}$. Let us consider the round number t that can be uniquely represented as $t = j \log n + i$ for $1 \leq i \leq \log n$ and $j \geq 0$. Station x transmits at round t if and only if x has a packet to be transmitted and $x \in S_i^{j \bmod m_i + 1}$. The order of sets of selectors "activating" stations is crucial for performance of the algorithm and motivate its name. This order is depicted on the Fig. 5.

5.6. Protocol correctness and performance analysis

Obviously in a single round at most k stations can transmit, since the sets S_i^j consist of at most k elements. We now analyze the performance of the protocol.

Theorem 5.5. *Assume that in round t there are r stations with nonempty queues, such that $2^i \leq r < 2^{i+1}$. The system will make at least $2^j/16$ packets heard on the channel before the round $t' = t + 8 \sum_{l=i}^{j} m_l \log n$ for some $j \geq i$.*

Proof. Let us first consider a set $X_0 \subset \{1, \ldots, n\}$ of stations such that $|X_0| = r$ and $2^{i-1} \leq r < 2^i$. Let $\mathcal{S}_i = \{S_i^1, \ldots, S_i^{m_i}\}$ be a $(n, 2^i)$-selector and $\mathcal{S}_{i+1} = \{S_i^1, \ldots, S_i^{m_{i+1}}\}$ be a $(n, 2^{i+1})$-selector for some $i < \log n$. We assume that stations from X_0 have non empty queues of packets. We observe all stations during $T = m_i + m_{i+1}$ rounds. We assume that the adversary can add packets to queues (even to initially empty queues) during the execution of the algorithm. Let X_t be the set of nonempty stations in round t. In the j-th round stations from $X_j \cap S_i^j$ transmit for $j < m_i$ and $X_j \cap S_{i+1}^{j-m_i}$ for $j \geq m_i$. In other words, in consecutive rounds transmit nonempty stations pointed by sets from $\mathcal{S}_i$, then stations from $\mathcal{S}_{i+1}$.

Lemma 5.6. *If less then $2^i/16$ different stations has transmitted during T rounds of the process then $|X_T| \geq \min\{r + 2^i/8, 2^{i+1}\}$.*

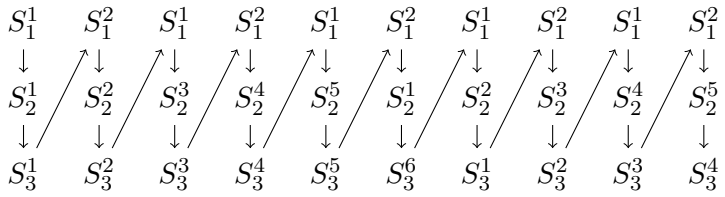


Figure 5: Interleaved Selectors: $\mathcal{A} = \{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3\}$, where $\mathcal{S}_1 = \{S_1^1, S_1^2\}$, $\mathcal{S}_2 = \{S_2^1, \ldots, S_2^5\}$ and $\mathcal{S}_3 = \{S_3^1, \ldots, S_3^6\}$.

PROOF. Let $Y = \bigcup_{i=1}^{T} X_i \setminus X_0$ be the set of all stations filled by the adversary during the process. Let $\mathcal{O}^*$ be the set of stations that are transmitted during the process. Moreover, let $\mathcal{T}(X)$ denote the set of the stations that transmitted at least once in the *static* case with the initial set X of nonempty stations, i.e. when the adversary does not add any packets.
Clearly, $|\mathcal{T}(X_0 \cup Y)| \leq |\mathcal{O}^*| + |Y|$. Indeed, adding Y to the set of stations with nonempty queues can increase the number of transmitting stations only by $|Y|$. On the other hand if a transmission of a station is blocked in the original process it must be also blocked in the case if all $X_0 \cup Y$ stations are nonempty at the beginning.
Let us consider two cases. In the first we assume $|X_0 \cup Y| < 2^{i+2}$. In follows that $|\mathcal{T}(X_0 \cup Y)| \geq 2^i/4$ because of the properties of selectors. Thus $2^i/4 \leq |\mathcal{O}^*| + |Y|$. We assumed however that $|\mathcal{O}^*| < 2^i/16$, thus $|Y| > 3/16 \cdot 2^i$. That is, the adversary added packets to at least $3/16 \cdot 2^i$ initially nonempty stations but less then $2^i/16$ has transmitted. Finally in he round T a least $r + 2^i/8$ are nonempty. In the remaining case, if $|X_0 \cup Y| > 2^{i+2}$ and only at most stations $2^i/16$ transmitted, the lemma holds trivially. □

Note that in **any** contiguous segment of $(m_i + m_{i+1}) \log n$ rounds, all sets of stations with nonempty queues from selectors $\mathcal{S}_i, \mathcal{S}_{i+1}$ are allowed to transmit (see Fig 5). Following Lemma 5 after $(m_i + m_{i+1}) \log n$ executed rounds at least one of the three events occurred: (1) $2^i/16$ transmitted; (2) the number of stations with nonempty queues increased by $2^i/8$; (3) there is at least 2^{i+1} nonempty queues.
Note that event (3) may occur at most $\log n - i$ times, similarly event (2) may occur at most $8(\log n - i)$ times till reaching the state of at least 2^{n-1} nonempty stations. Thus, after at most $\sum_{i=1}^{\log n - 1}(m_i + m_{i+1}) \log n + m_{\log n} \log n = O(\frac{n}{k} \log^2 n)$ rounds at least a fraction of nonempty stations will transmit at least one packet. □

Combining Theorem 5.5 with Theorem 5.3 we get:

Corollary 5.6.1. *The protocol achieves throughput* $\Theta\left(\frac{k}{n \log^2 n}\right)$ *on* k*-restrained channels.*

6. Algorithms simulations

In order to evaluate efficiency of developed protocols, we performed simulations for both new and existing algorithms and compared the results.

We analyzed the impact of the execution length, system size and injection rates on the queue sizes and channel restrain [36].
We collate Adaptive and Full-sensing versions of the 12-O'CLOCK algorithm as well as 8-light INTERLEAVED-SELECTORS and ROUND-ROBIN algorithms with BACKOFF exponential and polynomial algorithms. Our main simulation goals are to analyze and compare the following across the considered protocols:
General workflow for stable injection rates;
Maximal throughput, - we look for the lowest injection rates where queue size or latency show dependency on the number of rounds passed (because practically time-dependent behavior indicates instability);
Channel restrain below critical injection rates, so that channel restrain in stable executions could be evaluated.
A summary of the obtained results is presented in Figures 6-8b. Experiment results are presented without error bars to improve clarity, as several graphs are present in each figure. Each recorded result is an average of 120 experiments of one million rounds each.

6.1. Simulation implementation details

We have implemented algorithms 12 O'CLOCK adaptive and full-sensing versions, 8-light INTERLEAVED-SELECTORS, Round-Robin, as well as exponential, linear and square polynomial versions of BACKOFF algorithm in Java and Julia programming languages.

Backoff protocols. BACKOFF protocol is a popular randomised contention resolution algorithm. We follow the model and algorithm description from [31]. This kind of algorithm is defined as follows: each station S maintains a positive integer value ω called *window-size*. For each round t with station S having a packet in its queue, S randomly (uniformly) selects a transmission round t', such that $t' \geq t$ and $t' < t + \omega$. The initial value of the window size is $\omega = 1$ by default. If there is a collision on the channel at round t', S refers to the *window function* f defined by the algorithm to compute new window size ω': $\omega' = f(\omega)$. Otherwise the window size is reset: $\omega = 1$. Popular window functions include polynomial, square polynomial and exponential functions.
For the purposes of the simulation, we follow the parameters of window size functions defined in [31] as 2ω, $2\omega^2$ and 2^ω for polynomial, square polynomial and exponential functions respectively.

Additionally, in our simulation the size of the window ω is capped at constant 2048. Capping the window size is a technique commonly utilised in practice. It allows to protect protocols from unnecessary increase of the window size and thus improves their worst-case stability.

Non-adaptive protocols. Round-Robin protocol allows any station i to transmit alone in rounds i modulo n. 8-light Interleaved-Selectors are based on randomly generated binary matrices, tested to satisfy the definition of k-light (n, ω)-selector. Note that finding such selector is possible due to the small size of the utilised construction.

Adversary. In order to perform simulations, we need to define the behaviour of the adversary. We have chosen strategy of the adversary that seems to be challenging for the algorithm and reflects some real-life scenarios. We define an adversary by three parameters used at each round r: injection rate ρ – the probability that an adversary will have one more packet in its stock, burst-probability p – the probability of adversary making a decision to inject all of the stock packets at once, and finally the stock size limit β – a constant forcing the adversary to inject all of its stock packets once the stock size is equal to β.
We utilise two types of packet distribution in this section. We say that the packet distribution is *uniform* when the adversary selects stations to inject to with the same probability $P = \frac{1}{n}$. If uniform distribution is not specified, we assume that the adversary selects a station to inject to S_i with probability P_i, where $i \in \{1, 2, 3, \ldots, n\}$: $P_1 = P_2 = \frac{1}{3} + \frac{1}{3n}$; $P_{i>2} = \frac{1}{3n}$.
Injection rate ρ and burst-parameter p have values in $(0, 1)$. Note that the burst-probability parameter models the adversary injection behavior: between rare bursts of large numbers of packets (close to 0) and steady flow (close to 1). The stock-size β is a constant equal to 256, basing on operational buffer size limits. After performing some preliminary experiments for different values of p, we have chosen $p = 0.5$ for this presentation – it occurred not to influence the performance as much as expected.

Metrics. We took into consideration several measurements of queues of a protocol (at round r):
max-max - a maximal queue size of a single station occurring up to round r;
avg-max - an average, taken over r rounds, of a maximal queue size of stations at a round;

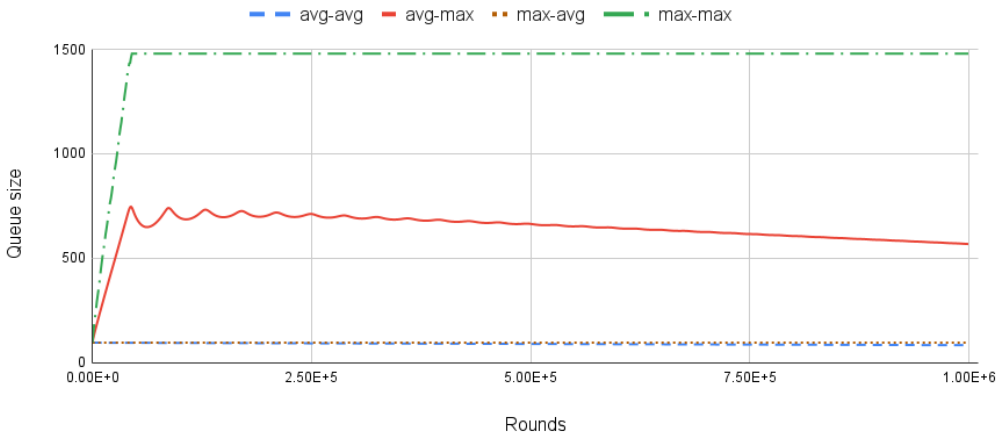


Figure 6: 12 O'CLOCK full-sensing protocol queues against injection rates $\rho = 0.968$, started with queues in each station equal to $q = 96$ during 1 mln rounds for a system size 32 against uniform packet distribution.

max-avg - a maximal over r rounds of an average queue size of all stations at a round;
avg-avg - an average over r rounds of an average queue size of all stations in a round.
Note that the max-max and max-avg measurements can not decrease and are always divergent against an adversary without burstiness limit (with probability 1).

Comparison of those measurements for 12 O'CLOCK full-sensing can be seen in Figure 6 for system size $n = 32$ against uniform packet distribution. The 12 O'CLOCK full-sensing protocol started with $3n = 96$ packets per station (i.e., the total system queue size equal to $3n^2$) stabilizes against injection rate $\rho = 0.968$, which is slightly smaller than the theoretical stability boundary $\rho = \frac{31}{32} = 0.96875$, for all four measurements and its both avg-max and avg-avg measurements decrease after handling the starting queues burst (Figure 6).
Based on the above results, we have chosen the avg-max measurement for further comparison of protocols. This is because when considering other three ways of measuring: max-max is highly volatile for the randomized protocols (and thus it would not be fair for comparison randomized and deterministic protocols) while avg-avg and max-avg do not envision the worst case scenario

we are focused on in this work. Note that the avg-avg measurement, studied in [31] and in many other previous papers considering stochastic injections, may yield stability while having single queues many times above the studied average.

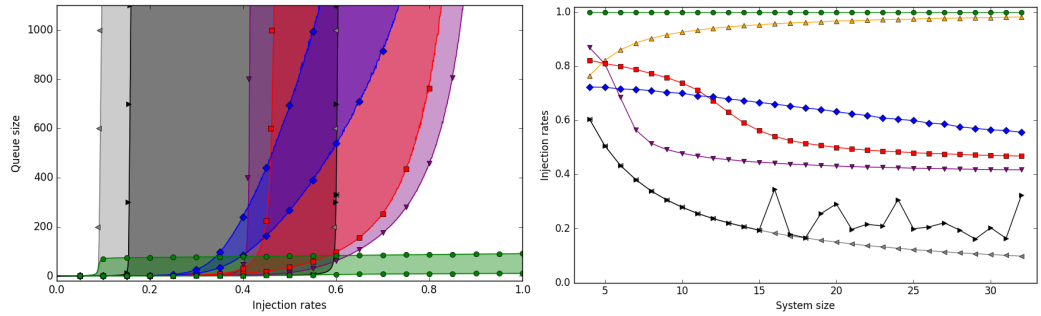
6.2. Bounds on stable injection rates

In order to see how system queues behave for different system sizes, we have combined simulation results for system sizes $n \in \{4, 5, \ldots, 32\}$ on a single plot (Figure 7a).

We have excluded the full-sensing version of 12 O'CLOCK since its results are similar to the adaptive version in most of the considered scenarios. In this section we discuss the combined boundaries in Figure 7a and the stable injection rates depicted in Figure 7b defined as minimal injection rates ρ for system size n required to make the value of avg-max measurement to exceed the constant value $\delta = 1024$.

Throughput of BACKOFF algorithms achieved in our simulations is similar to the results of simulations conducted by Hastad et al. [31]. The only differences are constants on the observed throughput. This difference between the two results can be explained by the following: we implemented more adversarial behavior instead of Poisson distribution, used 1 million instead of 10 millions iterations for experiment length, avg-max measurement instead of avg-avg (to better capture worst-case behavior), and finally we set-up a maximal window size limit to comply with real applications of BACKOFF. Specifically, the maximal window size limit improves the efficiency of exponential BACKOFF protocol in comparison to other versions of BACKOFF protocols in our context.

Non-adaptive protocols have the same Round-Robin implementation of selectors for system sizes $n \in \{4, 5, \ldots, 15, 17, 18\}$, because we were unable to generate better (n, ω)-selectors for $\omega \leq n/2$ required for INTERLEAVED-SELECTORS in those cases. It follows that their plots overlap. The best achieved stability bound is around $\rho = 0.6$ for system size $n = 4$, and it gradually decreases with the increasing system size (in a pace resembling hyperbola). On the other hand, we can observe an improvement of INTERLEAVED-SELECTORS over Round-Robin protocol for bigger systems: for some system sizes its stability range is even a few times bigger than the stability range of Round-Robin. The irregular shape of INTERLEAVED-SELECTORS stable injection rates in Figure 7b is caused by selectors being generated independently for each (larger) system size, which leaves a scope for further



(a) Queue size by injection rates $\rho \in [0, 1]$.

(b) Stable injection rates by system size.

(c) Relation between markers, colors and protocols.

Figure 7: Average round channel access and stable injection rates by system size.

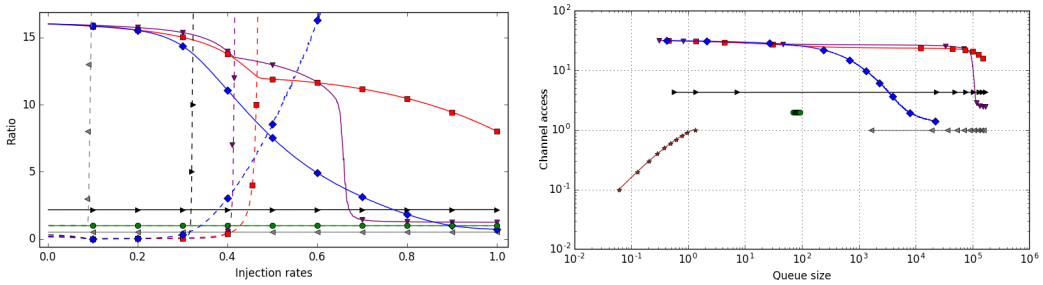
optimization of the quality of selectors.

Backoff protocols display dependency of queue stability on system size, and the following two phenomenons can be observed. First, the lower rank polynomial/function of BACKOFF protocol the wider extremes in stable injection rates it achieves for different system sizes, e.g. , $\rho \in [0.55, 0.7]$ for exponential version versus $\rho \in [0.45, 0.8]$ for square and $\rho \in [0.4, 0.85]$ for linear version, c.f., the values of ρ at the top boundaries of corresponding regions in Figure 7a. The second observation is that for smaller system sizes the protocols with lower rank function achieve higher stable injection rates while for larger systems (starting from some size specific for the considered functions) the tendency is opposite c.f., Figure 7b.

12 O'clock protocols have the least negative impact of an increase of system size over queue size stability, with 12 O'CLOCK adaptive protocol being a champion in this terms, c.f., Figure 7b. Note that the stable injection rates of 12 O'CLOCK full-sensing protocol improve with increasing system size.

6.3. Channel restraint and stability

In order not to discriminate randomized BACKOFF protocols, which may obtain large channel access peaks from time to time (unlike our deterministic protocols that ensure bounded channel access at any round), we count how many stations were switched-on on average (over rounds) to evaluate channel restraint. In Figure 8a we show the ratios of channel accesses and queue size of the considered protocols to the corresponding performances of 12 O'CLOCK adaptive protocol.



(a) Ratios of the average round channel accesses (solid lines) and the queue size (dotted lines) of a protocol to the corresponding performance of 12 O'CLOCK adaptive protocol.

(b) Average round channel accesses against queue sizes in logarithmic scale, with markers set every 0.1.

(c) Relation between markers, colors and protocols.

Figure 8: Average round channel access against queue sizes for injection rates $\rho \in [0, 1]$ and system size $n = 32$.

Observe that for BACKOFF protocols the total number of stations attempting to transmit or listen to the channel each round is close to the system size, when these protocols work within their stable boundaries. In contrary, the 12 O'CLOCK and non-adaptive protocols have a small number of switched on stations per round and this number is bounded by the respective constant.
In order to better illustrate bi-criteria comparison of protocols, we compare them with the STATE-AWARE protocol, which has full knowledge about all of the queues in the beginning of each round and transmits a packet from a station from the biggest queue. Note that this algorithm is a concept introduced to provide a reference to the performance of other protocols. It uses itself a global knowledge that is not accessible to the stations in the MAC model.
This protocol models close-to-optimal queues and channel access for given injection rates. Figure 8b presents our results in logarithmic scale: more efficient protocols in restraint-queue dimensions are closer to the STATE-AWARE protocol, as it represents the best performance algorithms can achieve. This makes 12 O'CLOCK adaptive protocol our champion for all injection rates and 8-light INTERLEAVED-SELECTORS to be the second for injection rates lower than $\rho = 0.3$. (Full-sensing version of the 12 O'CLOCK protocol has been omitted from the graphs as it behaved similarly to its adaptive version in our experiment).

7. Conclusions and discussions

We have proposed the k-restrained model for multiple access channels and studied throughput and queue stability of deterministic contention resolution protocols. We have developed protocols with proven constant upper bound on channel restraint and throughputs 1 and $1 - 1/n$, respectively for adaptive and full-sensing classes of protocols. k-light Selectors non-adaptive algorithm, though achieving smaller throughput, is provably almost optimal in the its restricted class of protocols. It was achieved thanks to a newly developed combinatorial tool, k-light selector. Proving accurate upper bounds on queue sizes of these protocols is an interesting open problem, as well as their further improvements by modifications of these protocols.
Simulations have shown preliminary evidence of performance in case of an adversary imposing asymmetrical queue load to the system (comparing to pure stochastic models). Backoff protocols were studied as the most commonly used contention-resolution approach. Experiments have repeated [31] results in regard of tendencies, with some differences in actual values of measurement, most likely caused by few differences in implementation details. Backoff protocols have shown limited throughput, stability, and inability to have small queues and channel restraint at the same time. 12 O'clock protocols, on the other hand, have shown excellent stability combined with low channel restraint.

References

[1] E. Hradovich, M. Klonowski, D. R. Kowalski, Contention resolution on a restrained channel, To appear in 26th IEEE ICPADS (2020).

[2] B. S. Chlebus, D. R. Kowalski, M. A. Rokicki, Adversarial queuing on the multiple access channel, ACM Transactions on Algorithms (TALG) 8 (2012) 1–31.

[3] B. S. Chlebus, D. R. Kowalski, M. A. Rokicki, Maximum throughput of multiple access channels in adversarial environments, Distributed Computing 22 (2009) 93–116. URL: http://dx.doi.org/10.1007/s00446-009-0086-4. doi:10.1007/s00446-009-0086-4.

[4] M. A. Bender, M. Farach-Colton, S. He, B. C. Kuszmaul, C. E. Leiserson, Adversarial contention resolution for simple channels, in:

SPAA, 2005, 2005, pp. 325–332. URL: http://doi.acm.org/10.1145/1073970.1074023. doi:10.1145/1073970.1074023.

[5] B. S. Chlebus, D. R. Kowalski, M. A. Rokicki, Adversarial queuing on the multiple access channel, ACM Trans. Algorithms 8 (2012) 5:1–5:31. URL: http://doi.acm.org/10.1145/2071379.2071384. doi:10.1145/2071379.2071384.

[6] M. A. Bender, J. T. Fineman, S. Gilbert, M. Young, How to scale exponential backoff: Constant throughput, polylog access attempts, and robustness, in: Proceedings of the Twenty-seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '16, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2016, pp. 636–654. URL: http://dl.acm.org/citation.cfm?id=2884435.2884482.

[7] L. Anantharamu, B. S. Chlebus, M. A. Rokicki, Adversarial multiple access channels with individual injection rates, Theory Comput. Syst. 61 (2017) 820–850. URL: https://doi.org/10.1007/s00224-016-9725-x. doi:10.1007/s00224-016-9725-x.

[8] L. Anantharamu, B. S. Chlebus, D. R. Kowalski, M. A. Rokicki, Medium access control for adversarial channels with jamming, in: A. Kosowski, M. Yamashita (Eds.), Structural Information and Communication Complexity, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 89–10.

[9] M. Bienkowski, T. Jurdzinski, M. Korzeniowski, D. R. Kowalski, Distributed online and stochastic queueing on a multiple access channel, ACM Trans. Algorithms 14 (2018). URL: https://doi.org/10.1145/3182396. doi:10.1145/3182396.

[10] B. S. Chlebus, V. Cholvi, D. R. Kowalski, Universal stability in multi-hop radio networks, Journal of Computer and System Sciences 114 (2020) 48–64. URL: https://www.sciencedirect.com/science/article/pii/S0022000020300544. doi:https://doi.org/10.1016/j.jcss.2020.05.009.

[11] B. S. Chlebus, E. Hradovich, T. Jurdziński, M. Klonowski, D. R. Kowalski, Energy efficient adversarial routing in shared channels, in: The 31st ACM Symposium on Parallelism in Algorithms and Architectures, SPAA '19, ACM, New York, NY, USA, 2019, pp. 191–200.

URL: http://doi.acm.org/10.1145/3323165.3323190. doi:10.1145/3323165.3323190.

[12] G. De Marco, G. Stachowiak, Asynchronous shared channel, in: E. M. Schiller, A. A. Schwarzmann (Eds.), Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2017, Washington, DC, USA, July 25-27, 2017, ACM, 2017, pp. 391–400. URL: https://doi.org/10.1145/3087801.3087831. doi:10.1145/3087801.3087831.

[13] Y.-J. Chang, T. Kopelowitz, S. Pettie, R. Wang, W. Zhan, Exponential separations in the energy complexity of leader election, in: Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, ACM, New York, NY, USA, 2017, pp. 771–783. URL: http://doi.acm.org/10.1145/3055399.3055481. doi:10.1145/3055399.3055481.

[14] T. Jurdzinski, M. Kutylowski, J. Zatopianski, Efficient algorithms for leader election in radio networks, in: PODC 2002, 2002, pp. 51–57. URL: http://doi.acm.org/10.1145/571825.571833. doi:10.1145/571825.571833.

[15] T. Jurdzinski, M. Kutylowski, J. Zatopianski, Weak communication in single-hop radio networks: adjusting algorithms to industrial standards, Concurrency and Computation: Practice and Experience 15 (2003) 1117–1131. URL: https://doi.org/10.1002/cpe.783. doi:10.1002/cpe.783.

[16] K. Nakano, S. Olariu, Randomized initialization protocols for ad hoc networks, IEEE Transactions on Parallel and Distributed Systems 11 (2000) 749–759. doi:10.1109/71.877833.

[17] P. Indyk, Explicit constructions of selectors and related combinatorial structures, with applications, in: Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '02, Society for Industrial and Applied Mathematics, USA, 2002, p. 697–704.

[18] B. S. Chlebus, D. R. Kowalski, Almost optimal explicit selectors, in: FCT, 2005, pp. 270–280.

[19] V. Cholvi, P. Garncarek, T. Jurdziński, D. R. Kowalski, Optimal packet-oblivious stable routing in multi-hop wireless networks, in: A. W. Richa,

C. Scheideler (Eds.), Structural Information and Communication Complexity, Springer International Publishing, Cham, 2020, pp. 165–182.

[20] S. Albers, Energy-efficient algorithms, Communications of the ACM 53 (2010) 86–96.

[21] S. Irani, K. Pruhs, Algorithmic problems in power management, ACM SIGACT News 36 (2005) 63–76.

[22] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, S. Wright, Power awareness in network design and routing, in: Proceeding of the 27th IEEE Conference on Computer Communications (INFOCOM), 2008, pp. 457–465.

[23] M. Andrews, A. F. Anta, L. Zhang, W. Zhao, Routing and scheduling for energy and delay minimization in the powerdown model, Networks 61 (2013) 226–237.

[24] M. Andrews, A. Fernández, L. Zhang, W. Zhao, Routing for energy minimization in the speed scaling model, in: Proceedings of the 29th IEEE International Conference on Computer Communications (INFOCOM), 2010, pp. 2435–2443.

[25] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, D. Wetherall, Reducing network energy consumption via sleeping and rate-adaptation, in: Proceedings of the 5th USENIX Symposium on Networked Systems Design & Implementation (NSDI), 2008, pp. 323–336.

[26] P. Bergamo, A. Giovanardi, A. Travasoni, D. Maniezzo, G. Mazzini, M. Zorzi, Distributed power control for energy efficient routing in ad hoc networks, Wireless Networks 10 (2004) 29–42.

[27] M. Gupta, S. Singh, Dynamic Ethernet link shutdown for energy conservation on Ethernet links, in: Proceedings of IEEE International Conference on Communications (ICC), 2007, pp. 6156–6161.

[28] C. Gunaratne, K. Christensen, B. Nordman, S. Suen, Reducing the energy consumption of Ethernet with adaptive link rate (ALR), IEEE Transactions on Computers 57 (2008) 448 –461.

[29] A. Ogierman, A. Richa, C. Scheideler, S. Schmid, J. Zhang, Sade: competitive mac under adversarial sinr, Distributed Computing 31 (2018) 241–254. URL: https://doi.org/10.1007/s00446-017-0307-1. doi:10.1007/s00446-017-0307-1.

[30] J. T. Fineman, S. Gilbert, F. Kuhn, C. Newport, Contention resolution on a fading channel, in: Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing, PODC '16, ACM, New York, NY, USA, 2016, pp. 155–164. doi:10.1145/2933057.2933091.

[31] J. Hästad, T. Leighton, B. Rogoff, Analysis of backoff protocols for multiple access channels, SIAM Journal on Computing 25 (1996) 740–774. URL: https://doi.org/10.1137/S0097539792233828. doi:10.1137/S0097539792233828. arXiv:https://doi.org/10.1137/S0097539792233828.

[32] P. Garncarek, T. Jurdzinski, D. R. Kowalski, Local Queuing Under Contention, in: U. Schmid, J. Widder (Eds.), 32nd International Symposium on Distributed Computing (DISC 2018), volume 121 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2018, pp. 28:1–28:18. URL: http://drops.dagstuhl.de/opus/volltexte/2018/9817. doi:10.4230/LIPIcs.DISC.2018.28.

[33] M. Chrobak, L. Gasieniec, W. Rytter, Fast broadcasting and gossiping in radio networks, Journal of Algorithms 43 (2002) 177 – 189. URL: http://www.sciencedirect.com/science/article/pii/S0196677402000044. doi:https://doi.org/10.1016/S0196-6774(02)00004-4.

[34] A. Ta-Shma, C. Umans, D. Zuckerman, Loss-less condensers, unbalanced expanders, and extractors, in: J. S. Vitter, P. G. Spirakis, M. Yannakakis (Eds.), Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece, ACM, 2001, pp. 143–152. URL: https://doi.org/10.1145/380752.380790. doi:10.1145/380752.380790.

[35] W. H. Kautz, R. C. Singleton, Nonrandom binary superimposed codes, IEEE Trans. Inf. Theory 10 (1964) 363–377. URL: https://doi.org/10.1109/TIT.1964.1053689. doi:10.1109/TIT.1964.1053689.

[36] E. Hradovich, Multiple access channel simulations, 2022. URL: https://github.com/ilkadi/Multiple-Access-Channel-Simulations. doi:10.5281/zenodo.6666530.