Ilya Hradovich

Efficient communication algorithms in shared channels with adversary

PhD dissertation

Supervisor
prof. dr hab. Marek Klonowski, Wrocław University of Science and Technology

Co-Supervisor
prof. dr Dariusz R. Kowalski, Augusta University and SWPS University

Institute of Computer Science
Polish Academy of Sciences

Warszawa 2021

Author's declaration:
I hereby declare that this dissertation is my own work.

August 27, 2021 .....

Ilya Hradovich

Supervisor's declaration:
The dissertation is ready to be reviewed

August 27, 2021 .....

prof. dr hab. Marek Klonowski

Co-Supervisor's declaration:
The dissertation is ready to be reviewed

August 27, 2021 .....

prof. dr Dariusz R. Kowalski

Contents

Chapter 1

Abstract

The fundamental problem of accessing a common resource by multiple actors is faced by many distributed systems, including processor transactional memory, wire and radio networks communication medium, and access to a shared resource on machines or data-centers. In many systems when more than one device attempts to access the common resource simultaneously, a *collision* occurs and in effect no device can use it.

Multiple-access channel (MAC) is a well-established model reflecting the key algorithmic challenges of such systems. In this model, 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. Usually we consider the problem of keeping the system *stable*, while the packets to be transmitted can be injected into devices' buffers in an arbitrary way. That is, the total number of packets kept in the buffers needs to be limited even in an infinite execution.

In this dissertation we expand the classical MAC model by introducing channel *restraint*, understood as a bound on the number of stations that can be switched on simultaneously. Apart from proving bounds this restraint puts on several classical protocol classes, we design and analyse optimal and near-optimal algorithms functioning within this restraint as well as implement simulations for those algorithms.

In this dissertation we further expand the model by introducing routing ability for algorithms on MAC. Combined with the channel restraint, this model substantially changes the way algorithms on MAC can operate. We provide algorithms as well as bounds on abilities of protocol classes to achieve certain levels of stability or packet latency.

The third original contribution presented here is a novel adversarial

average-case analysis method to study and compare algorithms performance. We demonstrate how such approach can be used for analysis of behaviour of some well-known algorithms. We also prove some dependencies between average-case and the popular worst-case analysis, including a counterpart of Little's Law, in the context of adversarial packet arrival.

Finally we introduce a new taxonomy for the consider classes of models and algorithms that covers and unifies a very wide spectrum of similar settings considered in a vast related literature.

Keywords: multiple-access channel, adversarial packet injection, parallel queuing, routing, channel restraint, latency, throughput, stability

Chapter 2

Streszczenie

Problem dostępu do wspólnego zasobu przez wiele urządzeń jest podstawowym zagadnieniem w systemach rozproszonych, takich jak transakcyjna pamięć procesora, przewodowa i bezprzewodowa komunikacja w sieciach oraz dostęp do współdzielonego zasobu maszyn w centrach danych.

Kanał wielodostępowy (MAC) to jeden z podstawowych modeli badanych w dziedzinie obliczeń rozproszonych. Jest on uważany za dobre przybliżenie wielu rzeczywistych systemów a jednocześnie umożliwia przeprowadzenie formalnej analizy bardzo licznych zagadnień algorytmicznych.

W rozważanym modelu stacje nadają pakiety do współdzielonego kanału komunikacyjnego w odrębnych odcinkach czasowych zwanych *rundami*. Ze względu na ograniczenia kanału, co najwyżej jedna udana transmisja może wystąpić w pojedynczej rundzie. W takim przypadku wszystkie stacje otrzymują nadaną wiadomość. Koncentrujemy się głównie na zagadnieniach, w których *adwersarz* wskazuje, do których stacji mają zostać dodane pakiety. Celem jest dostarczenie tych pakietów do wszystkich (lub wskazanych) urządzeń za pomocą tak ograniczonego kanału komunikacyjnego.

W niniejszej rozprawie zaprezentowane zostały oryginalne (w większości już opublikowane w naszych pracach) rozszerzenia klasycznego modelu MAC wraz z licznymi powiązanymi wynikami, które ich dotyczą. Znajdziemy w rozprawie zarówno algorytmy komunikacyjne wraz z ich formalną analizą, jak też wyniki badań dotyczące ograniczeń dla pewnych naturalnych klas problemów (granice dolne).

W rozprawie zaprezentowane zostały wyniki dla modelu, gdzie została ograniczona liczba *aktywnych* stacji. Oznacza to, że co najwyżej k stacji może w jednej rundzie być aktywnych (słuchać lub nadawać). Dla modelu tego, motywowanego głównie ograniczeniami energetycznymi, wykazano

ograniczenia dolne dla różnych klas protokołów komunikacyjnych i skonstruowano optymalne (lub bliskie optymalnym) protokoły.

Drugą grupą wyników stanowiących oryginalny wkład w niniejszą rozprawę jest naturalne rozszerzenie modelu pozwalające na przesyłanie pakietów wykorzystując stacje pośrednie. Wraz z opisanym wcześniej ograniczeniem jednoczesnej liczby aktywnych stacji tak wzbogacony model prowadzi do konstrukcji zupełnie innych algorytmów komunikacyjnych niż w modelu klasycznym.

Dla zaproponowanego modelu zaprezentowano i formalnie przeanalizowano liczne algorytmy pod względem ich poprawności, opóźnień w komunikacji oraz stabilności, czyli gwarancji, że liczba pakietów przetwarzanych jednocześnie w systemie jest ograniczona nawet w nieskończonym wykonaniu protokołu.

W rozprawie prezentujemy także analizę średniego przypadku wydajności algorytmów dla adwersarialnego modelu. Udowadniamy między innymi odpowiednik prawa Little'a.

Zaznaczmy, że większość wyników teoretycznych została poparta licznymi symulacjami.

Pracę uzupełnia nowa taksonomia algorytmów, która unifikuje bardzo różnorodne klasyfikacje obecne we wcześniejszej literaturze. Zaznaczmy, że wyżej wspomniana taxonomia została także opublikowana jako oryginalny wynik, w naszej wcześniejszej publikacji.

Chapter 3

Introduction

Networks of different kinds and purposes commonly contain areas of contention between different actors over the access to common medium. The medium constrains a system by collisions or denial of service, when more than one device attempts to use it simultaneously. In practice, channel access is constrained by physical factors, such as power, energy or availability. First, the "energy" spent by devices during such unsuccessful-for-most attempts is usually wasted. Second, for the case of multi-hop radio communication, too many attempts to transmit by neighbors may not only cause a collision in the considered node, but also in nodes of further distance. Third example, hardware systems are designed with a spike (maximal) power use in mind to prevent meltdown or blackout. The above examples have led us to an investigation of restrained-channels, as a natural extension of the classical shared-channel communication model with n devices (stations) attached to a single communication medium.

In our study we address those practical challenges via multiple-access channel (MAC) model. In this model time is divided into discrete intervals called *rounds*. In the model terminology, the shared medium and devices (actors) attached to it are referred to as *channel* and *stations*, respectively. Each station has a buffer of potentially unlimited size to store the incoming packets. We say that incoming packets are *injected* into the stations buffers. We call the ratio of the total number of packets injected into the system to the number of the passed rounds an *injection rate*.

Due to constraints of the channel, at most one station can successfully transmit a single packet during one round, and an additional restraint limits the number of simultaneous activities (transmissions or listenings) on the channel. Formally, there is an upper bound on the number of stations attached to the channel that can be switched on simultaneously, which

is interpreted as a cap on the channel use. Stations that are switched off cannot transmit nor receive packets from the channel, but they can have packets injected into them. We focus on the dynamic scenario, when packets are injected continually. The primary goal is to design an algorithm that guarantees *stability*, that is, a property that the sizes of queues in buffers. Clearly, the stability depends on the number of packets that can be added to the system (modeled as the *injection rate* of the adversary). The aim of the algorithmic effort is to find the highest possible injection rate ρ for with the system is stable. Note that ρ that guarantees stability is called usually *throughput*. We need to find the highest possible throughput.

Another important aim is to minimize the time packets spend in stations' queues, known as *latency*.

As another natural extension of the basic model presented in this dissertation we study dynamic routing on multiple access channels when packets are injected continually. In this approach, algorithms are allowed to use other stations of the system as the relay for packet re-transmission. A packet can be repeatedly handed over among the stations such that it hops through a sequence of stations in a store-and-forward manner until eventually it is heard by its destination station, which consumes the packet. We consider classes of algorithms defined by system restrictions for both restrained multiple access channels and restrained multiple access channels with routing models. These restricted algorithms may, for example, not use relay stations, or not use control bits in packets, or have the on-off status for each station scheduled in advance.

While doing our research, we have discovered ambiguity and parallel definitions of similar concepts used in papers even from a very close research fields causing difficulties in protocol classification, analysis and comparison of results. To mitigate it, our study contributes a generalized way of definition of multiple access channel algorithms. In this work we extract the common part of previous studies on parallel queuing with contention, and propose well-defined and unified criteria for classification of MAC-type scheduling algorithms. As there is a natural difference in performance evaluation between probabilistic and deterministic protocols, the worst case analysis commonly used for deterministic protocols does not serve well as the metric when applied to probabilistic protocols. This has been reflected in the extensive simulations presented in this dissertation as well as the assumed methodology while comparing various classes of protocols.

In order to better compare results in this area with the ones obtained for the related model of *stochastic* packet arrival, we propose an average-case

method of measurement and show some properties and results for this measure. We further connect the two areas by adopting the Little's Law for stochastic packet arrivals to the adversarial packet injection model. It describes the relationship between the system average queue size, latency and adversary injection rates.

Thesis Structure

In the 1st Chapter we discuss and present the algorithmic problems of multiple access channel (MAC), including the related studies in the field and model definition. The 2nd Chapter presents our unifying taxonomy for MAC protocols, linking channel and station capabilities to information used by particular protocol classes. Next, in the 3rd Chapter, we design and analyse algorithms for the restrained multiple access channel under the classical assumption of MAC model, where a packet has no specific destination station but needs to be "heard" on the channel. We simulate the designed algorithms and compare their performance to the classical Backoff protocols.

In Chapter 4, we design and study protocols for restrained multiple access channels under an additional assumption, in contrast to the above, that each packet needs to be delivered to a particular station. Chapter 5 presents a counterpart of Little's Law for adversarial queueing, together with the method of average case analysis for discrete multiple access channel algorithms run against an adversary. We apply this method to several well-known algorithms, showing how particular adversary strategies against those algorithms relate to the worst-case estimates. We conclude the dissertation in a brief summary in Chapter 6.

Majority of the content of this thesis first appeared in international conferences and journals in the form of papers. The author of the thesis was supported by Polish National Science Center (NCN) Grants:

- UMO-2015/17/B/ST6/01897
- UMO-2017/25/B/ST6/02553

We provide a short description of the published papers below, together with the general scope of the thesis author's contribution to those papers.

1. [HKK21a] "New View on Adversarial Queuing on MAC", joint work with Marek Klonowski and Dariusz R. Kowalski, appeared in IEEE

Communication Letters 2021. In this paper we proposed the comprehensive taxonomy of MAC protocols, modify and proof the Little's Law for adversarial queueing, define and apply the method for average case analysis of MAC algorithms in adversarial setting. The author of this thesis formulated the principles of the taxonomy together with the initial classification planes, proof of the Little's Law for adversarial queueing and initial adversary strategies for the average case analysis.

2. [HKK20] "Contention resolution on a restrained channel", joint work with Marek Klonowski and Dariusz R. Kowalski, appeared in IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS 2020). The authors studied how the introduction of restrained access control to MAC algorithms would affect class performance. The author of this thesis transformed two algorithms known in the literature to the restrained channel setting, achieving highest possible (theoretically proven) throughput and proved their correctness; he also created simulations to compare the efficiency of the studied protocols.

   Let us stress that simulations presented here are substantially extended comparing to the conference version.

3. [CHJ+19] "Energy efficient adversarial routing in shared channels", joint work with Bogdan S. Chlebus, Tomasz Jurdziński, Marek Klonowski and Dariusz R. Kowalski. [CHJ+19] appeared in Proceedings of the 31th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2019). In this paper we studied restrained multiple access channels together with the requirement on the packet to have one of the channel's stations to be its destination. The author of the current thesis contributed to the construction and analysis of the algorithm achieving the highest possible throughput, together with its analysis.

Apart from three published papers, this dissertation contains results included in two unpublished yet manuscripts (partially overlapping with the previous papers).

1. [HKK21b] "Restrained medium access control on adversarial shared channels", joint work with Marek Klonowski and Dariusz R. Kowalski. It is the extension of the paper above [HKK20]. The authors

provide algorithms and analysis for another class of protocols with rigid analysis of its correctness. Simulation results are provided in greater detail.

2. [CHJ+21] "Energy efficient adversarial routing in shared channels", joint work with Bogdan S. Chlebus, Tomasz Jurdziński, Marek Klonowski and Dariusz R. Kowalski. This is an extension of the paper [CHJ+19]. The authors provide algorithms putting higher emphasis on latency as the performance metric, as well as provide full proofs for all of the theoretical results.

Chapter 4

Algorithmic problems of MAC

We consider multiple access channel as a model of communication. In this model, 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.

In this chapter we describe in more detail the assumptions of the model, together with a thorough description of related work.

4.1 Model formalization

System. We follow the classical model of a shared channel, e.g., [CKR09, HLR96]. In this model we distinguish n stations attached to a transmission medium, called a *multiple-access channel (MAC)*, working according to the following rules:

- a packet transmitted by a station reaches all the stations instantaneously;
- 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.

	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 4.1: 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.

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 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 4.1.

Collision detection. Optionally, algorithms can rely on 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 4.2.

[1] In the basic model we assume that all stations are switched-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.

	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 4.2: 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.

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 any single round, the maximum number of packets that the adversary can inject is $\beta + \rho$.

This adversarial model of packet injection is called *leaky bucket*; it was used before to model traffic in shared channels, in particular in [CKR12, CKR09].

We introduce *injection pace* ν: the number of packets injected by the adversary during any prefix of P rounds is *at least* $\nu|P|$, were $|P|$ stands for the number of rounds in prefix P. Note that in previous work it was implicitly assumed that $\nu = 0$. We follow this assumption, and in all of the algorithms and analysis $\nu = 0$ unless it is stated otherwise.

Channel restraint. In our paper [HKK20] 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[†].

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 execution. 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.

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

We say that 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 restrain k and injection rates ρ is correct, if queues of all stations stay bounded at all times independently from adversary strategy and for any round the number of switched-on stations is at most k.

4.2 Previous and related work

OSI model. Open Systems Interconnection (OSI) model was created to allow the constitution of heterogeneous computer networks [RLNJ06]. The model adopted a layered architecture comprising the layers called: Physical, Data Link, Network, Transport, Session, Presentation, and Application. Each layer was tasked with a specific responsibilities and therefore associated algorithmic challenges.

The current work relates to the Data Link layer of the OSI model. This layer is responsible for moving information between multiple devices within the same logical network based on physical device addressing [DZ83].

Under the existing standards [IEEE802], it consists of two sub-layers. Medium access control (MAC) is the lower of the two sub-levels, responsible for the control of the hardware interacting with the communication medium. Logical link control (LLC) is the upper of the two layers, serving as the interface between MAC and the higher Network level of communication. Network layer provides the information to LLC in the form of frames, LLC directs the data to MAC, which in its turn controls Physical layer to transmit it to the medium. Reception of the information from the medium is performed in the opposite order.

Medium Access Control. MAC sub-layer primary functions defined by existing standards [IEEE802] are: control of the access to the communication medium, transmission error protection, frame recognition, station addressing.

The MAC sub-layer is further standardised by the area of application: bridging (mesh) networks [IEEE802.1D], Ethernet (local wired networks) [IEEE802.3], wireless local area networks [IEEE802.11], wireless personal area networks [IEEE802.15], broadband wireless metropolitan area networks [IEEE802.16] and wireless regional area networks [IEEE802.22].

In practice however, the research of medium access control algorithms is driven by the challenges of physical applications. Recently, MAC algorithms coexistence in overlapping wireless networks in the context of Internet of Things (IoT) was studied by Mendy et al. [MF21]. Underwater acoustic networks also grew to the field of its own, as survey of specialized MAC algorithms by Jiang et al. [Jia18] highlights. Industrial networks represent another field. Study by Shayo et al. [SMM20] organises the related knowledge for time division algorithms in this area. Protocols for power line communication (PLC) accounting for signal fluctuations in embeddded systems were studied by de Oliveira et al. [dVLR19]. Wireless sensor networks encompass limited compute and energy capacity of devices and are thoroughly studied on its own account. Specifics of the field were outlined in surveys by Demirkol et al. [DEA06] and by Yick et al. [YMG08]. Recent work by Dibaei et al. [DG20] reviews wireless MAC algorithms from the angle of full-duplex wireless networks. Akylidiz et al. in [AWW05] surveys architectures and applications of the mesh networks, together with their theoretical network capacities and communication protocols.

Cellular, aerial and satellite networks use shared channels under the term of non-orthogonal multiple access, which can be explained as definitions variations between fields as well as their respective problem scopes. Those networks utilise a number of orthogonal communication channels, with each particular channel facing medium access control algorithmic challenges [TDL+20].

Aerial vehicle communication in the context of cellular networks hosted by unmanned vehicles was studied by Sohail et al. [SLW18]. Recent advances in multiple access for cellular 5G networks were reviewed by Linglong et al. [DWD+18]. Satellite multiple access communication as the component of cellular 5G networks, with the review of related architectures and trade-offs between availability, resource utilisation and spectrum efficiency was studied by Xiaojuan et al. [YAL+19].

Multiple Access Channel. As the variations of medium access control algorithms were multiplying over the time, it became apparent that a model strong enough – to incorporate all of the various applications, and simple enough – to be able to deploy formal mathematics techniques, was needed to work with algorithms across the fields. Multiple Access Channel is the model fulfilling this call.

While it is difficult to trace the exact origins of the model, we can say that as early as in 1976 Kasami et al. used it in their study [KL76] with settings resembling those used in the current work.

The model of multiple access channel was considered in the number of papers, including but not limited to [Gol00, Chl18, HLR96, CKR12]. The use of it splits mainly by the aspect of packet arrival: there are papers preferring adversarial (i.e., arbitrary) packet arrivals [ACR17, AC15, ACKR19, CKR09, BCF+09, BFCH+05], as well as those focusing on the stochastic ones [GJKP00, HLR96, BJKK18].

As medium access control and multiple access channel abbreviation (MAC) overlap, please note that when we use it in this paper we refer to the latter and not to the former (with the exception of the previous paragraphs).

Apart from introducing new methods we extensively utilize many techniques used for constructing and analysis of algorithms. Below we recall most important ones.

Algorithm design methods. Several MAC-specific algorithm design patterns have emerged: conceptual token passed from station to station, withholding the channel and combinatorial selector structures [Gol00, Chl18, CKR12, Ind02].

Conceptual token technique utilises a distributed mechanism permitting only a single station to conduct transmissions on the channel. Rubin et al. employed this method on MAC in 1983 [RDM83] by designing algorithms utilising the concept of the token, but we believe that this method could of appear earlier. In more recent works by Anatharamu et al. [ACR09] and Chlebus et al. [CKR12, CCK20, CKR09] authors develop algorithms with the conceptual token technique, which can be seen as precursors to some of the algorithms discussed in the current thesis. Those works also utilise the concept of *round-robin* — passing the token from one station to another in the order of stations names, with the last station passing the token to the first station, hence creating a closed loop.

The technique of withholding the channel to the knowledge of author was first used by Chlebus et al. [CKR09]. This method permits stations satisfying certain defined by the algorithm conditions, to transmit its packets until the queue of a station is not empty or contains number of packets below some predefined limit.

Since then this technique has been widely used in the context of MAC, including but not limited to works by Chlebus et al. [CCK20, CKR12], Aldawsari et al. [AHJ20, ACK19] and Anatharamu et al. [ACKR19].

Explicit selectors and selective families were, to the knowledge of the author, first introduced by Komlós et al. [KG85]. Mathematical properties of those combinatorial structures were further researched by Indyk [Ind02] and Chlebus et al. [CK05].

Selectors have been used in the context of MAC since then, with a number of more recent works by Gorce et al. [GFK+17], De Bonis [De 19] and Grancarek et al. [GJK19] utilising it. The relation of universally strong selectors to packet-oblivious routing on multi-hop networks was studied by Cholvi et al. [CGJK20]. Randomised use of selectors as well as the definition of selectors was recently introduced by De Bonis et al. [DBV17].

Algorithm analysis methods. New algorithms require thorough analysis for correctness as well as measured performance metrics. Several analysis techniques have been developed over the years.

Little's Law in queueing theory describing the relationship between injection rates, queue size and latency, has been introduced by Little [Lit61] in the context of stochastic arrivals. The first attempt to obtain a similar formula for adversarial packet arrival was made by Chlebus et al. [CKR12], but it was made based on worst-case measurement and worked only for specific class of protocols.

Selectors and selective families mentioned in the previous paragraph has not been used only for constructing efficient algorithms. They also provide means to obtain lower bounds and limitations for some problems. Gargano et al. [GRV20] used superimposed codes to improve the known results in the area. De Bonis et al. utilised selectors to study bounds of full-duplex multi-packet reception on MAC [DV20].

Adversarial queueing. To the best of our knowledge, adversarial packet injections on multiple-access channel were considered for the first time in the context of contention resolution by Bender et al. in [BFCH+05] and Chlebus et al. [CKR06]. The authors of the latter paper introduced two types

of an adversary: window and leaky-bucket. Rosen in [Ros02] have established the relation between the two models and presented that they are equivalent in most of the settings except the extreme case of injection rate equal 1. We use the leaky-bucket type throughout this dissertation.

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 stations queues or packets latency is achievable include Bender et al. [BFGY16], Chlebus et al. [CKR09] and Anantharamu et al. [ACR17, ACKR11], 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 latency were studied by Bienkowski et al. [BJKK18]. Chlebus et al. in [CCK20] studied limitations of hearing on multiple access channel in relation to universal stability.

Energy and power constraints. Channel restraint introduced with the current work, can be intuitively compared in some aspects to the limit on power available to the system, which is novel to the model. On the other hand, there are some papers dealing with energy constraints in parallel environment, however for substantially different models/problems.

In the context of medium access control, several approaches were utilised in order to limit or control energy expenditure. Schemes for shutting down network interfaces for energy conservation when using the Ethernet were proposed by Gupta and Singh [GS07]. Gunaratne et al. [GCNS08] investigated policies to adaptively vary the link data rate in response to the demand imposed on the link rate as a means of reducing the energy consumption in Ethernet installations. Ogierman et al. [ORS+18] studied hardware-related challenges with a focus on adversarial jamming limited by the energy budget in medium access control protocols. Physical layer effects on a single hop fading signal were also studied by Fineman et al. [FGKN16], with particular attention to the spectrum reuse enabled by fading.

Algorithms managing energy usage were discussed in the surveys by Albers [Alb10] and Irani and Pruhs [IP05]. Routing subject to energy constraints have been by Chabarek et al. [CSB+08] and Andrews et al. [AAZZ13, AFZZ10]. Reducing network energy consumption via sleeping and rate-

adaptation was addressed by Nedevschi et al. [NPI+08]. Randomized queue-free throughput-based model for contention resolution with only bounds on transmission energy being known was studied by De Marco et al. [MS17].

Distributed power control improving the energy efficiency of routing algorithms in ad hoc networks has been proposed by Bergamo et al. [BGT+04]. Efficiency of broadcasting in ad-hoc wireless networks subject to the number of transmissions a node can perform, interpreted as energy constraint, was studied in papers by Gąsieniec et al. [GKK+08], Kantor et al. [KP16] and Karmakar et al. [KKPS17].

Jurdziński et al. [JKZ02] studied the problem of counting the number of active nodes in a single-hop radio network with the goal to simultaneously optimize the running time and the energy spent by each node, which is understood as the length of time interval when a node is awake. Klonowski et al. [KKZ12] considered energy-efficient ways to alert a single hop network of weak devices.

Kardas et al. [KKP13] studied energy-efficient leader election in single-hop radio networks. Chang et al. [CKP+17] studied the energy complexity of leader election and approximate counting in models of wireless networks.

Chang et al. [CDH+18] as well as Klonowski and Pająk [KP18] studied trade-offs between the time and energy for broadcasting in a multihop radio networks. Herlich and Karl [HK11] investigated saving power in mobile access networks when base stations cooperate to be active or passive in extending their range.

Chapter 5

The Taxonomy of MAC algorithms

Rapid progress in design of new algorithms, while welcomed, may naturally create ambiguity by introducing parallel definitions of similar concepts, thus causing difficulties in protocols' taxonomy, analysis and comparison of results. And this is the case for the area of MAC, what we have discovered when we started our work: algorithms proposed for various types of network, i.e., infrastructure wLANs, Ad-hoc networks, sensor networks, etc. — were coming together with model alterations incompatible with previous research.

The chapter is written based on part of the authors work published in [HKK21a]. In this chapter we first present the classifications used in literature and we continue into the unifying taxonomy developed to improve the granularity and applicability of the terminology across different fields. We conclude the chapter with the list of known and proven in the current work facts linked to the proposed taxonomy.

5.1 Existing naming conventions

One of the main challenges of MAC algorithms classification is in the multitude of fields algorithms are applied in and those fields forming independent silos developing their own terminology over the time.

We first review the history of development for related practically applied medium access control protocols classification, as it provides us with the insight of what more abstract multiple access channel algorithms need to satisfy. Later we follow on the history of MAC and model variations.

Medium access control protocol classes. To the knowledge of the author, typical medium access control algorithms classification exists only for the wireless communication field. Other areas, such as internet of things (IoT) [AS19] or underwater acoustic cables [Jia18], partially relay to it and rely on the field knowledge. Wireless medium access control protocols are classified in literature by method the stations access to the channel. Therefore, algorithms are divided into three main categories: random access, slotted (scheduled) access and hybrid (mixed). Jurdak et al. [JLB04] analyze and classify 34 wireless ad-hoc medium access control protocols using this method. Ulah et al. [UAK+17] utilises this classification for medium access control algorithms on wireless body area networks. Sami et al. [SNK+16] use this method in their classification of medium access control strategies for cooperative communication in wireless networks.

Random access protocols do not schedule time and contend for channels, relying on access randomisation to resolve potential collisions. Recent works in the area include a stochastic geometry-based model for low-power wide-network, written by Beltramelli et al. [BMÖG21] and compressive random access in cellular networks, written by Choi [Cho20].

Slotted access protocols require time to be synchronised across the system, divide the time into slots and make stations to switch-on and exchange packets in the beginning of each slot. *Framed access* protocols are the sub-class of slotted-access, where slots are further organised into fixed-size groups called frames. This allows algorithms to schedule channel access in great detail so that collisions on the channel are prevented from happening in the first place. Most common time division medium access control algorithms were examined by Tambaval et al. [TNS+19] in the context of emerging vehicular ad-hoc networks.

Hybrid access protocols is the combination of slotted and random access protocols. By using both scheduling and randomisation, this class allows for higher flexibility when compared to pure slotted access protocols and better predictability when compared to pure random access class. One of the earlier works by Rana et al. [RLNJ06] used this approach to adopt time slots allocation to available bandwidth. More recent work by Chen et al. [CDGZ20] worked on hybrid scheduling in heterogeneous half- and full-duplex wireless networks. Sundararaj et al. [SMK18] developed a hybrid energy-efficient medium access control algorithm for heavy-load wireless networks.

Multiple access channel protocol classes. Contrary to the classification approach applied in medium access control field, multiple access

channel field types algorithms by capabilities of the channel and stations attached to it. This can be observed in th survey of multiple access channel algorithms written by Goldberg [Gol00]. Such an angle to classification allows researchers to abstract away specifics of the environment into the capabilities the algorithm would require in order to function in this environment. The survey have distinguished between the two main classes of algorithms: *full-sensing* [Cap79] and *acknowledgement-based* [GJKP04]. The former assumes that each station has access to the history of transmission attempts performed by all stations, while the latter assumes that each station has access only to the history of own transmission attempts. Acknowledgement based class had three more deviations: almost-acknowledgement-based – as the sub-class provided with information about the estimated system size [PS95]; age-based – as the sub-class utilising randomisation of transmission probability based on packets age [Kel85]; backoff – as the sub-class utilising probabilities associated with the number of stations failed transmission attempts [HLR96].

Later works have further distinguished a subclass of full-sensing algorithms. This sub-class was called *adaptive* and it had an additional capability to add a small number of control bits to transmitted packets [CKR09, MK10]. Collision detection mechanism was introduced into the model later as well [ACR17, CKR12].

One can observe that those taxonomy systems do not provide planes of capabilities on which all of the protocols can be compared. Historically, a newly designed protocol classified as some class will have some additional capability influencing its performance. And precisely because of this capability it could not be directly compared to other protocols of the class or to other classes. This is the problem we attempt to address in our taxonomy proposed below.

5.2 Unifying classification

The classification of algorithms we are about to discuss was first introduced in our paper [HKK21a]. We follow the approach of defining algorithms in relation to channel and station capabilities. However we increase the granularity of those capabilities, so that they can be seen as orthogonal to each other. We believe that such an approach over time can allow the community to use it across the fields.

5.2.1 Taxonomy

We list the most important characteristics of the environment and the protocols found in the literature, with respect to which they could be aware-type and oblivious-type. In this taxonomy algorithms are defined by capabilities they are aware of:

Synchronization:
Time-aware - stations know the global index of any round. *Time-oblivious* - stations know only local time, i.e., the number of rounds from the beginning of their current execution.

Collision Detection:
Collision-aware - stations know when collision occurs.
Collision-oblivious - stations cannot differentiate between the collision and silent round.

Queue Size:
Queue-aware - stations know the size of their queues.
Queue-oblivious - stations know if there is a pending packet in their queues, but not the size of its queue.

Transmitting identity:
Source-aware - stations may include to their transmissions transmitter identities and to read those values from packets heard on the channel.
Source-oblivious - stations do not have access to transmitter identity of the packet.

Destination identity:
Destination-aware - stations may include to their transmissions receiver identities and to read those values from packets heard on the channel.
Destination-oblivious - stations do not have access to receiver identity of the packet.

Control bits:
Control-aware - stations have read and write access to the protocol-defined packet control bits.
Control-oblivious - stations cannot include additional bits into their transmissions.

Memory use:
Memory-aware - stations have memory capacity to store the current state and/or its historical values.
Memory-oblivious - stations have memory capacity only to track their own state and/or history of states.

Acknowledgement:
Acknowledgement-aware - stations receive an acknowledgement of their successful transmission in the same round with that transmission.
Acknowledgement-oblivious - stations do not receive acknowledgement of successful transmission.

Channel restraint:
Restraint-aware - the channel has an upper bound k on the number of simultaneously accessing stations, $k < n$, where n is the size of the system.
Restraint-oblivious - the channel has no upper bound on the number of simultaneously accessing its stations.

Routing:
Routing-aware - stations can transmit packets in multiple hops.
Routing-oblivious - the channel allows only single hop transmissions.

Random source:
Random-aware - stations have the ability to produce pseudo-random numbers.
Random-oblivious - stations have the ability to produce pseudo-random bits.

Importantly, each capability can be considered independently (we assume that Transmitting and Destination Identities do not require Control Bits capability to store station addresses in packets).

In what follows, we assume that algorithms are oblivious with respect to all capabilities, unless specified otherwise.

Please note that the Routing capability applies only to the Destination-aware channels. For the Destination-oblivious channels, the packet is delivered once it is successfully transmitted – rendering the Routing capability obsolete.

5.2.2 Taxonomy mapping

With the proposed taxonomy, we are able to map existing in literature algorithm classes to defined above granular capabilities for both: multiple access channel and medium access control settings. In relation to studies [Gol00, CKR09], multiple-access channel classes of Adaptive, Full-Sensing and Acknowledgement based algorithms can be mapped to the proposed taxonomy as follows:

- *Adaptive*: Queue, Time, Source, Memory and Control aware;
- *Full-sensing*: Queue, Time, Source and Memory aware;

- *Full-sensing with collision detection*: Queue, Time, Source, Memory and Collision aware;
- *Acknowledgement-based*: Acknowledgement-aware.

In relation to works [BMÖG21, TNS+19, SMK18], medium access control classes can be mapped to the proposed taxonomy as follows:

- *Random-access*: Random-aware;
- *Slotted-access*: Time aware;
- *Hybrid*: Random and Time aware.

One can observe that medium access control classes are too generic and do not allow in their current form for the assessment of channel capabilities. However we can expect field interoperability improvements once such work is completed.

5.3 Bounds by channel capabilities

In this section we formulate new and re-formulate old results on impossibilities using the aforementioned taxonomy.

Known impossibilities. Below we state sample results on (worst case) adversarial stability of MAC algorithms obtained in previous research. We express known results using the new taxonomy.

Fact 1. [**CKR09**] *Queue-oblivious and Control-oblivious algorithms are not stable against adversaries with injection rates equal to 1.*

Fact 2. [**CKR12**] *Adaptive and Collision-sensing algorithms can be stable against adversaries with injection rates* $\rho = 1$ *and* $\rho < 1$ *respectively.*

New impossibilities. Part of the current thesis is dedicated to proving bounds and impossibilities for the changes associated with the changes we introduce to the model. Below we summarise the impossibilities related to the channel capabilities, together with references to their respective points of origin.

Fact 3. *[Lemma 9 in section 6.4] Time-oblivious and Restraint-aware algorithms are not stable against adversary with any* $\rho > 0$.

Fact 4. *[Theorem 3 in section 6.4] Acknowledgement-aware, Restraint-aware and Time-aware algorithms for system size* n *and channel restraint* k *are unstable against adversaries with injection rates greater than* $\min\{\frac{k}{n}, \frac{1}{3\log n}\}$.

Fact 5. *[Theorem 8 in section 7.4.2] Time-aware, Destination-aware and Restraint-aware algorithms for restraint* 2 *and a system size greater than or equal to* 3 *are not stable against adversaries with injection rate* 1.

Fact 6. *[Theorem 16 in section 7.8.3] Destination-aware, Acknowledgement-aware, Restraint-aware, Time-aware and Routing-oblivious algorithms for system size* n *and channel restraint* k *are unstable against adversaries with injection rates greater than* $\frac{k(k-1)}{n(n-1)}$.

Chapter 6

Restrained Multiple Access Channel

In this chapter we are investigating how introduction of the channel restraint k, understood as an upper bound on the number of active (listening or transmitting) stations per round, influences the throughput on that channel for different classical classes of algorithms (Adaptive, Fully-sensing, Acknowledgement-based).

Apart from algorithms design and their rigid formal analysis, a part of this chapter is devoted to experimental results proving the efficiency of the constructed protocols for realistic systems' sizes. We also show that our algorithms outperform back-off-type protocols both in terms of throughput efficiency and system stability (i.e., queue sizes) in the model with restraint. The results presented in this chapter can be considered as extensions of the results from our paper [HKK20].

6.1 A summary of the results

We construct optimal or nearly-optimal solutions for different classes of protocols studied in the literature: achieving throughput 1 for adaptive protocols, throughput 1 − (for any fixed) for full-sensing protocols, and throughput $\Theta(\frac{k}{n \log^2(n)})$ for acknowledgement based protocols (the latter result is complemented by the upper bound $\min\{\frac{k}{n}, \frac{1}{3 \log n}\}$ for this class of protocols). All the performance bounds of algorithms are presented in Table 6.1.

In this chapter we follow algorithm classification from our original paper [HKK20]. For the details of how Adaptive, Full-sensing and Acknowledgement-based classes map to the newly introduced taxonomy, see Chapter 5.

Algorithm	Sec.	Injection	Queues	Restraint	Class
12 O'clock-ad	6.2	$\rho = 1$	$O(n^2 + \beta)$	2	Adaptive
12 O'clock-fs	6.3	$\rho < 1$	$O(n^2 + \beta)$	3	Full-sensing with CD
k-Light IS	6.4	$\rho < \Theta\left(\frac{k}{n \log^2 n}\right)$		k	Acknowledgement-based
Impossibility	6.4	$\rho > \min\{\frac{k}{n}, \frac{1}{3 \log n}\}$	Stable	k	Acknowledgement-based

Table 6.1: 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.

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 algorithms without decreasing throughput of the system (i.e., comparing to the respective families of protocols without channel restraint described e.g. in [CKR09]). 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., acknowledgement based 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 and $O(n \log^2 n)$ for the acknowledgment-based one.

Let us stress that our acknowledgement-based algorithm uses a newly introduced combinatorial structure, called *k-light selector*, which we thoroughly study for its own independent interest.

6.2 Adaptive protocol 12-O'clock

6.2.1 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.

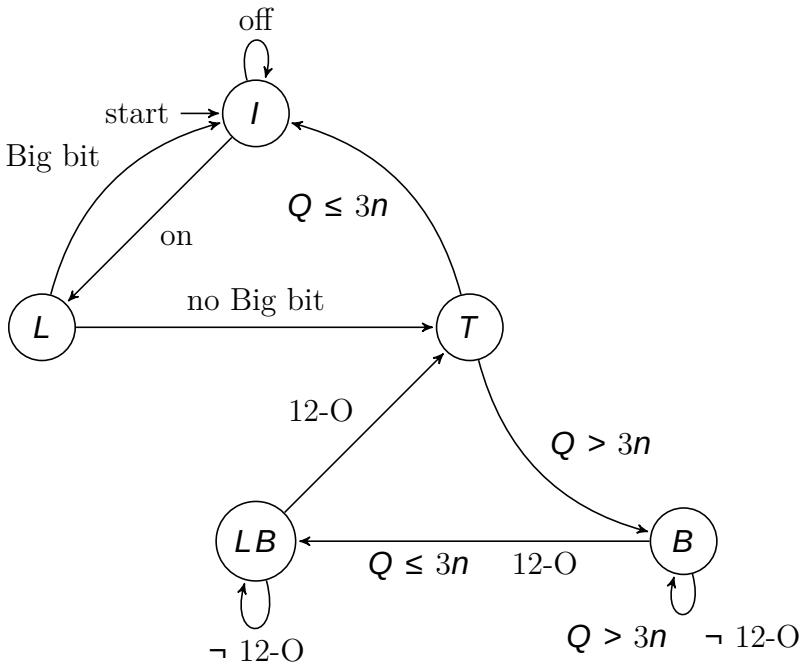


Figure 6.1: Finite state machine for a station in 12-O'clock adaptive algorithm. Station states are represented by nodes: I stays for Idle state; L for Listening state; T for Transmitting state; B for Big state; LB for Last-Big state. Note that from the system perspective there are two stations with the starting state being Transmitting and Listening respectively, otherwise the starting state is Idle as shown. Arrows represent checks performed by the distributed algorithm, where Q stands for station queue size, n for system size, *12-O* for the current round being the end-of-cycle round; *Big bit* represents a control bit set in packet by station in either Big or Last-Big states; *on* and *off* representing clock-based actions of switch-on and switch-off respectively. Note that it takes one round to execute a check of each arrow.

```
1  Procedure transmitToTheChannel()
2      switch s.state do
3          case Transmitting do
4              if s.queue > 3n then
5                  s.state := Big;
6                  s.transmit();
7              else
8                  if s.queue > 0 then
9                      s.transmit();
10                 s.state := Idle;
11         case Big do
12             if round = 0 mod n AND s.queue ≤ 3n then
13                 s.state := Last-Big;
14                 s.moveBigToFront(s.id);
15             s.transmit();
16         case Last-Big do
17             s.transmit();
18             if round = n - 1 mod n then
19                 s.state := Transmitting;
```

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

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. 6.1.

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:

- `transmitToTheChannel` (Algorithm 1) – executed in the transmission phase of the round;
- `listenToTheChannel` (Algorithm 2) – executed in the listening phase of the round.

Both procedures are executed by switched-on stations.

```
1  Procedure listenToTheChannel(channel)
2      switch s.state do
3          case Idle do
4              if shouldWakeUp() then
5                  s.state = Listening;
6          case Listening do
7              switch channel.state do
8                  case Silence do
9                      s.state = Transmitting;
10                 case Normal do
11                     s.state = Transmitting;
12                 case Big do
13                     s.state = Idle;
14                 case Last-Big do
15                     s.state = Idle;
16                     s.moveBigToFront(channel.id);
```

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

Methods. Procedures rely 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, 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 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.

6.2.2 Protocol correctness and performance analysis

Consider the total size of the queues in the beginning of the cycle. If the size is greater than $= n(3n-1)+1$ we say that 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 adopting 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 1. *Each cycle is of one of the above four types.*

Proof Algorithm's initialization conditions (Subsection 6.2.1) 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 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. *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 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 1. *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 $+n+\beta$ packets in the stations at the end of any cycle for dense interval threshold . Indeed, the biggest possible number of packets that the system can start a cycle with is equal to , 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 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 $+n$ 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 , 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 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, 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 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 $+(n-1)^2+n+\beta$ on system queues. Since only one of those stations has the right to transmit, collision never occurs and channel restrain is 2.

It should be noted 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 below.

That is, packets in the station queue should be stored in some sort of memory. To fairly compare different protocols - queue size needs to be taken into an account together with the size of the state.

6.3 Full-sensing version of protocol 12 O'clock

6.3.1 Protocol overview

The 12-O'clock full-sensing with collision detection protocol is an adaptation of the protocol described in Section 6.2 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 $\rho \leq \frac{n-1}{n}$.

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 6.2*.

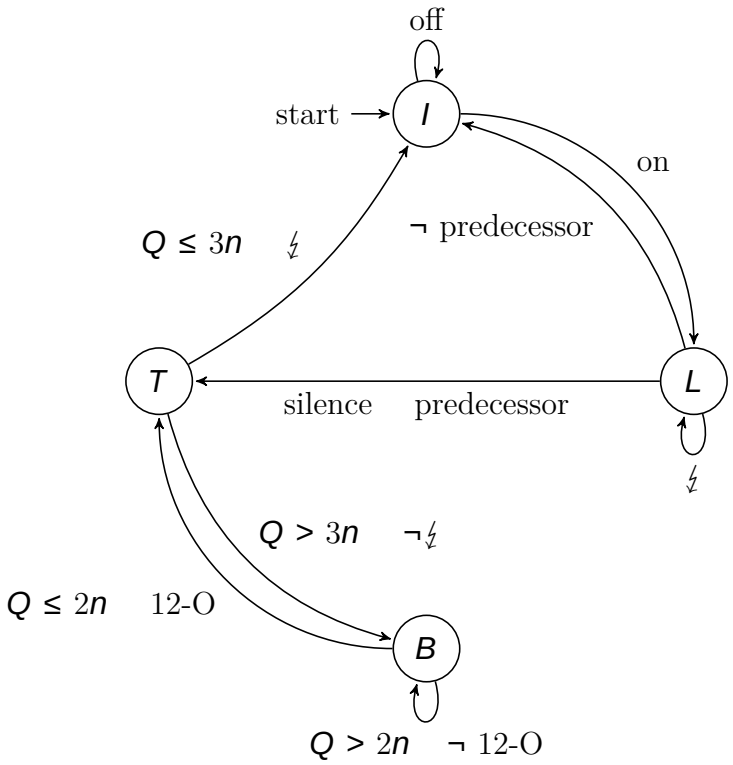


Figure 6.2: Finite state machine for a station in 12-O'clock full-sensing algorithm. Station states are represented by nodes: I stays for Idle state; L for Listening state; T for Transmitting state; B for Big state. Note that from the system perspective there are two stations with the starting state being Transmitting and Listening respectively, otherwise the starting state is Idle as shown. Arrows represent checks performed by the distributed algorithm, where Q stays for station queue size, n for system size, *12-O* for the current round being the end-of-cycle round; *silence* for no transmission heard on the channel, $\lightning$ for collision heard on the channel; *predecessor* stays for a packet transmitted by the predecessor of the station heard on the channel; *on* and *off* representing clock-based actions of switch-on and switch-off respectively. Note that it takes one round to execute a check of each arrow.

```
1 Procedure transmitToTheChannel()
2     switch s.state do
3         case Transmitting do
4             if s.queue > 0 then
5                 s.transmit();
6                 s.transmitted = true;
7         case Big do
8             s.transmit();
```

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

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:

- `transmitToTheChannel` (Algorithm 3) – executed in the transmission phase of the round;
- `listenToTheChannel` (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.

```
1  Procedure listenToTheChannel(channel)
2      switch s.state do
3          case Idle do
4              if shouldWakeUp() then
5                  s.state = Listening;
6          case Listening do
7              switch channel.state do
8                  case Silence do
9                      s.state = Transmitting;
10                 case Collision do
11                     s.state = Listening;
12                 case Transmission do
13                     if channel.transmitterId=predecessor.id then
14                         s.state = Transmitting;
15                     else
16                         s.state = Idle;
17                         s.moveBigToFront(channel.id);
18         case Transmitting do
19             switch channel.state do
20                 case Silence do
21                     s.state = Idle;
22                 case Collision do
23                     s.state = Idle;
24                     s.moveBigToFront(predecessor.id);
25                 case Transmission do
26                     if s.transmitted then
27                         if s.queue > 3n then
28                             s.state := Big;
29                         else
30                             s.state = Idle;
31                     else
32                         s.state = Idle;
33                         s.moveBigToFront(predecessor.id);
34         case Big do
35             if mod(round,n) = n-1 AND s.queue ≤ 2n then
36                 s.state := Transmitting;
37                 s.moveBigToFront(s.id);
```

Algorithm 4: 12 O'clock full-sensing algorithm, listening 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 $n - 2$ 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.

6.3.2 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 $= n(3n - 1) + 1$ 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 Last 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 adopting 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. *Each cycle is of one of the three above types.*

Proof The initial conditions of the algorithm specified in Subsection 6.3.1, 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 5. *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 (described in Section 4.2) 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 6. *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 7. *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 6). 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 2. *The 12 O'clock full-sensing protocol achieves throughput* $1 - \frac{1}{n}$ *on a channel with restrain of* 3 *and the maximum number of packets stored in a round is at most* $+ (n-1)^2 + n + \beta = O(n^2 + \beta)$.

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 slots.

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 $+ n + \beta$ 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 , 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 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 $+n+\beta$. indeed, as in the beginning of the preceding cycle the interval was sparse, the number of packets was not bigger than , 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 7 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 6. 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 $+(n-1)^2+n+\beta$ on the sum of the queue sizes in a round.

6.3.3 Stability bound improvement

It was proved in [CKR09] 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 8. *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 $= n((2+k)n - 1) + 1$, since the Big station definition has changed. Following the logic of the proof of Theorem 2, 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 $+ k(n-1)^2 + n + \beta = O(kn^2 + \beta)$.

6.4 Acknowledgment-based protocols

In this section we consider acknowledgment-based protocols in k-restrained model and present the results first published in [HKK20].

First we prove two limitations for this class of protocols. One of this limitations restricts the protocol class to use global-clock mechanism and it is followed as the basic requirement later through this section.

Secondly, we introduce a new combinatorial construction called *k-light selectors*. 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 $\Theta\left(\frac{k}{n \log^2(n)}\right)$.

6.4.1 Upper bound on throughput

Lemma 9. *There is no correct, k-restrained acknowledgment-based algorithm with channel restrain $k < n$ without a global-clock mechanism for any $\rho > 0$.*

Proof Assume that P is a correct k-restraint deterministic acknowledgment-based protocol without a global clock in the system of n stations. Then for each station s_i, there is a default starting sequence p_i, where i is the index of the station. Because P is correct, each p_i contains a first occurrence of transmission bit 1. Let t_i be the position of the first transmitting bit

in the sequence p_i. Because the system is not equipped in the global clock mechanism, stations' starting rounds are set by adversary. Let us say that s_i is the first switched-on round of station S_i. It follows that the first transmission of station i occurs at round $s_i + t_i$. In order to overload the system adversary follows the strategy: choose round e as $e = max\{t_1, \ldots, t_n\}$; start station S_i at round $s_i = e - t_i$. Then all $n > k$ stations transmit at round e and thus P has to overflow channel restrain k. In consequence P is not correct.

Theorem 3. *Acknowledgment-based algorithms in the k-restrained channel, $k < n$, cannot achieve throughput higher than* $\min\{\frac{k}{n}, \frac{1}{3 \log n}\}$.

Proof To prove the theorem, assume first that $\frac{k}{n} \leq \frac{1}{3 \log n}$. Consider a period of τ consequent rounds. Suppose, to a contradiction, that during τ rounds the adversary can inject $\tau \cdot k/n + 1$ packets. The channel restrain of k implies that at most k stations can be active and, therefore, during τ rounds there could be at most $\tau \cdot k$ activities in total. There are n stations in the system, hence, by pigeon-hole principle, there is a station allowed to transmit at most $\tau \cdot k/n$ packets during τ rounds. Acknowledgement-based protocols with global clock provide adversary with a power to know stations schedules in advance, as the adversary can calculate values of the protocol function for any round and for each station; hence, it can pick a station S, such that the number of scheduled switch-on rounds is minimal within the system. Once the station S is chosen, the adversary can inject $\tau \cdot k/n + 1$ packets into the queue of S. Queues of arbitrary length would be generated by iteration of the procedure, thus the system cannot be stable, which results in contradiction. This proves that ρ cannot exceed $\frac{k}{n}$. The second case when the minimum formula equals to $\frac{1}{3 \log n}$ follows directly from Thm. 5.1 in [CKR09].

6.4.2 k-light Selectors

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

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

Note that this definition is a special case of a *selective family* [CK05]. The intuition behind 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 S.

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

Theorem 4. *K-light (n, ω)-selector of size $m = O((\omega + n/k) \log n)$ exists.*

Proof We divide the proof of the formula

$$m = O((\omega + n/k) \log n) \tag{6.1}$$

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. [CGR02].

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, $S = (S_1, \ldots, S_m)$ is a random family. Let us consider any **fixed** sets X, Y $\{1, \ldots, n\}$, such that $\omega/4 \leq |X| \leq \omega$; $|Y| \leq \omega/4$ and a **random** S_i.

$$\Pr[S_i \text{ avoids } Y \text{ and hits } X] = \frac{\binom{|X|}{1}\binom{n-|X|-|Y|}{l-1}}{\binom{n}{l}} = |X| \cdot l \cdot \frac{(n-|X|-|Y|)^{\underline{l-1}}}{n^{\underline{l}}} =$$
$$\frac{|X| \cdot l}{n-l+1} \prod_{i=0}^{l-2} \frac{n-|X|-|Y|-i}{n-i} > \frac{\frac{\omega}{4} \cdot l}{n} \prod_{i=0}^{l-2} \frac{n-\frac{5}{4}\omega-i}{n-i} \geq$$
$$\frac{\frac{\omega}{4} \cdot l}{n} \left(1 - \frac{\frac{5}{4}\omega}{n-l+2}\right)^{l-1} \geq \frac{\frac{\omega}{4} \cdot \frac{n}{\omega}}{n} \left(1 - \frac{\frac{5}{4}\omega}{n/4}\right)^{l-1} \geq$$
$$\frac{1}{4}\exp(-5) = c > 0\,. \tag{6.2}$$

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:

$$\sum_{|X|=\frac{\omega}{2}} \binom{n}{|X|} \sum_{y=0} \binom{n}{|Y|} (1-c)^m \leq \omega^2 n^{2w} (1-c)^m \leq$$
$$n^{4\omega}(1-c)^m \leq e^{4\omega \ln n - m \ln(1-c)} < 1\,. \tag{6.3}$$

Note that the last inequality holds for some $m = O(\omega \log n)$. That is, for such m the random structure $S = (S_1, S_2, \ldots, S_n)$ 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 S and use it for the reminder of the proof.

Now we show that S is a (n, ω)-selector. Let us take any X such that $\omega/2 \leq |X| \leq \omega$ and $Y =$. By the property of 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 $Y = Y \quad \{r_1\}$. Since still $\omega/4 \leq |X| < \omega$ and $|Y| \leq \omega/4$ we can find $S_{i_2} \quad S$, 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 $Y = Y \quad \{r_2\}$. 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 (6.1) $O((n/k)\log n)$ for $\frac{n}{k} > \omega$, first we need to construct an $\frac{n}{\omega}$-light selector S of size $m = O(\omega \log n)$. Clearly, this is possible using the above construction. Then we need to partition each $S_i \quad S$ into $\frac{n}{k\omega}$ 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.

6.4.3 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 $(n, \ , d, \delta, \varepsilon)$-*disperser* if it has the following properties: $|V| = n$ and $|W| = d/\delta$; each $v \quad V$ has d neighbors; for each $A \quad V$ such that $|A| \geq \ $, the set of neighbors of A is of size at least $(1 - \varepsilon)|W|$. Ta-Shma, Umans and Zuckerman [TUZ01] showed how to construct, in time polynomial in n, an $(n, \ , d, \delta, \varepsilon)$-disperser for any $\leq n$, 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 [KS64] 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., [TUZ01]. Let $N_G(v)$ stay for the set of neighbors of node $v \quad V$ in graph G. Let $M = \{M_1, \ldots, M_a\}$ 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 [KS64]; 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 $S(n, \omega, k, c)$ 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 $v \quad V$ such that v is a neighbor of the x-th node in W and $v \quad M_y$; i.e., $F_{x \cdot a + y} = M_y \cap N_G(x)$. Next, we split every F_j into $|F_j|/k$ subsets S of size at most k each, and add them as elements of the selector $S(n, \omega, k, c)$. Note that each set S_i from $S(n, \omega, k, c)$ 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. *$S(n, \omega, k, c)$ is a k-light (n, ω)-selector of length*

$$m = O(\min\{n, (\omega + n/k) \text{ polylog } n\}) \tag{6.4}$$

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

Proof First we show that $S(n, \omega, k, c)$ 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 $S(n, \omega, k, c)$. 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 $S(n, \omega, k, c)$.

Claim. Every $w \in N_G(B)$ has more than $c\delta$ neighbors in A, where $c\delta$ is a disjunctness parameter of 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 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

$$F_{w \cdot a+y} \cap A = (M_y \cap N_G(w)) \cap A = M_y \cap (N_G(w) \cap A) = \{v\} \qquad (6.5)$$

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 $S(n, \omega, k, c)$, 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 $S_j \in S(n, \omega, k, c)$ 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

$$(1 - \varepsilon)|W| \cdot c\delta = (1 - \varepsilon)\Theta((\omega/4 + 1)d/\delta) \cdot c\delta > \omega d \,, \qquad (6.6)$$

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 $S(n, \omega, k, c)$ 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 M. Hence, the amortized number of children $S \in S(n, \omega, k, c)$ of a set F_j, parameter α, is at most

$$\frac{a \cdot (|V| \cdot d)}{k} \cdot \frac{1}{a|W|} + 1 \,. \qquad (6.7)$$

The length m of this selector is thus at most

$$\min\{n, a|W|\alpha\} = O\left(\min\left(n, \delta^2 \log^2 n \cdot \omega d/\delta + \frac{nd \cdot (c\delta)^2 \log^2 n}{k}\right)\right) = O\left(\min\left\{n, (\omega + n/k)\,\mathrm{polylog}\, n\right\}\right), \tag{6.8}$$

since $d = O(\mathrm{polylog}\, n)$ 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.

6.4.4 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 $S_1, \ldots, S_{\log(n)}$, where 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, $S_i = \{S_i^1, \ldots, S_i^{m_i}\}$. 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 in the t round 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. 6.3.

6.4.5 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 6. *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 $j \geq i$.*

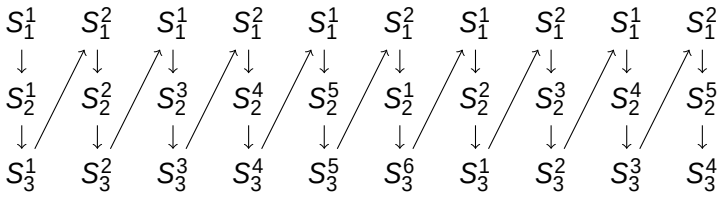


Figure 6.3: Interleaved Selectors: $A = \{S_1, S_2, S_3\}$, where $S_1 = \{S_1^1, S_1^2\}$, $S_2 = \{S_2^1, ..., S_2^5\}$ and $S_3 = \{S_3^1, ..., S_3^6\}$.

Proof Let us first consider a set $X_0 \quad \{1,\dots,n\}$ of stations such that $|X_0| = r$ and $2^{i-1} \le r < 2^i$. Let $S_i = \{S_i^1,\dots,S_i^{m_i}\}$ be a $(n, 2^i)$-selector and $S_{i+1} = \{S_i^1,\dots,S_i^{m_{i+1}}\}$ 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 $j \ge m_i$. In other words, in consecutive rounds transmit nonempty stations pointed by sets from S_i, then stations from S_{i+1}.

Lemma 10. *If less then $2^i/16$ different stations has transmitted during T rounds of the process then $|X_T| \ge \min\{r + 2^i/8, 2^{i+1}\}$.*

Proof Let $Y = \bigsqcup_{i=1}^{T} X_i \setminus X_0$ be the set of all stations filled by the adversary during the process. Let O be the set of stations that are transmitted during the process. Moreover, let $T(X)$ 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, $|T(X_0 \quad Y)| \le |O| + |Y|$. 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 $X_0 \quad Y$ stations are nonempty at the beginning.

Let us consider two cases. In the first we assume $|X_0 \quad Y| < 2^{i+2}$. In follows that $|T(X_0 \quad Y)| \ge 2^i/4$ because of the properties of selectors. Thus $2^i/4 \le |O| + |Y|$. We assumed however that $|O| < 2^i/16$, 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 $|X_0 \quad Y| > 2^{i+2}$ 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 S_i, S_{i+1} are allowed to transmit (see Fig 6.3). Following Lemma 6.3 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 6 with Theorem 4 we get:

Corollary 1. *The protocol achieves throughput* $\Theta\left(\frac{k}{n \log^2 n}\right)$ *on* k*-restrained channels.*

6.5 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.

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 6.4a-6.6b. 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.

6.5.1 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 [HLR96]. 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 $\omega : \omega = f(\omega)$. 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 [HLR96] 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.

Acknowledgment-based protocols. Round-Robin protocol allows any station i to transmit alone in rounds i modulo n. 8-light `InterleavedSelectors` 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.

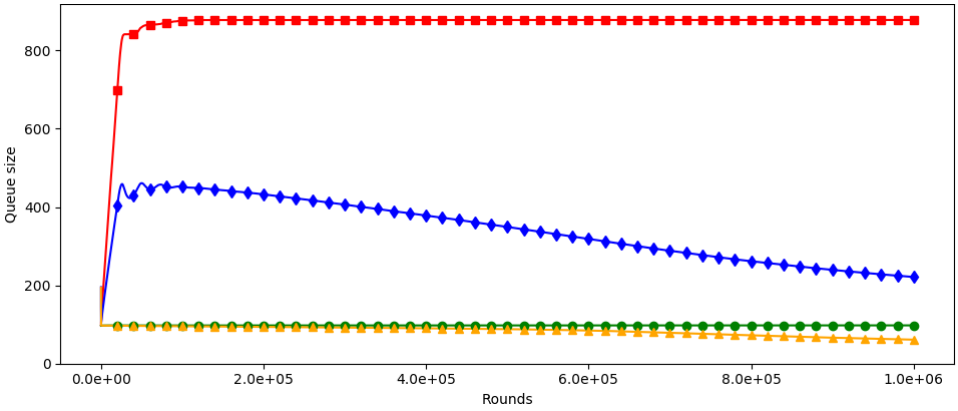
Adversary. In order to perform simulations, we need to define the behaviour of the adversary. We have chosen strategy of the adversary that seems to be challenging for the algorithm and reflects some real-life scenarios. We define an adversary by three parameters used at each round r: injection rate 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 $i \quad \{1, 2, 3, \ldots, n\}$: $P_1 = P_2 = \frac{1}{3} + \frac{1}{3n}$; $P_{i>2} = \frac{1}{3n}$.

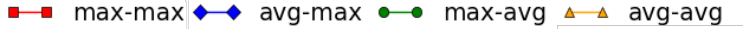
Injection rate ρ and burst-parameter p have values in $(0, 1)$. Note that the burst-probability parameter models the adversary injection behavior: between rare bursts of large numbers of packets (close to 0) and steady flow (close to 1). The stock-size β is a constant equal to 256, basing on operational 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;
- **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.



(a) 12 O'clock full-sensing protocol queues against injection rates $\rho = 0.968$, started with queues in each station equal to $q = 96$.



(b) Relation between markers, colors and measurements of queues

Figure 6.4: Protocols during 1 mln rounds for a system size 32 against uniform packet distribution.

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 6.4a 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 6.4a).

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 [HLR96] and in many other previous papers considering stochastic injections, may yield stability while having single queues many times above the studied average.

6.5.2 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 $n \quad \{4, 5, \ldots, 32\}$ on a single plot (Figure 6.5a).

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 6.5a and the stable injection rates depicted in Figure 6.5b 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. [HLR96]. 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.

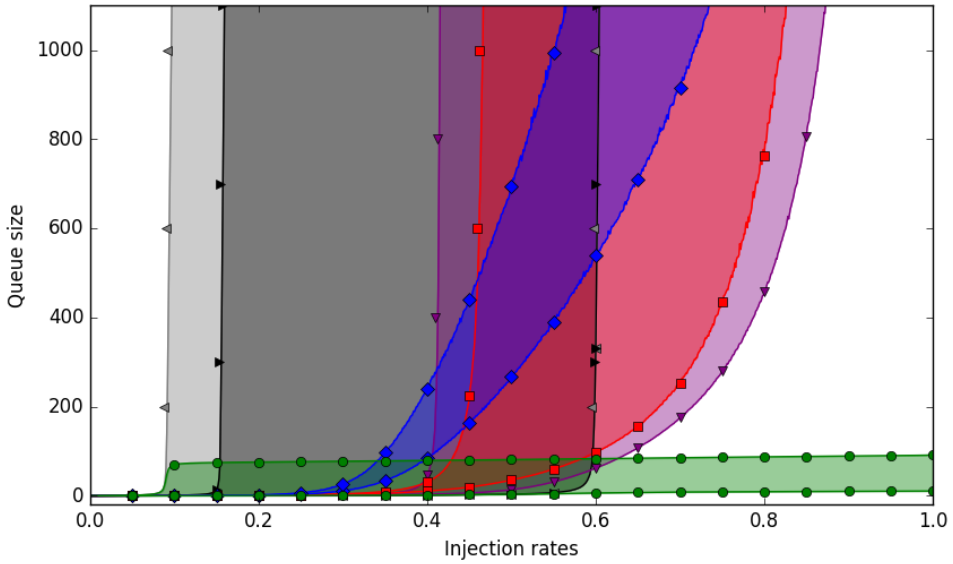
Acknowledgment-based protocols have the same Round-Robin implementation of selectors for system sizes $n \in \{4, 5, \ldots, 15, 17, 18\}$, because we were unable to generate better (n, ω)-selectors for $\omega \leq n/2$ required for Interleaved-Selectors in those cases. It follows that their plots overlap. 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 6.5b is caused by selectors being generated independently for each (larger) system size, which leaves a scope for further optimization of the quality of selectors.

Backoff protocols display dependency of queue stability on system size, and the following two interesting 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 6.5a. 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 6.5b.

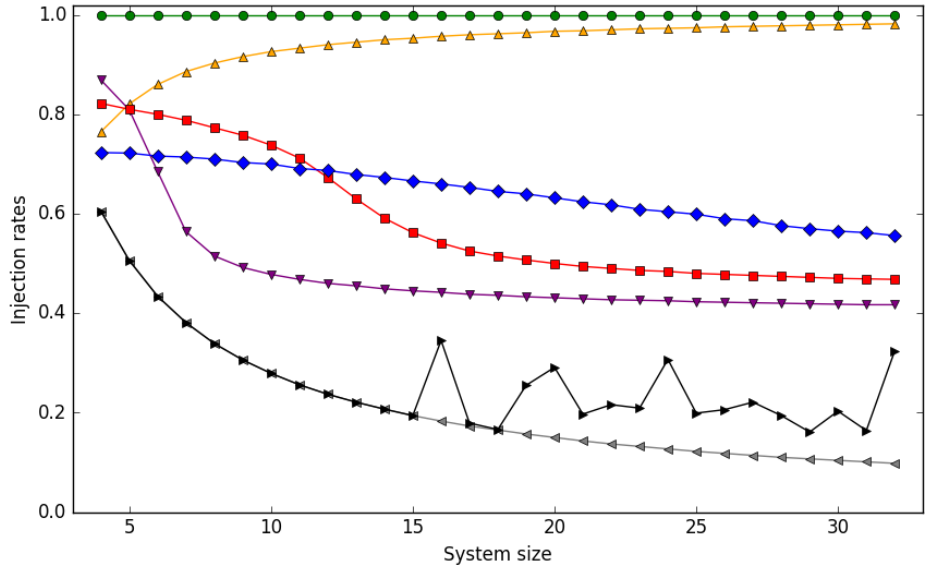
12 O'clock protocols have the least negative impact of an increase of system size over queue size stability, with 12 O'clock adaptive protocol being a champion in this terms, c.f., Figure 6.5b. Note that the stable injection rates of 12 O'clock full-sensing protocol improve with increasing system size.

6.5.3 Channel restrain 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



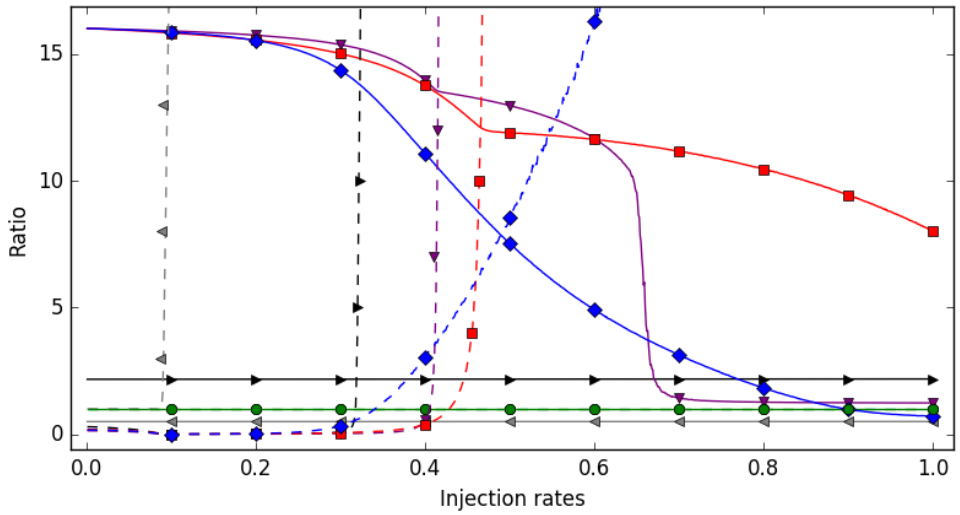
(a) Queue size by injection rates ρ $[0, 1]$.



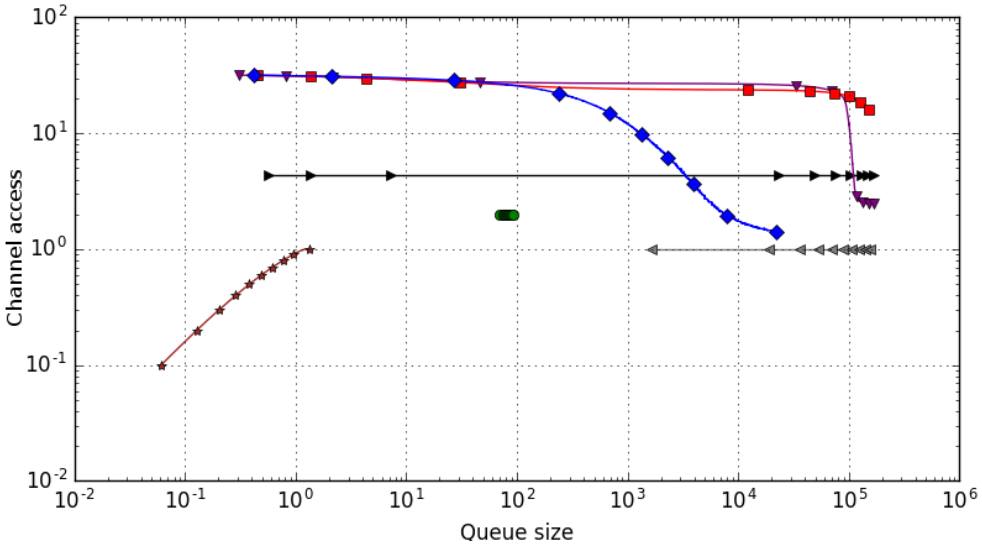
(b) Stable injection rates by system size.

(c) Relation between markers, colors and protocols.

Figure 6.5: Average round channel access and stable injection rates by system size.



(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 6.6: Average round channel access against queue sizes for injection rates ρ $[0, 1]$ and system size $n = 32$.

channel restrain. In Figure 6.6a we show the ratios of channel accesses and queue size of the considered protocols to the corresponding performances of 12 O'clock adaptive protocol.

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 Acknowledgment-based 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 6.6b presents our results in logarithmic scale: more efficient protocols in restrain-queue dimensions are closer to the State-aware protocol, as it represents the best performance algorithms can possibly 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).

Chapter 7

Routing-assisted communication on MAC

In this chapter we consider an extension of the standard multiple access channel model, where: (1) each packet has one of the stations connected to the channel as its destination; (2) a packet can only be delivered if its destination station was switched-on in the round of packets transmission; (3) algorithms can deliver packets in multiple hops, using stations other than source and destination ones, for routing. Below we present a more formalised and detailed model.

7.1 Shared Access Channels model extension

Packet construction. A packet $p = (d, c)$ consists of its *destination address* d and its *content* c. A destination is an integer in $[0, n-1]$ interpreted as a name of the station to which the packet needs to be delivered. A packet's contents is the information that the packet carries, which does not effect how the packet is handled.

Routing packets. Each injected packet needs to be delivered to its destination station. We say that a packet $p = (d, c)$ gets *delivered* in a round t when the following occurs: (1) a packet p is transmitted in round t and is heard on the channel, (2) the destination station d is switched on in round t. If a packet gets delivered then it is "consumed" by the destination station and disappears from the system.

A packet may hop from station to station in a store-and-forward manner. In effect a packet may be transmitted and heard on the channel a number of times.

If a station transmits a packet, which is then heard on the channel, then the packet is removed from the queue of the transmitting station. If a packet is heard on the channel but is not delivered in this round, then some station may *adopt* the packet by adding it to the queue; such a new station handling the packet becomes a *relay* for the packet and treats it as if it were injected directly by the adversary.

The task of *routing* is defined as follows: while packets are continually injected into the system, stations transmit them such that they are eventually delivered. The total number of packets that are queued in a round is referred to as the *queue size* in this round. The *delay of a packet* p is defined as the difference $t_2 - t_1$ between the round t_2 in which packet p gets delivered and the round t_1 in which packet p was injected.

Clearly, the channel restraint 2 is minimum to make the tasks of point-to-point communication feasible in principle, since at least one transmitter and one receiver need to be switched on in a round.

Routing algorithms. Routing is performed by distributed algorithms that are executed by all the stations concurrently. Correctness of a routing algorithm means that each injected packet is eventually delivered to its destination and a delivery occurs exactly once for each packet.

A station switched-on in a round either transmits a packet or senses the channel (listens to it) in this round. Whether control bits are included in packets is a feature of algorithms. The bits encoding packet's destination address **are not** considered as control bits. Algorithms that have a packet without any control bits are called *plain-packet* ones, they make a subclass of *general* routing algorithms.

The destination address of a packet is just a station's name, so it is represented by $O(\log n)$ bits. We consider only routing algorithms that use the conservative amount of $O(\log n)$ control bits per packet. This restriction on the number of control bits transmitted per round makes coordinating actions among the stations complex and in consequence we need much more subtler algorithm design.

Routing algorithms that do not use relay stations are said to *route directly*, and otherwise they are said to *route indirectly*. Algorithms that route directly make each packet hop only once, from the station into which the packet got injected, straight to the destination.

A routing algorithm is called *channel oblivious* when it determines in advance, prior to the start of an execution, for each station i and each round t, whether station i is on or off in round t. When a channel-oblivious algorithm

execution on a channel is a subject to a channel restraint of at most k we call such algorithm k-*channel-oblivious*. Observe that k-channel-oblivious algorithm is more restricted than a simply k-channel-restraint algorithm. Indeed, the latter has the ability to adjust which stations switch on on what rounds while the former cannot.

Classification. In this chapter we follow algorithm classification from our original paper [CHJ+19]. In the terminology of the taxonomy introduced in the chapter 5, algorithms studied in this chapter are:

- Time-aware, as the global clock is required for them to function;
- Destination-aware, as this is the main property of the shared channel compared to the classical MAC;
- Restraint-aware, as they provide a mechanism to adjust to the restraint k of the channel;
- Acknowledgement-aware, as MAC model assumes that stations know if the packet was delivered in the end of each round.

Proposed in the current chapter division into General and Plain-packet protocols reflects Control-bits channel capability. General protocols are Control-aware, as they can attach control bits to packets. Symmetrically, Plain-packet algorithms are Control-oblivious. Indirect and Direct routing defined earlier in this chapter reflects Routing-aware and Routing-oblivious capabilities respectively.

7.2 A review of subprocedure algorithms

In this chapter we utilise several existing algorithms as building blocks. Below we present short their descriptions as well as related important facts for those algorithms.

Move-Big-To-Front (MBTF). MBTF is an adaptive algorithm maintaining dynamic list of all stations in private memory of each station described in [CKR09]. Such a list is initialized at each station to have all the names of stations arranged in the increasing order: $0, 1, 2 \ldots, n-1$. The local lists are manipulated in the same way by all stations, based on channel feedback, hence they are identical copies of each other. The algorithm schedules exactly one station to transmit in a round, so that collisions never

occur. This is implemented by having a conceptual token traversal through the stations, which is initially assigned to the first station on the list. A station with the token broadcasts a packet, if it has any, otherwise the round is silent. A station with token considers itself *big* in a round when it has at least n packets; it attaches a control bit to every packet it transmits to indicate its *big* status. A *big* station is moved to the front of the list and it takes the token with it. If a station that is not *big* transmits in a round, or when it pauses due to the lack of packets while holding the token (so that the round is silent), then the conceptual token is (virtually) passed at the end of this round to the next station on the list (ordered in a cyclic fashion). The algorithm is stable against adversary injection rates $\rho = 1$.

Round-Robin-Withholding (RRW). RRW is a full-sensing algorithm operating in Round-Robin fashion [CKR09], that is – stations gain access to the channel in the cyclic order of their identities. There is only one station with a right to transmit, which is said to hold a conceptual token. Once the station receives the token, it withholds the channel to unload all the packets in its queue. A silent round is a signal for the next station, in the cyclic order of identities, to take over the conceptual token. The algorithm is stable against adversary injection rates $\rho < 1$.

Old-First-Round-Robin-Withholding (OF-RRW). OF-RRW is an extension of the RRW algorithm [ACKR19]. It is obtained by categorising packets into *old* and *new*. Intuitively, packets categorized as *new* become eligible for transmissions only after all the packets categorized as *old* have been heard. Formally, an execution is structured as a sequence of phases, which are contiguous segments of rounds of dynamic length, and then the notions of old versus new packets are defined with respect to them.

A phase is defined as a full cycle made by the conceptual token visiting the stations. No additional communication is needed to mark a transition to a new phase as all the stations can detect this by monitoring the position of the virtual token. A token leaves a station holding it after the station has transmitted all its old packets while new packets may remain waiting for the next token's visit. In a given phase, packets are old when they had been injected in the previous phase, and packets injected in the current phase are considered new for the duration of the phase. If a new phase begins, the old packets have already been heard on the channel and the new ones immediately graduate to becoming old. This means that the "old-go-first" principle is implemented by having packets injected in a given

phase transmitted only in the next phase. In particular, the first phase does not include any transmissions of packets, as all the packets, if any, are new. Specifically, algorithm OF-RRW operates by manipulating the token similarly as algorithm RRW does, except that when a station gets access to the channel by transmitting successfully, then the station unloads all the old packets, while new packets stay in the queue when the token is passed to the next station. The algorithm is stable against adversary injection rates $\rho < 1$ as proved in [ACKR19].

7.3 A summary of the results

We develop deterministic distributed algorithms routing adversarial traffic on the multiple access channels and assess their efficiency in the worst-case sense, where performance bounds depend on the known number of stations n and an unknown adversary. One of the algorithms maintains bounded queues for the maximum injection rate 1 subject only to the channel restraint 3. This channel restraint is provably optimal, in that obtaining the same throughput with the channel restraint 2 is impossible. Algorithms that have bounded latency for each fixed adversary of injection rate less than 1 are said to be universal. We give universal algorithms subject to the minimum channel restraint 2 that have the latency polynomial in the number of stations n. One of these algorithms uses control bits in packets and has latency $O(n^2)$ and another has stations transmit plain packets only and attains latency $O(n^3 \log^2 n)$.

An algorithm is k-channel-oblivious if at most k stations are switched on in a round and for each station the rounds it is switched on are determined in advance. We give a k-channel-oblivious algorithm that has latency $O(n)$ for adversaries of injection rates less than $\frac{k-1}{n-1}$ and show that there is no k-channel-oblivious stable algorithm against adversaries with injection rate greater than $\frac{k}{n}$. An algorithm routes directly when it does not utilize relay stations, in that each packet makes only one hop straight to its destination from the station it is injected into. We give a k-channel-oblivious algorithm routing directly that has latency $O\left(\frac{n^2}{k}\right)$ for adversaries of sufficiently small injection rates that are $O\left(\frac{k^2}{n^2}\right)$. We develop a k-channel-oblivious algorithm routing directly that is stable for injection rate $\frac{k(k-1)}{n(n-1)}$ and show that no k-channel-oblivious algorithm routing directly can be stable against adversaries with injection rates greater than $\frac{k(k-1)}{n(n-1)}$. All the performance bounds of algorithms and impossibility results are tabulated in Table 7.1.

Algorithm/Impos.	Sec.	Injection	Latency	Queues	Restr.	Properties
Max throughput:						
Orchestra	7.4.1	$\rho = 1$	∞	$2n^3 + \beta$	3	NObl-Gen-Dir
Impossibility	7.4.2	$\rho = 1$		Stable	2	
Universality:						
Count-Hop	7.5.1	$\rho < 1$	$\frac{2(n^2+\beta)}{1-\rho}$		2	NObl-Gen-Dir
Adjust-Window	7.5.2	$\rho < 1$	$\frac{18n^3 \log^2 n + 2\beta}{1-\rho}$		2	NObl-PP-Ind
Min Latency:						
Two-Hops	7.6.1	$\rho < \frac{1}{2}$	$\frac{6n+\beta}{1-2\rho}$		2	NObl-Gen-Ind
One-Hop	7.6.2	$\rho < \frac{1}{3}$	$\frac{6n+\beta}{1-3\rho}$		2	NObl-Gen-Dir
Oblivious indirect:						
k-Cycle	7.7.1	$\rho < \frac{k-1}{n-1}$	$(32+\beta) \cdot n$		k	Obl-PP-Ind
Impossibility	7.7.2	$\rho > \frac{k}{n}$		Stable	k	Obl
Oblivious direct:						
k-Clique	7.8.1	$\rho \leq \frac{k^2}{2n(2n-k)}$	$8\frac{n^2}{k}(1+\frac{\beta}{2k})$		k	Obl-PP-Dir
k-Subsets	7.8.2	$\rho = \frac{k(k-1)}{n(n-1)}$	∞	$2\binom{n}{k}(n^2+\beta)$	k	Obl-Gen-Dir
Impossibility	7.8.3	$\rho > \frac{k(k-1)}{n(n-1)}$		Stable	k	Obl-Dir

Table 7.1: A summary of the performance bounds of algorithms and impossibility results, broken into four main sub-groups. 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 or routing are as follows: Obl = oblivious, NObl = non-oblivious, Gen = general, PP = plain-packet, Dir = direct, Ind = indirect. The impossibilities are of existence of stable routing algorithms subject to the given injection rate, channel restraint, and properties of algorithms and routing. A bound on latency is also a bound on the number of queued packets. Latency ∞ means that it is possible for some packets never to be delivered. The parameters n and k are known, i.e. they can be part of code of algorithms.

In the second column we also point where a given result is described in our document.

7.4 Maximizing throughput

We present a direct-routing algorithm stable for injection rate 1. Clearly, this is the maximum throughput possible on multiple access channels. The algorithm requires channel restraint to be at least 3. We show that the number 3 is best achievable in this sense by proving the impossibility of attaining throughput 1 with channel restraint 2.

7.4.1 General direct-routing algorithm Orchestra

An algorithm stable for injection rate 1 which we give is called Orchestra. It schedules at most three stations to be simultaneously switched on at any round, with at most one of them transmitting. The algorithm builds on the paradigms developed in [CKR09], which gave a broadcast algorithm with throughput 1 for multiple-access channels without channel restraints.

We call a group of $n-1$ consecutive rounds of an execution a *season* if the last round t of the group satisfies $t \equiv 0 \pmod{n-1}$. For each season, there is a unique station associated with it called a *conductor*. Stations that are different from a conductor are called *musicians*. A conductor for a season transmits a packet in every round of this season, so there are no silent rounds. A round when a packet heard on the channel includes only control bits is called *light*.

Every station keeps an ordered list of all the stations. These lists are the same in every station at the beginning of a season; at such a moment they represent one list, which we call the *baton list*. Initially, the baton list consists of all the stations ordered by their names. Stations assume the role of conductors in their order on the baton list. The first station on the list is assigned to serve as a conductor for the first season.

The positions of stations on the baton list are understood as follows: the front entry of the list is considered as the first station on the list, then the consecutive stations have their positions increased by one, and the tail entry occupies the last nth position. In particular, if a station at position i moves to become the head of the baton list, then this station acquires position 1 while each station at the original position $j < i$, which means closer to the front than i, gets its position incremented to $j+1$, so that its distance from the head of the list increases.

The process of assigning conductors to seasons can be visualized as passing a virtual baton from station to station, such that a station holding the baton is a conductor. When a season ends then the baton is typically passed on to the next station on the baton list. The order determined by the list is understood in a cyclic sense, in that the first station becomes a conductor after the last one in the list has concluded its assignment. An exception for this process occurs when a conductor is moved to become the head of the baton list while keeping the baton.

A conductor of a season is switched on in each round of the season. A musician switches on during a season either to *learn* or to *receive*, or possibly both.

A packet transmitted by a conductor is destined to the musician who receives and contains control bits for the musician who is to learn. We say that the conductor *sends*

the packet to the musician who is to receive and *teaches* the musician who is to learn.

We explain the actions of teaching/learning and sending/receiving next.

The purpose of *learning* is to obtain information from the conductor, in particular one pertaining to a schedule to receive packets in the next season with the same conductor. A station learns by extracting control bits from a packet transmitted by the conductor and interpreting them as round numbers to be switched on during the next season when the same stations acts as conductor.

The purpose of *receiving* is to obtain packets injected into a conductor and destined for a receiving musician.

For a packet transmitted by a conductor to effectively serve its purpose, the following three involved stations need to be switched on simultaneously: the conductor, the learning musician, and the receiving musician. Musicians switch on during a season in the following manner. First, the musicians switch on to learn: they do it one by one, in the order of their names, for one round at a time. Second, the musicians switch on to receive: they do it according to the schedule taught during the latest previous season when the same station was a conductor.

A packet injected into a station, when it acts as a conductor, stays *new* for the duration of this season, and after that becomes *old*. A packet injected into a musician becomes old immediately. In particular, when a new season begins, then all the packets queued in the stations are old. At the start of a season, when some station becomes a conductor, this station computes a schedule to send the old packets during the next season when it will become a conductor again. The schedule concerns only these old packets that have not been scheduled yet for the current season. A conductor schedules packets to send in the order of their injections.

A station considers itself *big* if it has at least $n^2 - 1$ old packets in its queue. A conductor that is big at the beginning of a season teaches each musician of this status, by suitably setting a toggle bit in packets. After a musician learns this information, it moves the conductor to the front of its private copy of the baton list. Such a season concludes with all the musicians having identical private lists representing the baton list, with the conductor at the front. A big conductor keeps the baton for the next season, after moving to the front of the baton list, and stays at the front as long

	1	2	3	4	5	6	7	8	9	10	11	12
Conduct	A		B		C		A		A		B	
Learn	B	C	A	C	A	B	B	C	B	C	A	C
Receive	B	C	A	C	A	B	B	B	B	B	C	C

Figure 7.1: Example execution of Orchestra algorithm: top row represents rounds from 1 to 12; leftmost column stands for station roles: conduct, learn and receive; A, B, C stand for station names. In rounds 1 to 6 we observe seasons following the order of station names: stations A, B and C conduct in seasons in rounds $\{1, 2\}$, $\{3, 4\}$, $\{5, 6\}$ respectively. We assume that in round 1 station A has informed station B of having two packets destined to station B. We also assume that similarly, in round 4, station B has informed station C that it has two packets destined to C. In what follows, station A breaks the conducting order of names in round 9 by considering itself big; this information, together with the information of B receiving two packets from A in the next season has been passed to stations B and C in rounds 7 and 8 respectively. Station A has not claimed big status further, thus station B follows the order of conducting list in rounds 11 to 12.

as it is big.

This mechanism allows a single station to act as the conductor for long stretches of seasons, possibly indefinitely, in the case when the adversary injects packets into only one station.

An example of Orchestra algorithm execution can be seen in Figure 7.1.

We group seasons into contiguous intervals of seasons, depending on the heaviness on traffic during these seasons. If the total number of packets in the queues at the beginning of a season is greater than $n^3 - 2n + 1$ then the season belongs to a *dense interval of seasons*, which means the traffic will be heavy. Otherwise, a season belongs to a *sparse interval of seasons*, which means that traffic might be light. We expect that there exist big stations when traffic is heavy. A station is *pre-big* in a round of an interval of seasons if it has not been big during this interval before the round. A station is *post-big* in a round of an interval of seasons if it is not big now but it has been big by this round during the interval.

Theorem 7. *There are at most* $2n^3 + \beta$ *packets queued in a round of execution* Orchestra *algorithm against an adversary of injection rate* 1 *and with a burstiness coefficient* β.

Proof Let $D = n^3 - 2n + 1 = n(n^2 - 2) + 1$ be the number of old queued packets used to differentiate between dense and sparse intervals of seasons. A big station has at least $n^2 - 1$ old packets in its queue. By the pigeonhole principle, there exists at least one big station during a season in a dense interval of seasons.

We estimate the number of queued packets during a season by relating the season to either a dense or a sparse interval. The adversary's capability to inject packets due to the burstiness coefficient is accounted for only once at the end of a derivation of an upper bound on the number of queued packets.

First, we consider sparse intervals. The system starts a season with empty queues, so the first season belongs to a sparse interval. If the system starts a season in a sparse interval then it has at most D packets. The adversary can inject at most $(n - 1)$ packets during a season. So there can be at most these many old packets in stations within a sparse interval:

$$D + (n - 1) = n^3 - 2n + 1 + n - 1 = n^3 - n \ , \tag{7.1}$$

not including burstiness. This is also an upper bound on the number of queued packets when a sparse interval ends and a dense interval begins.

Next, we consider dense intervals. In such an interval, the adversary can be assumed to inject at full power, namely, a packet per round. If a packet is heard on the channel, then this does not affect the number of queued packets, since only one packet gets injected. Otherwise, if a light packet is heard, this results in the number of queued packets incremented by 1. This makes an upper bound on the number of light packets heard on the channel serve as an upper bound on the increase on the number of queued packets.

We claim that neither big nor post-big stations can contribute light rounds when acting as conductors during dense intervals. It follows that only pre-big stations contribute light rounds. We prove this claim next.

A big station has at least $n^2 - 1$ packets in its queue at the beginning of a season when it obtains the baton. It must have had at least $n - 1$ old packets to schedule at the beginning of the previous season it was conducting, since the adversary could inject at most these many packets in the meantime during n seasons, without accounting for burstiness:

$$n(n - 1) = n^2 - 1 - (n - 1) \ . \tag{7.2}$$

A conductor that had at least $n - 1$ old packets at the beginning of the previous season it conducted, has already scheduled a full season, so it sends

a packet in each among the $n - 1$ rounds of the current season. We conclude that a big station contributes no light rounds as long as it gains and maintains the status of a big conductor.

Consider a station i that is post-big but not big. Station i has an opportunity to transmit only when there is no big station before it on the list, as such a station would be visited by the baton first and moved the baton back to front. When station i receives the baton and i is not big, then there is a big station after i, because such a station exists in every season of a dense interval. The first big station encountered by the baton is moved to the front of the baton list, thereby incrementing the i's position to $i+1$. The position of a station cannot increase more than $n - 1$ times in this way. The last time when i was big, it had at least $n^2 - 1$ packets in its queues and was placed at the front of the baton list. Station i had at least these many packets when ending a season in which it was a big conductor:

$$n^2 - 1 - (n - 1) = n(n - 1) \; . \tag{7.3}$$

So the station can afford to increase its position up to n times while consistently sending $n - 1$ packets per season when serving as a conductor. We conclude that a post-big station contributes no light rounds during seasons in a dense interval.

We are finally ready to count light rounds in a dense interval, all of which could be contributed by pre-big conductors only. There are at most $n - 1$ pre-big stations in the system at the beginning of a dense interval, because at least one station is big. Light rounds occur in a season when the conductor has fewer than $n - 1$ old packets scheduled to send; let us assume conservatively that such a conductor does not have any scheduled packets to send, to maximize the number of light rounds the season contributes. A pre-big station i becomes a conductor only when the first big station on the baton list is behind station i. After the baton leaves and eventually reaches a big station, this big station advances to the front and the i's position shifts by 1. Such shifts can occur at most $n - 1$ times. It follows that a pre-big station can contribute at most $(n - 1)^2$ light rounds when acting as conductor in a dense interval. Therefore all the pre-big stations together contribute at most $(n - 1)^3$ light rounds as conductors.

The bound (7.1) estimates the number of queued packets when a dense interval begins. This number can grow by at most the number of light rounds while the adversary injects at full power, plus burstiness. These three parts together contribute the following:

$$n^3 - 1 + (n-1)^3 + \beta = (n-1)(2n^2 - 2n + 1) + \beta = 2n^3 - 4n^2 + 3n - 1 + \beta \le 2n^3 + \beta \; .$$

(7.4)

This quantity serves as the ultimate upper bound on the number of queued packets.

7.4.2 Lower bound for maximum throughput

Algorithm Orchestra requires at least 3 stations to be switched on in each round. We show that this is necessary for any algorithm to have throughput 1.

Lemma 11. *Given an algorithm for a system of $n \geq 3$ stations, let us assume that we have defined an execution of the algorithm until some round t_{i-1} such that the following holds: at least one station s has no packets in its queues, no other station has packets to be delivered to s, and there are at least $i-1$ packets in the system. Then either the execution can be extended without bounds and the number of packets in the system grows unbounded or here exists a round $t_i > t_{i-1}$ such that the execution can be extended until t_i in a way that the round t_i satisfies the following conditions:*

(a) *no packet is successfully transmitted at round t_i, and*

(b) *by the end of round t_i at least one station s has no packet in its queues and no other station has packets destined for station s, and*

(c) *after round t_{i-1} and by the end of round t_i, one packet per round has been injected into the system on average and the burstiness was at most 1.*

Proof Let t_0 be the first round, and i be an integer such that $i > 0$. Let s_1 and s_2 be two stations different from s. We consider the following two possibilities to extend an execution, as determined by the adversary after round t_{i-1}.

Case I: No packet is injected into station s, station s_1 gets one packet injected into it addressed to s in each odd round and one packet addressed to s_2 in each even round:

Case II: No packet is injected into station s, station s_1 gets one packet injected into it addressed to station s_2 in each round.

For station s, these two cases to extend the execution are indistinguishable up to a round t when station s becomes switched on for the first time after round t_{i-1}. Now there are two possible continuations:

Continuation 1: such a round t does not exist; in the execution determined by Case I the number of packets addressed to station s grows unbounded.

Continuation 2: such a t exists; then the execution determined by Case II extended to round $t_i = t$ satisfies the following:

(a) no packet is heard at round t, as there are no packets involving station s in the system;

(b) station s has no packets in its queues, since it had no packets in round t_{i-1} and no packet in the system was addressed to it, and between the rounds t_{i-1} and t_i (inclusive) this has not changed;

(c) the adversary injects exactly one packet per round.

By the properties (a) through (c) above, and by the assumption that there are at least $i - 1$ packets in the system by round t_{i-1}, the number of packets at the end of round t_i is at least i.

Lemma 11 gives a sequence of rounds $(t_i)_{i \geq 0}$ such that there are at least i packets queued in the system at round t_i.

Theorem 8. *No algorithm can be stable for channel restraint* 2 *and a system size greater than or equal to* 3 *against adversaries with injection rate* 1*.*

Proof Suppose that such an algorithm exists, to arrive at a contradiction. The argument is by induction on the round numbers. Consider the first round of an execution of the algorithm, to provide a base for induction. By the assumption about the channel restraint, at least one station s needs to be switched off. The assumptions of Lemma 11 are satisfied for $t_0 = 1$, so that $i = 1$.

Next we show the inductive step. We assume that the assumptions of Lemma 11 are satisfied for some $i \geq 1$, so that there is an execution determined up to some round t_{i-1} such that the following holds: at least one station s has no packets, no station has packets addressed for s, and there are at least $i - 1$ packets in the system.

By Lemma 11, this prefix of an execution up to round t_{i-1} either could be extended to a full execution such that the number of queued packets grows unbounded, or there is a round t_i and an extension that satisfy the assumption of Lemma 11 for $i + 1$. To conclude, either there is i such that from round t_i there is an unstable extension of the execution, or we could

continue extending the execution through rounds t_j, for all integers j. In the latter case, the number of packets in the system grows unbounded with j, and the resulting execution is unstable.

7.5 Universal routing

We give two routing algorithms with universally bounded latency. One routes directly while using control bits in packets to coordinate stations and the other routes indirectly with plain packets only.

7.5.1 Direct-routing algorithm Count-Hop

The direct-routing algorithm using control bits in packets is called Count-Hop; it operates as follows. One station is dedicated to serve as a *coordinator* and the other stations are *workers*. An execution is structured into *phases*. Packets transmitted in a phase need to be *old*, in that they were injected in the previous phase. Packets injected in the current phase are *new* for the duration of the phase. At a round when a phase ends, all the packets available in the system become old for the next phase. These are the only old packets for the next phase, which means that each station knows which among its packets are old, for the duration of this phase, when a new phase begins. The first phase consists of n rounds during which all the stations are switched off. Each of the following phases proceeds through *stages*, which are consecutive rounds spent by the stations working to accomplish some task.

A phase is partitioned into n stages, one for each receiving station. Such a stage for each receiving station consists of three substages. During the first substage, each station, except for the receiving station v and the coordinator, transmits once, sending a packet with the number of old packets destined for v. This information allows the coordinator to assign to each station a number of consecutive rounds to transmit all its packets to v. The second substage consists of the coordinator transmitting the offset number for each station to be switched off waiting for its turn to transmit. Finally, the third substage has all the stations switch on one by one when the turn comes to transmit the old packets destined to v, while the station v is switched on during the whole substage and the coordinator is switched off. An example of Count-Hop algorithm execution can be seen in Figure 7.2.

	1	2	3	4	5	6	7	8	9	10	11	12
A	X			O			X	X				
B		X			O							
C			X			O			X	X	X	
D	O	O	O	X	X	X						X
E							O	O	O	O	O	O
	I.			II.			III.					

Figure 7.2: Example execution of Count-Hop algorithm: top row represents rounds from 1 to 12; leftmost column stands for station names A, B, C, D, E; X stands for transmission and O for reception of packets; bottom row represents sub-phases. In this example station E is the receiving station and D is the coordinator. In the first sub-phase (rounds $1, 2, 3$), stations A, B, C inform the coordinator station D about the number of old packets they have in their respective queues destined to the receiving station E. In the second sub-phase (rounds $4, 5, 6$), the coordinator station announces to stations A, B, C their respective reserved ranges of rounds for transmission in the third sub-phase. Finally, in the third sub-phase (rounds 7 to 12), stations A, B, C, D transmit all of the packets destined to station E.

Theorem 9. *A direct-routing algorithm* Count-Hop *requires the channel restraint* 2*, is stable for each injection rate* $\rho < 1$*, and its latency for such an injection rate is at most the following:*

$$\frac{2(n^2+\beta)}{1-\rho} \,. \tag{7.5}$$

Proof Each packet is delivered from the station of injection to the station of destination in one direct hop, by the algorithm's design. There are $(n-1)^2$ rounds spent on transmitting auxiliary numbers, only. While such packets are transmitted, the adversary can inject new packets. These packets will extend the duration of the next phase by up to $\rho(n-1)^2$ rounds. This phenomenon can be iterated in a cascade-like manner, since when packets are transmitted, the adversary can use this time to inject even more packets. Taking into account all the possible extensions of phases, the duration of any phase is at most the following:

$$(n^2+\beta)(1+\rho+\rho^2+\ldots) = \frac{n^2+\beta}{1-\rho} \,. \tag{7.6}$$

A packet stays in a station during at most two consecutive phases.

7.5.2 Indirect-routing algorithm Adjust-Window

We describe an indirect-routing plain-packet algorithm that requires only a constant channel restraint, but has universally bounded latency and attains packet delay $O(n^3 \log n)$. The algorithm is called Adjust-Window.

An execution of Adjust-Window is structured into segments called *time windows*. The size of a time window may increase in the course of an execution. The current size of a window is denoted by L. All the stations use the same value of L at each round.

Packets injected before the current time window are called *old* and packets injected during the current time window are called *new* for the duration of the window. The goal to achieve during a window is to deliver all the outstanding old packets to their destinations. Whether or not this goal is accomplished in a particular time window may depend on the magnitude of L. Old packets that do not get delivered in a window remain old for the duration of the next window. If some old packets are not delivered in the current time window, then the window size L gets doubled to become $2L$, which determines the duration of the next window. Otherwise, if all the old packets are successfully delivered in a window, then the window size L stays the same, and so the duration of the next window stays the same as well.

Algorithm Adjust-Window works with channel restraint 2. An execution is organized such that in each round at most one station transmits. If a station i transmits a packet in a round and another station j is switched on, then we say that station i *sends* the transmitted packet to j. If a station i sends a packet to station j and the packet is heard on the channel, then station i removes the packet from its queue and station j either consumes it, if it is addressed to j, or else adopts it and becomes its relay station. This means that packets may hop from station to station, and routing may be indirect.

A time window is partitioned into three stages: Gossip, Main, and Auxiliary. The goal of a Gossip stage is to exchange information between stations regarding the numbers of old packets in their queues with particular destinations. In the Main stage, the stations transmit old packets directly to their destinations according to a schedule based on the information exchanged and learned during the preceding Gossip stage. A station knows the part of such a schedule relevant to its actions: it knows when to transmit packets to which destinations, and in which rounds to listen to packets addressed to it. It may happen that a station i needs to convey some information to a station j while station i does not have packets with

the destination j, then i sends some packet(s) to j whose destination is different from j. An Auxiliary stage deals with delivering such relayed packets to their destinations, as well as handling old packets at stations that could not participate in neither Gossip nor Main stages due to lacking sufficiently many packets.

All packets transmitted on the channel may only be plain packets. This means that numbers encoded as strings of bits cannot be piggybacked on packets. Instead, we design a protocol called *coded transfer* (of bits) to encode sequences of bits by way of sequences of transmissions of single packets, with rounds of transmissions possibly interspersed with silent rounds. One round of coded transfer can convey one bit. Coded transfer needs to overcome the following technical obstacle: a station that is supposed to transmit a packet to convey a bit needs to have at least one packet available in its queue, which after a successful transmission is removed from the queue. This implies that stations with empty queues cannot transmit packets and so their lack of transmission activity needs to be properly interpreted by the other stations.

Coded transfer of bits works as follows. Lets assume that a station i is to transfer r bits $B_1, B_2, \ldots, B_r$ to another station j, for $0 \leq i, j < n$, and the size of the queue of station i is at least r. Then, in r consecutive rounds, station i sends a packet to j in the kth consecutive round if and only if $B_k = 1$, for $1 \leq k \leq r$, while station j listens to the channel. This approach makes the transmitting station i use one packet for each transmitted bit 1 and no packet for a 0. Station i may transmit packets not addressed to j, if packets addressed to j are not available at i; if this occurs then station j adopts them and becomes their relay.

The stages of a window take a specific duration, depending on n and L. Let L_G, L_M, L_A denote the number of rounds of a Gossip stage, a Main stage and an Auxiliary stage, respectively. These three numbers sum up to L: $L_M + L_G + L_A = L$. The magnitudes of L_G and L_A are determined next, and the remaining part of L rounds of a window is taken by a Main stage.

We specify that stations without sufficiently many old packets do not participate either in Gossip stages or in Main ones. We categorize such stations as *small*. In what follows, the notation $\lg x$ stands for $\log_2(x+1)$. Formally, a station is small in the considered window of size L if the size of its queue at the beginning of that time window is less than $4n \lg L$; otherwise, the station is *large* in the window. A large station has sufficiently many packets to transmit should they be needed to convey bits by coded transfer.

A Gossip stage. The goal of a Gossip stage is to share information among the stations about the contents of their queues at the beginning of the current time window. Such transmission of information is performed indirectly by coded transfer.

A Gossip stage consists of n^2 *phases*, indexed by all pairs (i, j) for $1 \leq i, j \leq n$. Each phase takes $2+3\lg L$ consecutive rounds. Thus, a Gossip stage takes these many rounds:

$$L_G = n^2(2 + 3\lg L)\ . \tag{7.7}$$

An (i, j)-phase for $i = j$ is structured as follows. The station j listens to the channel in each round of a phase, as the only station that does so. If the station i is small, it stays silent for the whole phase; otherwise, if i is large, it conveys some information to j as follows. The station i sends a packet to j in the first round of the phase to notify j that i is large. Then, in the second round of the phase, i sends a packet to j if and only if its queue size is greater than L. Finally, during the following $3\lg L$ rounds of the phase, i conveys the following three numbers to j by coded transfer:

1. the minimum of its queue size and L,
2. the number of packets in its queue with destination j, or L if the number of packets to j is at least L,
3. the number of packets in its queue with destinations k such that $k < j$, or L if the number of such packets is at least L.

An example of a single phase of Adjust-Window in the Gossip stage, can be seen in Figure 7.3.

At the end of a Gossip stage, each station j knows one of the following about each station i, where $0 \leq i < n$ and the size of the queue of station i is measured at the beginning of a Gossip stage:

(a) the queue size of i is less than $4n\lg L$, or otherwise

(b) the queue of i has more than L packets to some destination, or otherwise

(c) the exact size of the queue of i, the number of packets in i with destinations k such that $k < j$, and the number of packets in the station i with the destination j, when none of the cases (a) nor (b) holds.

	1	2	3	4	5	6	7	8	9	10	11
i	X	X	X				X	X			X
j											
	1	1	1	0	0	0	1	1	0	0	1

Figure 7.3: An illustration of a phase of Adjust-Window algorithm in the Gossip stage: top row represents rounds from 1 to 12; leftmost column stands for station pair indexes i, j, where $1 \leq i, j \leq n$ and $i = j$; X stands for transmission; bottom row represents binary numbers received by station j as the metadata of the transmission pattern. In this example, we assume that system size $n = 4$, window length is $L = 4$, i-th station queue is 64, station i has three packets destined to station j, there is one packet in total in the queue of station i destined to any of stations k, such that $k < j$. In the first round of the phase, station i informs that it is large. In the second round, it signals that the size of its queue is greater than L. In the following nine rounds station i encodes three numbers: $L = 4$, as its queue of 64 packets is greater than L and 4 is the minimum of the two; the number of packets destined to j (three packets); number of packets destined to stations k, such that $k < j$ (one packet).

This information determines the size of the next time window. Namely, if there is some station i such that the queue size of i is greater than L then the window size is doubled to become $2L$. Similarly, if none among the queue sizes is greater than L but the sum of the queue sizes of all the stations is greater than the length of the Main stage, the window size is also doubled to become $2L$. If none of the two conditions holds, the time-window size L stays the same for the duration of the next time window.

A Main stage. If it is known that some stations have their queue sizes greater than L, then the Main stage is dedicated to the station with the smallest name among them, which spends all the rounds transmitting its packets. Suppose otherwise that no station has the size of the queue greater than L. Let m be the total number of packets queued in the stations, which is known by each station. The stations that are small in this window, meaning with fewer than $2 + 3n \lg L$ packets in queues at the beginning of the window, do not transmit in this stage, as if they had no packets. Based on the information collected in the Gossip stage, every station can compute on its own a comprehensive schedule for delivering the minimum of L_M and m packets from their queues that have been already stored

in these queues at the beginning of the current time window. The schedule determines the sender of a packet and the destination of a packet for each round. Transmitting according to such a schedule completes the stage, where only a transmitter and receiver are switched on in each round.

A Main stage, as given above, has stations operate based on the sizes of their queues at the beginning of the current time window. The actual numbers of old packets that the stations have in their queues, when a Main stage was planned, might have changed during the Gossip stage. This is because, as stations transmit old packets during a Gossip stage, these packets are not necessarily received by their destination stations, and so still need to be forwarded by the stations that received them and now should act as relays. This issue is taken care of by Auxiliary stages.

An Auxiliary stage. The goal of this stage is to deliver all the old packets that are in the queues of small stations along with the packets received by the stations in a Gossip stage during the coded transfer that still need to be forwarded. This task is accomplished by the following Round-Robin style algorithm.

A stage is structured into *phases* of n^2 rounds each, indexed by the pairs (i, j) for $0 \leq i, j < n$. In a round (i, j) of a phase, j listens and i sends a packet to j, provided that i has such a packet in its queue. An illustration of a single phase can be found in Figure 7.4.

	1	2	3	4	5	6	7	8	9	10	11	12
A	X	X		O			O			O		
B	O			X		X		O			O	
C		O			O							O
D			O			O			O			

Figure 7.4: Illustration of a phase of Adjust-Window algorithm in the Auxiliary stage: top row represents rounds from 1 to 12; leftmost column stands for station names A, B, C, D; X stands for transmission O for reception. In this example, station A transmits packets to stations B and C in rounds 1 and 2 respectively; station B transmits packets to stations A and D in rounds 4 and 6 respectively. Rounds 3, 5 and 7 to 12 are silent, as station A has no packets destined to station D, station B has no packets destined to station C, stations C and D have no packets in their queues.

Since each small station has at most $4n \lg L$ packets at the beginning

of a window, and a station can receive at most these many packets

$$(2 + 3 \lg L) \cdot (n - 1) \leq 4n \lg L \tag{7.8}$$

during a Gossip stage, provided that $2 \leq \lg L$, it is sufficient to execute $8n \lg L$ phases to guarantee that all the considered packets are delivered to their destinations. We may specify that an Auxiliary stage takes at most

$$L_A = n^2 \cdot 8n \lg L \tag{7.9}$$

rounds.

To summarize, a Gossip stage consists of $n^2(2 + 3 \lg L) \leq 4n^2 \log L$ rounds and an Auxiliary stage takes $8n^3 \log L$ rounds. A Main stage takes the remaining rounds, their number being at least

$$L - 4n^2 \lg L - 8n^3 \lg L \geq L - 9n^3 \lg L \ , \tag{7.10}$$

for sufficiently large n. We set the initial value of L to the smallest natural number such that the following inequality holds:

$$L - 9n^3 \lg L \geq \frac{1}{2} L \ . \tag{7.11}$$

Thus the first Main stage takes at least half of the length of the first window, and so there is enough room for the first Gossip and Auxiliary stages to be completed.

Theorem 10. *A plain-packet algorithm* Adjust-Window *needs the channel restraint* 2 *and has the following its latency for each adversary of injection rate* $\rho < 1$ *and burstiness* β *is bounded by:*

$$\frac{18n^3 \log^2 n + 2\beta}{1 - \rho} \ , \tag{7.12}$$

where n *is sufficiently large with respect to* ρ *and* β*.*

Proof It suffices to have such a window length L that the duration of a Main stage is greater than the largest number of packets that might be injected in a window, which is $\rho L + \beta$. Let us assume temporarily that ρ and β are known to the stations and therefore the initial value of L can be properly determined, based on these ρ and β. This assumption may be dropped, as we show later.

A Main stage has at least $L - 9n^3 \lg L$ rounds, so it suffices for a window size L to satisfy the following inequality:

$$L - 9n^3 \lg L \geq \rho L + \beta \ . \tag{7.13}$$

The above inequality holds for $L = \frac{9n^3 \lg^2 n + \beta}{1-\rho}$, for n that is sufficiently large with respect to ρ and β, as can be verified directly. The latency is at most $2L$ because a packet may spend two consecutive windows in a queue, first as a new packet and then as an old one. This completes the analysis in the case when L is properly set at the beginning of an execution.

Next we incorporate into the analysis the mechanism by which the length of the next window may get increased after a current window is over. If the window size is not increased at the end a time window W, then all packets injected before W are delivered during W and therefore only packets injected during W are present in queues at the end of W. Then our estimates of window size from the beginning of the proof apply.

Suppose the size of a time window W is greater than the window size of the immediately preceding window. In general, let $W_1, W_2, W_3, \ldots, W_k$ be a sequence of consecutive windows, where W_{i-1} occurs directly after W_i, for each i, and window W occurs directly after W_1. Let moreover this sequence be such that the size of the window W_{i-1} is greater than the window size of the window W_i, for each $1 < i \leq k$. This means that the window size was increased at the end of W_i, and also either W_k is the first window of the considered execution or the size of the window preceding W_k is equal to the size of the window W_k. Thus, all packets injected before W_k are delivered during W_k. Therefore, as the window size of W_i is twice as large as the window size of W_{i+1} for $1 \leq i < k$, the number of packets in all queues at the beginning of the window W is at most the following:

$$\rho L \cdot \left\lfloor \frac{1}{2} + \frac{1}{2^2} + \cdots + \frac{1}{2^k} \right\rfloor + \beta \leq \rho L + \beta \ . \tag{7.14}$$

In each execution, the window size L eventually becomes sufficiently large to provide that all packets injected before a window are transmitted within this window, and this final window size is at most

$$L = \frac{9n^3 \lg^2 n + \beta}{1 - \rho} \ , \tag{7.15}$$

by an argument applied as in the first part of this proof. The latency is again at most twice the length of such a longest window.

7.6 Minimizing latency

In this chapter we develop two general algorithms attaining the minimum latency $O(n)$ for constant channel restraint. Of those algorithms, one routes directly and another indirectly.

7.6.1 General indirect routing algorithm Two-Hops

We give an algorithm for channel restraint 2 that has $O(n)$ latency for a constant range of injection rates. The algorithm does not have universally-bounded latency. The algorithm is called Two-Hops.

An execution has a structure similar to that for algorithm Count-Hop. One station is dedicated to serve as a *coordinator* and the other stations are *workers*. An execution is structured into *phases*. Packets transmitted in a phase need to be *old*, in that they were injected in the previous phase. Packets injected in the current phase are *new* for the duration of the phase. At a round when a phase ends, all the packets available in the system become old for the next phase.

The first phase consists of n rounds during which all the stations are switched off. Each of the following phases proceeds through *stages*, which are consecutive rounds spent by the stations working to accomplish some task. In the first stage of a phase, the workers begin by transmitting once each in the order of their names. Each transmitted packet carries the number of old packets that the worker station holds. The coordinator collects all these numbers, which allows to assign to each station a range of consecutive rounds of length equal to the number of its packets for exclusive transmissions. The second stage proceeds by the coordinator transmitting the consecutive numbers of rounds when stations need to switch on and transmit all their packets, while the workers switch on to hear this packet one by one in the order of their names. The third stage proceeds by each station switching on during the allocated range of consecutive rounds and transmitting all its packets one by one while the coordinator collects them. After hearing all the old packets, the coordinator knows how many of them are destined for each particular station, so it can compute a schedule of ranges of consecutive rounds when each recipient station would need to switch on to hear all the packets. The forth stage proceeds by the coordinator transmitting the consecutive numbers of rounds when stations need to switch on to hear all the packets destined for them, while the workers switch on to hear this packet one by one in the order of their names. Each transmitted packet also

includes the offset number of rounds to wait for the beginning of the next phase. The fifth stage proceeds by the coordinator transmitting the packets it handles one by one in the order of destination-station names, while the receiving stations switch on at proper ranges of consecutive rounds to listen to the channel. The last packet in a batch for a receiving station includes a bit to indicate that the batch is over so that the station should switch off.

Theorem 11. *Algorithm* Two-Hops *requires the channel restraint of* 2*, it is stable for each injection rate* $\rho < \frac{1}{2}$*, and its latency for such injection rates is at most the following:*

$$\frac{6n+\beta}{1-2\rho} \ . \tag{7.16}$$

Proof Each packet may make two hops, which effectively doubles the injection rate. It follows that the bound $2\rho < 1$ is necessary to have stability. The number of rounds per phase when packets are not transmitted is $3(n-1)$, which occurs during the first, second, and fourth stage. In a phase, the adversary may inject packets during such $3(n-1)$ rounds and during rounds when packets are heard. This may lead to continually increasing phases when each extra time t on hearing packets in a phase allows the adversary to inject ρt more packets for the next phase, which will take time $2\rho t$ to be transmitted. Additionally, the adversary may increase the number of injected packets by the burstiness coefficient. The number of packets queued during consecutive phases may increase geometrically, if such a pattern is iterated. We obtain the following bound on the number of rounds in a phase:

$$(3(n-1)+\beta)(1+2\rho+(2\rho)^2+\ldots) \leq \frac{3(n-1)+\beta}{1-2\rho} \ . \tag{7.17}$$

A packet spends at most two consecutive phases waiting to be delivered. The burstiness coefficient β can be used at most once if the adversary injects at full power. It follows that the following is an upper bound on the delay of a packet:

$$\frac{6n+\beta}{1-2\rho} \ , \tag{7.18}$$

for $\rho < \frac{1}{2}$.

7.6.2 General direct routing algorithm One-Hop

A general algorithm for channel restraint 2 that has $O(n)$ latency for a constant range of injection rates is called One-Hop. One station has a role of *coordinator* and other stations are *workers*. We assume existence of a static list of names of stations and give to the first station from the list the role of coordinator.

An execution is structured into *phases*. Each phase has associated with it set of packets injected in the previous phase. We call those packets *old* packets. Each phase consists of three stages: (1) evaluation of the number of old packets, (2) creation of schedule connecting transmitting stations to receiving ones and (3) execution of the schedule.

In the first stage, the coordinator in $n - 1$ rounds collects number of old packets from each worker station in a Round-Robin way. From this data it calculates the total number of old packets m. Next, in a sequence of another $n - 1$ rounds, the coordinator provides each worker W with the number of old packets m, together with the sum of number of packets in queues of all workers before station W on the list. The latter is used as the switch-off timeout for station W.

In the second stage the coordinator learns in m rounds the destination of each packet: workers switch-on one in a time, one round per packet, to provide this information. After that, the coordinator calculates the number of receiving rounds required of each worker W and sum of receiving rounds for all workers before W on the list. The latter is used as a switch-off timeout for station W. In a sequence of $n - 1$ rounds the coordinator delivers this information accordingly to each worker. In the following after that m rounds it informs workers about rounds they are required to be receiving in the next stage. It is assumed that transmitting stations would follow the order of the list and withhold the channel until all of their respective old packets are transmitted.

Finally in the third stage of m rounds, packets are successfully delivered.

Theorem 12. *Algorithm* One-Hop *requires the channel restraint of 2, it is stable for each injection rate* $\rho < \frac{1}{3}$*, and its latency for such injection rates is at most the following:*

$$\frac{6n + \beta}{1 - 3\rho} \,. \tag{7.19}$$

Proof Consider a phase P. Lets assume that P has d old packets in the queues of the stations in the beginning of the first round of P. It follows

that the length of P is $3(n-1)+3d$ rounds. The length of the phase P is triple the number of old packets in the phase, hence the bound $3\rho < 1$ is necessary to ensure stability. During the phase P an adversary can inject up to $3\rho(n+d-1)+\beta$ new packets, effectively extending the length of the next phase P. Taking into account recurrent extension, we obtain the following bound on the length of the phase:

$$(3(n-1)+\beta)(1+\rho+\rho^2+\ldots) \leq \frac{3(n-1)+\beta}{1-\rho}.$$

The packet latency is at most the double of that value, as packets injected in the beginning of phase P will be transmitted by the end of the next phase P. Hence the upper bound on the delay of a packet:

$$\frac{6n+\beta}{1-3\rho}, \tag{7.20}$$

for $\rho < \frac{1}{3}$.

7.7 Channel-oblivious indirect routing

Let an integer $k < n$ denote a channel restraint. We present now a plain-packet k-channel-oblivious algorithm called k-Cycle.

7.7.1 Plain-packet algorithm k-Cycle

The algorithm operates as follows. Up to k stations are switched on in each round. The stations are assigned into (non-disjoint) *groups* of size k each. The ith group is denoted as G_i, for $1, \ldots,$. Group G_1 consists of the k stations $0, 1, \ldots, k-1$, the next group G_2 comprises the station $k-1$ and the next $k-1$ stations $k, k+1, \ldots, 2k-2$, the next group G_3 includes station $2k-2$ and the next $k-1$ stations $2k-1, 2k, \ldots, 3k-3$, and so on, with the last group padded with dummy stations if needed. The underlying idea is that a group consists of k stations with consecutive numbers, and a group G_{i+1} starts from the last station in group G_i and includes the next $k-1$ stations. In general, the number of groups is at most $\leq \frac{n-1}{k-1}+1$.

We assume that the inequality $2k \leq n+1$ holds. Observe that if it does not, the algorithm selects a more restrictive channel restraint $k < k$, such that $2k = n+1$. It follows that there are at least two groups in any execution of k-Cycle algorithm.

Note that two consecutive groups share one station, called a *connector* of these groups, with group G sharing station 0 as a connector with

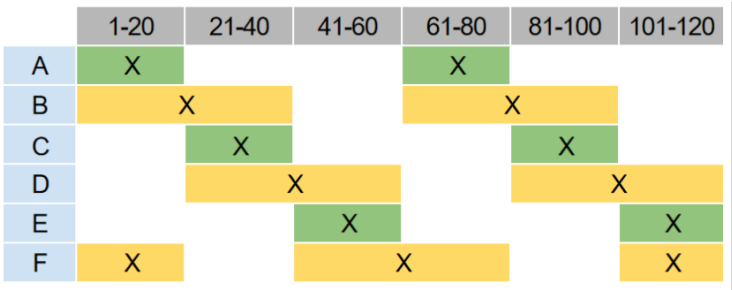


Figure 7.5: Example execution of k-Cycle algorithm: top row represents rounds from 1 to 120; leftmost column stands for station names A, B, C, D, E, F; X stands for stations being switched on in corresponding rounds. In this example system size $n = 6$,channel restraint $k = 3$, time segment is $\delta = 20$. There are three groups: $G_1 = \{A, B, F\}$, $G_2 = \{B, C, D\}$ and $G_3 = \{D, E, F\}$. Observe that stations B, D and F are coordinators.

group G_1. The stations in a group are ordered by their names into an ordered cycle. Groups themselves are also arranged into an ordered cycle as follows: group G_{i+1} follows group G_i, for $i <$, and group G_1 follows G .

In each round t of an execution, all the stations in some group G_i are switched on, with the other stations switched off; we say that group G_i is *active* in round t. The pattern of activity among the groups follows round robin according to the order cycle of the groups. A group is active for a time segment of these many rounds:

$$\delta = \frac{4(n-1)k}{n-k} . \tag{7.21}$$

When this time segment ends, the next group in the cyclic order takes over. You can find an illustration of an example execution of k-Cycle algorithm in Figure 7.5.

Each group executes an algorithm related to the broadcast algorithm Old-First-Round-Robin-Withholding (OF-RRW) during the consecutive rounds the group is active. Algorithm OF-RRW was described in Section 7.2, we adapt it as a building block of routing algorithms; the details of the adaptation are given next.

There is a conceptual token associated with each group. The actions of stations in a group are controlled by feedback from the channel. The feedback is the same for all the stations in a group, which allows to handle the token in such a manner that it is not duplicated nor lost. The token passes through all the stations in a group in a Round-Robin manner. When the token completes the whole cycle then this also ends a *phase*. Packets

injected or adopted during a phase are *new* for this phase and otherwise they are *old* for this phase. When a station receives a token then it transmits all its old packets one by one. If there is no old packet to transmit by a station holding the token then this station does not transmit anything, which results in a silent round. A silent round triggers the token to advance to the next station in the group in their cyclic order. When a station holding the token of a group G_i transmits then the packet is heard on the channel by all the stations in the group G_i. If the destination station of this packet belongs to G_i then the packet gets delivered and otherwise the station in G_i that is a connector with G_{i+1} adopts the packet and becomes its relay. This mechanism of handling packet implies that a packet may hop through all the groups until it reaches its destination station.

Theorem 13. *Algorithm* k-Cycle *routes packets correctly, when the channel restraint is at least* k*, and has latency at most* $(32+\beta)\cdot n$ *against an* (ρ,β)*-adversary such that* $\rho < \frac{k-1}{n-1}$.

Proof A group operates as a virtual cycle of k stations executing broadcast algorithm OF-RRW. Such a cycle in isolation would have broadcast latency at most

$$\frac{2}{1-\rho}\cdot k+\beta(1+\rho) \le \frac{2k}{1-\rho}+2\beta\ , \tag{7.22}$$

for an injection rate satisfying only the inequality $\rho < 1$; see [ACKR19]. A packet may perform at most $\frac{n-1}{k-1}$ hops through consecutive groups, which effectively amplifies injection rate by this factor. Therefore injection rates need to be less than $\frac{k-1}{n-1}$ to make routing stable.

The bound (7.22) on packet delay has two parts, among which $\frac{2k}{1-\rho}$ applies to packets injected within the injection-rate component and 2β applies to the packets for which injection-rate component does not suffice and they need the adversary's burstiness to justify their injection; see [ACKR19]. We consider the delay of packets by categorizing the packets into two groups: those for which the injection-rate component $\frac{2k}{1-\rho}$ suffices and the remaining ones to which the burstiness component 2β needs to apply to justify their delay. This categorization of packets is for analysis only.

For packets accounted for as injected subject to the injection-rate constraint, the bound $\frac{2k}{1-\rho}$ on packet delay becomes at most

$$\frac{2k(n-1)}{n-k} \tag{7.23}$$

after combining with the upper bound $\rho < \frac{k-1}{n-1}$ on injection rates.

Bound (7.23) on packet delay is less than the duration of a continuous segment of rounds of activity of a group of stations determined by (7.21). This implies that all the packets held by the stations in a group, and accounted for as injected subject to the injection-rate constraint, are heard on the channel when the group becomes active. This also means that these among such packets that are addressed to other groups will hop through the connector to the next group, while their current group is active. Such hops will continue without delay other than that incurred by the period of activity of the group where these packets reside.

The bound (7.23) needs to be increased to the duration δ for a period of activity (7.21) of a group and then multiplied by the number of hops a packet can make, to obtain a bound on latency of routing. This yields the following estimate

$$\frac{4k(n-1)}{n-k} \cdot \frac{n-1}{k-1} \leq \frac{8(n-1)^2}{n-k} \leq 16(n-1)\,, \tag{7.24}$$

assuming $2k \leq n+1$. This bound accounts for a full cycle of activity of all the groups, but a packet may spend another such cycle waiting for the group into which is got injected to become active. This means that $32 \cdot n$ is a bound on latency, restricted to packets that can be accounted for as injected subject to the injection-rate restriction.

Next, we estimate the delay of packets that need the adversary's burstiness to account for their injection.

As we have presented above in the equation (7.23), $\frac{2(n-1)k}{n-k}$ is the number of rounds needed for packets that can be accounted for as injected subject to the injection-rate restriction. Out of the total duration length of $\delta = \frac{4(n-1)k}{n-k}$ rounds, what remains are $\frac{2(n-1)k}{n-k}$ rounds that can be used to transmit a surplus of packets due to the burst of injections.

Out of these many rounds, at most k can be wasted because the token visits stations without packets. What remains are at least

$$\frac{2(n-1)k}{n-k} - k\,, \tag{7.25}$$

rounds.

Observe that if $2k \leq n+1$ then $\frac{2(n-1)}{n-k} \geq 4$, and so the quantity (7.25) is at least $3k$. The packet delay of the considered packets is thus at most $\frac{2\beta}{3k}$ multiplied by the number of groups, which is at most the following:

$$\frac{2\beta}{3k} \cdot \left\lceil \frac{n-1}{k-1} + 1 \right\rceil \leq \frac{2\beta}{3k} \cdot \frac{n+k-2}{k-1} \leq \frac{2\beta}{3k} \cdot \frac{\frac{3}{2}n-2}{k-1} \leq \beta n\,. \tag{7.26}$$

The bound on latency is the maximum of the partial upper bounds $32n$ and βn.

Next we give an impossibility result which demonstrates that the bound on injection rate in Theorem 13 is very close to optimal.

7.7.2 Throughput of channel-oblivious indirect routing

Fact 7. *For each n and $k < n$, a k-channel-oblivious routing algorithm is unstable against adversaries with injection rates greater than $\frac{k}{n}$.*

Proof If a station is switched on in a round then this contributes one *station-round*. A range of consecutive rounds τ of $|\tau| = t$ rounds can contribute at most kt station-rounds. By the double-counting principle, there is some station v which is switched on for at most $\frac{kt}{n}$ rounds during these t rounds. The adversary with injection rate ρ can inject at least ρt packets into station v during these rounds. Even if v transmits successfully in each round in τ then it can transmit at most $\frac{kt}{n}$ packets. If $\rho > \frac{k}{n}$ then there remain at least these many packets that need to be queued by v:

$$\rho t - \frac{kt}{n} = t\left(\rho - \frac{k}{n}\right). \tag{7.27}$$

This number can be made arbitrarily large for a suitably large t.

Corollary 2. *There is no k-channel-oblivious universal algorithm when $k = c \cdot n$, for a constant $c < 1$.*

Proof By Theorem 7, a k-channel-oblivious algorithm is unstable for injection rates that are greater than the ratio $\frac{k}{n} = c < 1$.

7.8 Channel-oblivious direct routing

Let an integer $k < n$ denote the channel restraint. Now we present a plain-packet k-channel-oblivious algorithm that routes packets directly.

7.8.1 Plain-packet algorithm k-Clique

It is called k-Clique. There are up to k stations switched on in each round. We assume that k is even and divides $2n$, to simplify the notation. The stations are partitioned into $\frac{2n}{k}$ disjoint sets of size $\frac{k}{2}$ each. These sets are combined in $\frac{n}{k}(\frac{2n}{k} - 1)$ *pairs* of size k each. There are at least 3 pairs,

assuming $\frac{k}{2} \leq \frac{n}{3}$. If $\frac{k}{2} > \frac{n}{3}$ then we can decrease k by keeping fewer stations switched on, so that the inequality $k \leq \frac{2n}{3}$ holds.

In each round t of an execution, all the stations in some pair are switched on, with the other stations switched off; we say that the pair is *active* in round t. The pairs are arranged into a virtual cycle to assign them the rounds of activity in a Round-Robin manner. A pair is active for one round at a time, and then the next pair takes over. When a pair is active then its stations execute an algorithm based on the principle of broadcasting algorithm OF-RRW. A station that has the token transmits all the old packets whose destinations are among the stations that make up the pair.

Theorem 14. *If algorithm* k-Clique *is executed against a* (ρ, β)*-adversary then it has a bounded latency for injection rates* $\rho < \frac{k^2}{n(2n-k)}$*. Moreover if the injection rate* ρ *is at most* $\frac{k^2}{2n(2n-k)}$ *the latency is at most* $8\frac{n^2}{k}(1+\frac{\beta}{2k})$ *.*

Proof Let $m = \frac{n(2n-k)}{k^2}$ be the number of pairs. We will use the bound $m \leq 2\frac{n^2}{k^2}$. A strategy for the adversary that maximizes queues and latency works by injecting packets into one pair with destinations in the same pair as well. Since a pair is allotted one round out of a segment of rounds equal to the number of pairs, an injection rate needs to be less than the inverse of the number of pairs, which is $\frac{1}{m} = \frac{k^2}{n(2n-k)}$. For each pair, when time is scaled only to the rounds which are assigned for the pair to execute OF-RRW, the injection rate is less than 1, so the algorithm has bounded latency.

Suppose that the inequality $\rho < \frac{1}{m}$ holds. A pair of k stations operating in isolation, and with time scaled only to the rounds assigned to the pair to be active, would have effective injection rate $m \cdot \rho$ and so its latency would be at most the following

$$\frac{2}{1-m\rho} \cdot k + \beta(1+m\rho) \leq \frac{2}{1-m\rho} \cdot k + 2\beta \ , \tag{7.28}$$

after applying the known bound on broadcast latency of OF-RRW derived in Theorem 1 in [ACKR19]. The bound needs to be increased by a multiplicative factor of m, since a pair operates in one round only in a segment of m rounds. This gives the following estimate on latency:

$$\frac{2m}{1-m\rho} \cdot k + 2\beta m \leq \frac{2n^2}{k} \cdot \frac{2}{1-m\rho} + \frac{4\beta n^2}{k^2} \ , \tag{7.29}$$

which holds for any injection rate satisfying $\rho < m^{-1}$. Assuming additionally that the inequality

$$\rho n(2n - k) \leq \frac{k^2}{2} \tag{7.30}$$

holds, we can use the following estimate:

$$\frac{2}{1 - m\rho} = \frac{2}{1 - \rho\frac{n(2n-k)}{k^2}} = \frac{2k^2}{k^2 - \rho n(2n - k)} \leq 4 \, . \tag{7.31}$$

This yields $8\frac{n^2}{k}(1 + \frac{\beta}{2k})$ as a bound on latency.

7.8.2 General algorithm k-Subsets

We describe a direct-routing k-channel-oblivious algorithm achieving the throughput $\frac{k(k-1)}{n(n-1)}$ for any burstiness β. The algorithm is called k-Subsets. It uses algorithm Move-Big-To-Front (MBTF) described in Section 7.2 as a subroutine. MBTF provides stability for injection rate 1 with any burstiness, for a multiple access channel *without* any channel restraint.

Let us fix enumeration of all k-element subsets of set $\{1, 2, 3, \ldots, n\}$ in order: $A_0, \ldots, A_{\gamma-1}$, where $\gamma = \binom{n}{k}$. For an integer i such that $0 \leq i \leq \gamma - 1$ and an integer $j \geq 0$, the rounds of the form $i + j\gamma$ make *thread* i. Algorithm MBTF operates in γ instantiations corresponding to the threads. Each such an instantiation has a dedicated queue in every station. The stations in A_i are active during thread i and process packets assigned to this thread. An execution is structured into *phases*, each of length γ, such that each thread has one round in a phase. A packet injected during a phase j is treated by an instantiation of MBTF that handles it as if it were injected "at round j." These specifications mean that the algorithm is k-channel-oblivious.

At the beginning of a phase, a station v assigns all the packets it received in the previous phases to the threads in the following manner. For each station w, station v keeps track of the numbers $x_0(w), \ldots, x_{\gamma-1}(w)$, which represent the respective numbers of packets addressed to w and already allocated by v to threads $0, \ldots, \gamma - 1$. Station v allocates the packets addressed to w that it received in the previous phase such that the resulting allocation is as balanced as possible, subject to the constraint that a packet addressed to w can be allocated to a thread i only if both stations v and w are in the set A_i. The algorithm routes directly, since when a packet is heard transmitted in a round assigned to a thread i, the receiver is switched on by virtue of belonging to A_i.

Obtaining balancing allocations implies that we make the numbers $x_0(w), \ldots, x_{\gamma-1}(w)$ differ by at most 1 at the beginning of each phase.

Theorem 15. *For each $k < n$, algorithm k-Subsets is stable against adversaries with injection rate $\frac{k(k-1)}{n(n-1)}$ and the number of queued packets is at most $2\binom{n}{k}(n^2+\beta)$ in every round.*

Proof Let $\lambda = \frac{k(k-1}{n(n-1)}$ denote the injection rate we consider. Suppose that the algorithm is not stable for injection rate λ, to arrive at a contradiction. There exists a thread i in which some queue corresponding to packets arriving at station v with address w and assigned to this thread grows unbounded. Since algorithm MBTF is stable for injection 1 and a fixed burstiness, there exists an infinite sequence of rounds $t_1, t_2, \ldots, t_j, \ldots$, for all $j \geq 1$, such that the number of packets from station v to w that get assigned to thread i by round j is at least $|t_j|/\gamma + j + 2$. Indeed, each thread is executed once every γ rounds, so the execution of algorithm MBTF in thread i would be stable if burstiness were bounded.

The algorithm allocating packets to threads guarantees that the number of packets with the same pair (v, w), of the source v and destination w, assigned by v to threads with stations v and w being active is almost balanced in each time period, in that the difference between any two of them is either -1 or 0 or 1. A thread with this property is determined by the stations different from v and w, so their number equals the following:

$$\binom{n-2}{k-2} = \binom{n}{k} \cdot \frac{k(k-1}{n(n-1)} = \gamma\lambda \ . \tag{7.32}$$

The number of packets injected to v addressed to w by round t_j, for every $j \geq 1$, is at least $(t_j/\gamma + j + 2) - 1$ multiplied by the number of threads handling packets from v to w. It follows that the following is a lower bound on the number of packets addressed to w that are assigned by v to some threads by round t_j:

$$\lambda t_j + \gamma\lambda(j+1) = \lambda(t_j + \gamma) + \gamma\lambda j \ . \tag{7.33}$$

Therefore, the number of packets that arrive at v by round $t_j + \gamma$ is at least $\lambda(t_j + \gamma) + \gamma\lambda j$. This contradicts the restrictions on the adversary for sufficiently large j, as the burstiness would be exceeded by round $t_j + \gamma$, and completes showing stability. A bound on the number of queued packets follows from the respective bound for algorithm MBTF given in [CKR09], which is applied independently for each thread.

It may occur in an execution of algorithm k-Subsets that some packets never get delivered and so remain queued forever. The algorithm can be modified to prevent this as long as injection rates are less than $\frac{k(k-1}{n(n-1)}$.

Namely, it is sufficient to replace algorithm MBTF, used as a procedure in k-Subsets, by Round-Robin-Withholding (RRW), see Section 7.2. The resulting algorithm is stable for any injection rate less than $\frac{k(k-1)}{n(n-1)}$ and achieves bounded latency. By the performance bounds of RRW, see Theorem 2 in [ACKR19], the latency is $\Theta(\gamma \cdot (n+\beta))$, for a fixed adversary with injection rate less than $\frac{k(k-1)}{n(n-1)}$. The latency bound is at least $\gamma = \binom{n}{k}$, which is exponential in n when k is linear in n.

7.8.3 Throughput of channel-oblivious direct routing

We give a matching lower bound on throughput, which demonstrates that the throughput in Theorem 15 is maximum achievable in the class of channel-oblivious algorithms.

Theorem 16. *For each integer n and $k < n$, and for any k-channel-oblivious algorithm routing directly, and for every adversary with an injection rate greater than $\frac{k(k-1)}{n(n-1)}$, some executions of the algorithm may be unstable against this adversary.*

Proof We will count the following quantities: for each ordered pair (x, y) of different stations x and y, if they are switched on simultaneously in a round then this rounds contributes one *station-pair round*. A range of consecutive rounds τ of $|\tau| = t$ rounds can contribute at most $k(k-1)t$ station-pair rounds. By the double-counting principle, there is some ordered pair of stations (w, z) such that w and z are switched on together for at most $\frac{k(k-1)}{n(n-1)} \cdot t$ rounds in the range of consecutive rounds τ. The adversary with injection rate ρ can inject at least ρt packets into station w during these rounds. Let all these packets be destined for z. Even if w transmits successfully in each round in τ such that w is switched on along with z, then it can transmit at most $\frac{k(k-1)}{n(n-1)} \cdot t$ packets. If $\rho > \frac{k(k-1)}{n(n-1)}$ then there remain at least these many packets that need to be queued by w:

$$\rho t - \frac{k(k-1)}{n(n-1)} t = t(\rho - \frac{k(k-1)}{n(n-1)}) \, . \tag{7.34}$$

This number can be made arbitrarily large for a suitably large t.

Chapter 8

Average case analysis on MAC

In order to be able to compare various approaches (e.g., stochastic and adversarial input) in a fair way, we propose to pursue average-case analysis, which is a novel approach in case of adversarial packet arrivals. In this chapter we start from looking into the Little's Law adaptation to the multiple access channel. This law defines the relation between the average system packet latency, queue size and the rate of adversary injections. Later we present the technique to study the algorithm average case performance on three known in literature algorithms.

8.1 Previous and related work

Adversarial queueing on MAC. To the knowledge of author, previous studies on adversarial queueing on multiple access channel were focused on providing lower and/or upper bounds on algorithms or algorithms classes performance. For instance, Bender et al. in [BFCH+05] use adversarial queueing to study the worst case of Backoff protocols on MAC. Similarly Chlebus et al. in [CKR06] have defined an adversary to study with greater precision the bounds on the performance of MAC algorithms classes.

More recently, Aldawsari et al. [ACK19] utilise adversarial approach as the framework to study worst-case latency scenario of ad-hoc broadcast algorithms. Garncarek et al. [GJK18] investigate the worst case stability of deterministic algorithms against adversarial packet injection. Bender et al. [BKPY18] use adversarial packet arrivals to prove the worst-case performance of the new version of Backoff protocol developed by authors.

Chlebus et al. [CKR09, CKR12] have developed deterministic algorithms now considered classical: Round-Robin-Withholding (RRW), Move-Big-To-Front (MBTF) and Search-Round-Robin (SRR). Those algorithms

became classical, as they were among the first algorithms with their proven adversarial worst-case performance being asymptotically optimal for respective classes. Recently, in the context of MAC channel jamming investigation, Anatharamu et al. [ACKR19] provided upper bounds on average system queue sizes and latency for those algorithms.

In the current chapter, we introduce the concept of the average-case adversarial analysis, which can be seen as the special case of the worst case analysis. In practice, medium access control algorithms are applied in specific environments (i.e. underwater acoustic cables, satellite communication, etc.), and we believe that those environments can be represented by the range of adversarial strategies. This way, with the proposed average-case analysis technique, researches gain the ability to compare algorithms by their practical applicability to specific areas – not only by the worst case performance which might or might not be occurring in the real environment. We apply the proposed technique to deterministic broadcast algorithms RRW, MBTF and SRR.

Little's Law. Little's Law was first formulated in 1961 by Little [Lit61] in the context of stochastic queueing. The law has stated that the average number of items in a queuing system is equal to the average arrival rate of items to the system multiplied by the average latency of an item in the system. Numerous law extensions and their applications were summarised in more recent works by Whitt [Whi91], Wolff [Wol10] and Little [Lit11] .

Those authors distinguish the following main developments: the generalisation of the law, stationary version of Little's Law, Rate Conversation Law and the distributional form of Little's Law.

The generalisation of the law involves associating of items in the queue with specific weights (e.g. Brumelle [Bru71], Heyman et al. [HS80]).

In Franken [FKAS82] and Miyazawa [Miy77] authors study a stationary distribution of the size of queues (and some other parameters) in various settings.

Note that in contrary to existing works, our contribution lifts the assumption on existence of average arrival rate of items. Therefore our formulation of the Little's Law is both – more general in the aspect of packets arrival and novel to the field.

The author of the last survey is John Little. He has summarized most important achievements related to the law named after him in the period of over 50 years.

8.2 Little's Law for adversarial queueing on MAC

For any prefix of t rounds, in which m packets were injected into the system, we use the following notation:

- Time-prefix injection rate is defined as $\psi_t = \frac{m}{t}$;
- Time-prefix average system queue size Q_t is the average number of packets in the all queues up to round t;
- Time-prefix average system latency L_t is the average number of rounds starting from injection and ending at minimum of round t and the number of the round of a transmission of a given packet over all m packets injected up to round t.

Lemma 12. *For any prefix of t rounds in which m packets were injected into the system, time-prefix injection rates are a quotient of the time-prefix average queue size over the time-prefix average latency: $\psi_t = \frac{Q_t}{L_t}$.*

Proof Consider a prefix of t rounds with m packets injected into the system within this time. We observe the execution: set $e_{ij} = 1$ if the i-th arrived packet was in the queue of some station in the system in the j-th round; $e_{ij} = 0$ otherwise, for $i \in \{1, 2, 3, \ldots, m\}$ and $j \in \{1, 2, 3, \ldots, t\}$.

The time-prefix latency is the sum of rounds spent in queue by each packet, divided by the number of packets m:

$$L_t = \frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{t} e_{ij} \ . \tag{8.1}$$

The time-prefix system queue size is the sum of all packets available in all queues divided by the time-prefix length t:

$$Q_t = \frac{1}{t} \sum_{j=1}^{t} \sum_{i=1}^{m} e_{ij} \ . \tag{8.2}$$

It follows that $L_t \cdot m = Q_t \cdot t$. Finally, $\psi_t = \frac{m}{t} = \frac{Q_t}{L_t}$.

Let us consider an infinite execution, wherein infinite number of packets are injected. We denote the system lower average queue size and the upper average queue size as, accordingly:

$$\underline{Q}_A = \liminf_{t \to \infty} Q_t \ ; \quad \overline{Q}_A = \limsup_{t \to \infty} Q_t \ . \tag{8.3}$$

Similarly, the lower average latency and the upper average latency are:

$$\underline{L}_A = \liminf_{t \to \infty} L_t \ ; \quad \overline{L}_A = \limsup_{t \to \infty} L_t \ . \tag{8.4}$$

Note that $\underline{L}_A$ is always at least 1, since we need at least one round for transmitting an injected packets.

For a case when the system lower average queue size equals the upper one, $\underline{Q}_A = \overline{Q}_A$, and when the system lower average latency equals the upper one, $\underline{L}_A = \overline{L}_A$, we define the system average queue size and, accordingly, the system average latency, as follows: $Q_A = \lim_{t \to \infty} Q_t \ ; \ L_A = \lim_{t \to \infty} L_t \ .$

Lemma 13. *The limit of the time-prefix injection rates ψ_t, with t going to infinity, is bounded by the adversary injection rate ρ from above and by the minimal injection pace ν from below:*

$$\nu \leq \liminf_{t \to \infty} \psi_t \leq \limsup_{t \to \infty} \psi_t \leq \rho \ . \tag{8.5}$$

Proof The middle inequality is obvious. Note that, the adversary can inject at most β additional packets into the system, while keeping injection rates at the highest available rates ρ:

$$\limsup_{t \to \infty} \psi_t \leq \limsup_{t \to \infty} \frac{\rho t + \beta}{t} = \rho \ . \tag{8.6}$$

Adversary injection rates cannot drop below the minimal injection pace ν for any prefix of t rounds, by the adversary definition. It follows that

$$\liminf_{t \to \infty} \psi_t \geq \liminf_{t \to \infty} \frac{\nu t}{t} = \nu \ . \tag{8.7}$$

Fact 8. *Properties of limit superior and limit inferior imply:*

$$\liminf_{t \to \infty} \psi_t \leq \frac{\underline{Q}_A}{\overline{L}_A} \ ; \quad \limsup_{t \to \infty} \psi_t \geq \frac{\overline{Q}_A}{\underline{L}_A} \ . \tag{8.8}$$

Theorem 17. Little's Law *for adversarial queueing on MAC: the relation of the lower/upper average latency to the queue size of an algorithm A, run under an adversary with injection pace ν and rate ρ, is:*

$$\nu \leq \frac{\underline{Q}_A}{\overline{L}_A} \leq \frac{\overline{Q}_A}{\underline{L}_A} \leq \rho \ . \tag{8.9}$$

Proof The proof follows in part the classic approach, c.f.,[BG92]. Note however that we need a more subtle argument to handle the case of **adversarial** behavior. By combination of Lemma 13 and Fact 8, the first and the last inequalities hold:

$$\nu \leq \liminf_{t \to \infty} \psi_t \leq \frac{\overline{Q}_A}{\underline{L}_A} \; ; \quad \frac{\overline{Q}_A}{\underline{L}_A} \leq \limsup_{t \to \infty} \psi_t \leq \rho \; . \tag{8.10}$$

The in-between inequality follows from the properties of limit inferior and limit superior, and the fact that $\underline{L}_A \geq 1$.

Corollary 3. *For an adversary strategy, such that $\rho = \nu$, the injection rate is equal to the ratio of the system average queue size and the average latency: $\nu = \rho = \lim_{t \to \infty} \psi_t = \frac{Q_A}{L_A}$.*

8.3 Deterministic Broadcast Algorithms

We study the system average queues for selected protocols.

Round-Robin-Withholding (RRW) [ACKR19, CKR12] is a full-sensing[†] algorithm operating in round-robin fashion, that is – stations gain access to the channel in the cyclic order of their identities. There is only one station with a right to transmit, which is said to hold a conceptual token. Once the station receives the token, it withholds the channel to unload all the packets in its queue. A silent round is a signal for the next station, in the cyclic order of identities, to take over the conceptual token.

We design adversary strategy to counter RRW algorithm. Adversary strategy is coherent with the definition of the (ρ, β)-adversary, see Section 4.1. The strategy is called Run-Robin-Run. Consider first $\rho > 1/2$. We assume that the adversary injects packets with rate/pace $\rho = \nu$ and burstiness $\beta \geq 1$ in the following manner. The k-th packet is injected in the round $t_k = \min_t\{t : \rho t \geq k\}$ to a *target station* specified below.

The target station is changed in time in a following way. At the beginning the last station is the target station. When the target station receives the token according to the RRW protocol, the status of target station is transferred to its predecessor (in the cyclic order of identities) and we say that a new *pass* starts.

[†]In this chapter we follow algorithm classification from our original paper [HKK21a]. For the details of how Adaptive and Full-sensing classes map to the newly introduced taxonomy, see Chapter 5.

We simulate an execution of RRW algorithm against Run-Robin-Run adversary strategy. See Figure 8.1 for simulation results compared to visualised upper bounds derived in Fact 9.

Fact 9. *[ACKR19] Upper bounds on queue sizes and packet latencies for* RRW *are* $O\left(\frac{\rho n}{1-\rho}+\beta\right)$ *and* $O\left(\frac{n}{(1-\rho)^2}+\frac{\beta}{1-\rho}\right)$*, respectively.*

Theorem 18. *Algorithm* RRW *has* $Q_A = \Theta(\frac{\rho n}{1-\rho})$ *and* $L_A = \Theta(\frac{n}{1-\rho})$ *against Run-Robin-Run* (ρ, β)*-adversary strategy, for any* $0 < \rho = \nu < 1 \leq \beta$*.*

Proof Let $q_0 = 0$ be the number of packets in stations' queues at the beginning of the protocol, and iteratively, for $i \geq 1$, let q_i be the number of packets at the beginning of the i-th subsequent pass. Note that the first pass starts after the first $n-1$ rounds of the execution, during which no packet is transmitted. Thus, $q_1 = \rho(n-1)$, because exactly $\rho(n-1)$ packets can be injected by the adversary during the first $n-1$ rounds. Let us consider q_i, for $i > 1$. One can observe that the $(i-1)$-st pass takes $q_{i-1}+n-1$ rounds: q_{i-1} transmitting rounds followed by $n-1$ silent round when the token is advanced through consecutive stations (in the cyclic order). Thus, $q_i = \rho(q_{i-1}+n-1)$. One can see, e.g., by induction, that

$$q_i > \frac{\rho(n-1)(1-\rho^i)}{1-\rho} - \frac{1}{1-\rho}\,. \tag{8.11}$$

Assuming sufficiently big n and due to assumed $\rho > 1/2$, we get the lower bound $q_i > \frac{n-1}{2(1-\rho)}$ on the number of packets at stations' queues at the beginning of i-th pass. Let us generalize this bound to *any* round in the i-th pass. At the beginning of the pass there are at least $\frac{n-1}{2(1-\rho)}$ packets, thus transmitting half of them takes at least $\frac{n-1}{4(1-\rho)}$ rounds. Since we consider the case $\rho > 1/2$, the adversary injects at least $\frac{n-1}{8(1-\rho)}$ packets during that period. Note that the newly injected packets are to be transmitted in the next pass. This means that in every single round of the pass there are at least $\frac{n-1}{8(1-\rho)}$ packets, and consequently $Q_A \geq \frac{n-1}{8(1-\rho)}$, which is $\Theta(\frac{n}{1-\rho})$ by Fact 9, which is $\Theta(\frac{\rho n}{1-\rho})$ since $1/2 < \rho < 1$. The estimate $\Theta(\frac{n}{1-\rho})$ on L_A follows directly from Corollary 3.

Note that in the remaining case of $0 < \rho \leq 1/2$, the same strategy of the adversary as in the proof of the previous case gives $Q_A = \Theta(\rho n)$ and $L_A = \Theta(n)$, which are asymptotically equal to $\Theta(\frac{\rho n}{1-\rho})$ and $\Theta(\frac{n}{1-\rho})$, respectively, due to the assumption $0 < \rho \leq 1/2$.

Move-Big-To-Front (MBTF) [ACKR19, CKR09] is an adaptive algorithm maintaining a dynamic list of all stations in private memory of each

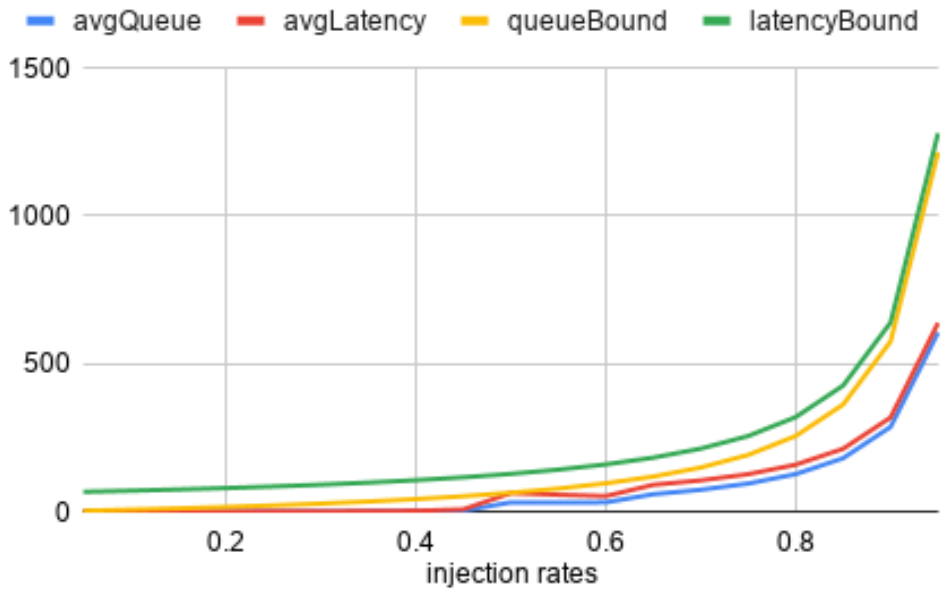


Figure 8.1: RRW protocol average queue and latency compared to the theoretical bounds by injection rates, after 10 mln rounds for a system size 64 against the respective adversary strategy.

station. Such a list is initialized at each station to have all the names of stations arranged in the increasing order: $0, 1, 2 \ldots, n-1$. The local lists are manipulated in the same way by all stations, based on channel feedback, hence they are identical copies of each other. The algorithm schedules exactly one station to transmit in a round, so that collisions never occur. This is implemented by having a conceptual token traversal through the stations, which is initially assigned to the first station on the list. A station with the token broadcasts a packet, if it has any, otherwise the round is silent. A station with token considers itself *big* in a round when it has at least n packets; it attaches a control bit to every packet it transmits to indicate its *big* status. A *big* station is moved to the front of the list and it takes the token with it. If a station that is not *big* transmits in a round, or when it pauses due to a lack of packets while holding the token (so that the round is silent), then the conceptual token is (virtually) passed at the end of this round to the next station on the list (ordered in a cyclic fashion).

We design adversary strategy to counter MBTF algorithm. The strategy is called Avoid-Big. In the beginning of the execution the adversary chooses arbitrary $l = \rho \cdot (n-1)$ out of n stations as *target stations*, starting from station 1 and such that any two of them are of distance at least $\frac{n}{l} - 1$ in the cyclic order, with station 0 being non-target (it is feasible since

$l < n$). Adversarial strategy further consists of two stages. In the first stage packets are injected to the target stations until each of them has exactly $n - 1$ in its queue. In the second stage, the adversary keeps the number of packets in queues of target stations at level $n - 1$ or $n - 2$, by injecting an available packet to the last transmitting target station.

We simulate an execution of MBTF algorithm against Avoid-Big adversary strategy. See Figure 8.2 for simulation results compared to visualised upper bounds derived in Fact 10.

Fact 10. *[ACKR19] Upper bounds on queue sizes and packet latencies for* MBTF *are* $O(\rho n^2 + \beta)$ *and* $O\left(\frac{n^2}{1-\rho} + \frac{\beta}{1-\rho}\right)$, *respectively.*

Theorem 19. *Algorithm* MBTF *has* $Q_A = \Theta(\rho n^2)$ *and* $L_A = \Theta(n^2)$ *against Avoid-Big* (ρ, β)*-adversary strategy, for any* $\frac{2}{n} \leq \rho = \nu < 1 \leq \beta$.

Proof In the first stage of Avoid-Big adversarial strategy packets are injected to the target stations until each of them has exactly $n-1$ in its queue. We argue that it happens eventually. First, during this stage, MBTF behaves like round-robin algorithm, as there is no *big* station. Next, during $\frac{1}{\rho} \cdot n$ rounds, l (non-big) target stations can remove at most $\frac{1}{\rho} \cdot l$ packets form their queues, while at the same time the adversary injects $\rho \cdot \frac{1}{\rho} \cdot n > \frac{1}{\rho} \cdot l$ packets to them, thus first stage finishes in at most $\frac{1}{\rho} \cdot n \cdot l \cdot (n-1)$ rounds.

In the second stage, the adversary keeps the number of packets in queues of target stations at level $n-1$ or $n-2$, by injecting an available packet to the last transmitting target station; since the next target station is at least $\frac{n}{l} - 1$ positions ahead and $\rho \cdot \left(\frac{n}{l} - 1\right) \geq 1$, there is at least one packet available before the next target station transmits, which raises the queue level of the last transmitting target station back from $n - 2$ to $n - 1$. Thus, in any n rounds each target station cannot become big, transmits and receives one packet. The surplus packets available to the adversary are injected to (non-target) station 0. Hence, $Q_A \geq l \cdot (n-2) = \Omega(\rho n^2)$, since $\rho \geq \frac{2}{n}$. Thus, by Fact 10: $Q_A = \Theta(\rho n^2)$. Finally, by Corollary 3: $L_A = Q_A/\rho = \Theta(n^2)$.

Search-Round-Robin (SRR) [ACKR19, CKR12] is a full-sensing algorithm with collision detection proceeding as a systematic continuous search for the next station with packets to transmit. The search is binary over the list of stations' names. One instance of a full sweep through all the stations is called a *phase*. A phase starts with the interval $[0, n-1]$ representing

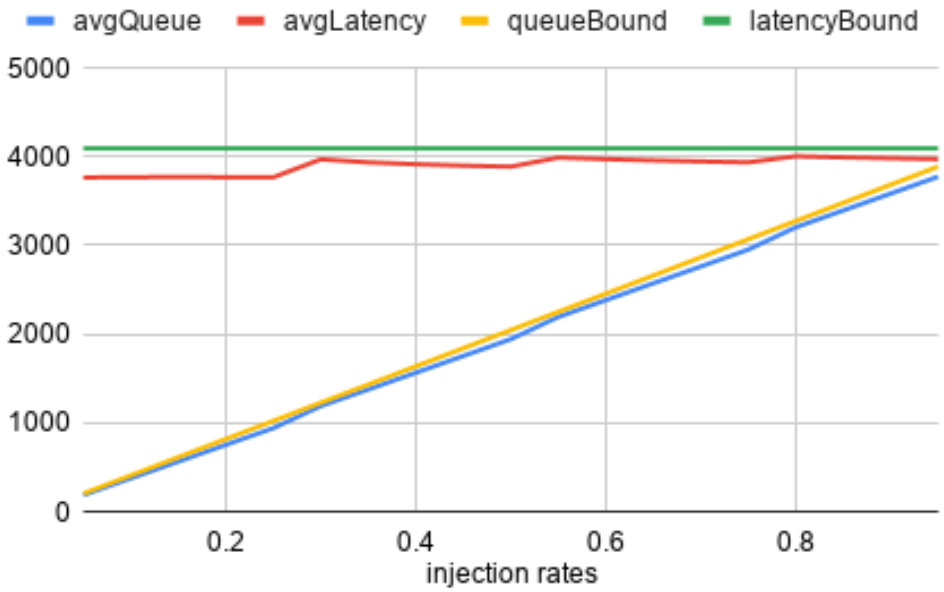


Figure 8.2: MBTF protocol average queue and latency compared to the theoretical bounds by injection rates, after 10 mln rounds for a system size 64 against the respective adversary strategy.

all the stations placed on the stack, and it ends when the stack becomes empty.

If a station with pending packets is identified by the search, the search is suspended while the station withholds the channel to transmit all its packets. After all the packets held by a station have been unloaded, a silent round follows, which triggers the search to be resumed.

A basic step in searching is to verify if there is a station with pending packets, whose name is in a given interval of integers. Such a step is accomplished by all the stations in the interval transmitting their packets. Every station receives the same feedback from the channel, whether it transmitted or not, thus all the stations know if the interval is empty (silence), or it contains a single station (packet heard), or it contains multiple stations (collision). A search for the next station is completed by a packet heard. A silence indicates that no station in the tested interval has packets and the interval is discarded. A collision results in having the interval partitioned into two halves of equal sizes, with one part processed immediately while the other one is pushed on a stack of the binary search protocol to wait. If a processed interval becomes empty or it is verified by silence that there is no station with packets in it, then a new interval is obtained by popping the stack.

We design adversary strategy to counter SRR algorithm. The strategy is called Sow-Discord. W.l.o.g. we can assume that both n and β are powers of 2. Sow-Discord adversary strategy operates as follows. In the first round, β packets are injected. The adversary divides the packets into two halves. The first $\beta/2$ packers are split into $\beta/4$ pairs. Each pair is injected into two consecutive stations, one packet per station. The pairs of stations are injected evenly over n stations, starting from the first station, labelled 0. More formally, the adversary injects one packet to each station with a label (in the binary form) from the set:

$$S = \{b_1 \dots b_l 00 \dots 0b : b_1, \dots, b_l, b \quad \{0,1\}\} ,$$

where $l = \log(\beta/4)$. The remaining $\beta/2$ packets are injected into the last station, labelled $n-1$.

We simulate an execution of SRR algorithm against Sow-Discord adversary strategy. See Figure 8.3 for simulation results compared to visualised upper bounds derived in Fact 11.

Fact 11. *[ACKR19] For $\rho \leq \frac{1}{2+\log n}$, queue sizes and packet latencies of SRR are $O(\beta)$ and $O(\beta \log n)$, respectively.*

Theorem 20. SRR *has $Q_A = \Theta(\rho\beta \log \frac{n}{\beta})$ and $L_A = \Theta(\beta \log \frac{n}{\beta})$ against Sow-Discord (ρ, β)-adversary strategy, for any $0 < \rho = \nu \leq \frac{1}{2+\log n}$, $\beta > 4$.*

Proof In contrast to previous protocols, here we first prove a lower bound for latency.

Consider the station labeled $n-1$. Observe that this station starts to transmit the remaining $\beta/2$ packets after all other packets are successfully transmitted. This takes $\beta/2 + \beta/4 \cdot (\log(n) - \log(\beta/4)) = \Theta(\beta \log n/\beta)$; i.e., at least half of injected packets need to wait at least $\Theta(\beta \log n/\beta)$ rounds. Combining it with upper bound in Fact 11, we get $L_A = \Theta(\beta \log n/\beta)$. By Corollary 3: $Q_A = \Theta(\beta\rho \log(n/\beta))$.

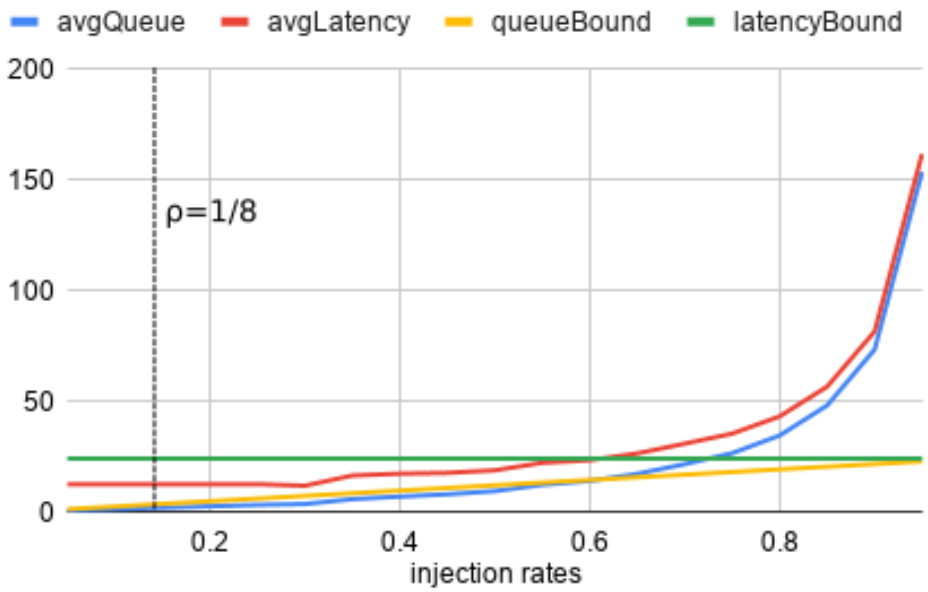


Figure 8.3: SRR protocol average queue and latency compared to the theoretical bounds by injection rates, after 10 mln rounds for a system size 64 against the respective adversary strategy; $\rho = \frac{1}{2+\log n} = 0.125$ stands for the threshold where the chosen bounds apply.

Chapter 9

Summary

In this thesis we have reviewed the recent history of the multiple access channel, together with the practically applied area of medium access control. We have followed through the existing system of algorithms classifications and comparisons. On top of the existing development, we provided a granular system capabilities-driven taxonomy of algorithms, allowing to further improve interoperability of different fields of science. Next, we summarised some known impossibilities for the algorithm capabilities defined by this taxonomy.

In the main part we expanded the existing multiple access channel model by the introduction of channel restraint – a constraint abstracting the use of power in the system. In this expanded model we constructed algorithms for channels of different capabilities. Those algorithms achieved throughput 1 for adaptive class, throughput 1 − (for any fixed) for full-sensing class, and sub-optimal throughput $\Theta(\frac{k}{n \log^2(n)})$ for acknowledgement based class. Rigid theoretical proof of algorithms correctness was complemented by comprehensive experiments. Those simulations confirmed the theoretical results and provided an insight into how new algorithms relate to the algorithms known in literature.

We have further expanded the multiple access channel model with the introduction of packet destination and the ability of stations to route packets indirectly. We have designed a number of algorithms for this model. One of the algorithms maintains bounded queues for the maximum injection rate 1 subject only to the channel restraint 3. We developed universal algorithms subject to the minimum channel restraint 2 that have the latency polynomial in the number of stations n. We constructed a k-channel-oblivious algorithm that has latency $O(n)$ for adversaries of injection rates less than $\frac{k-1}{n-1}$ and showed that there is no k-channel-oblivious stable algorithm against adver-

saries with injection rate greater than $\frac{k}{n}$. We gave a k-channel-oblivious algorithm routing directly that has latency $O\left(\frac{n^2}{k}\right)$ for adversaries of sufficiently small injection rates that are $O\left(\frac{k^2}{n^2}\right)$. We developed a k-channel-oblivious algorithm routing directly that is stable for injection rate $\frac{k(k-1)}{n(n-1)}$ and showed that no k-channel-oblivious algorithm routing directly is stable against adversaries with injection rates greater than $\frac{k(k-1)}{n(n-1)}$.

We adopted Little's Law to the context of adversarial queueing on multiple access channel. In this law, we proved inequalities holding for the case when the asymptotic average for packet arrivals into the system do not exist. With the use of this technique, we compared the worst-case performance bounds of algorithms known in literature to those achievable on average by specific adversary strategies. We have demonstrated the usefulness of the Little's Law for adversarial queueing on MAC, by applying it in the average case analysis technique.

Bibliography

[AAZZ13] M. Andrews, A. F. Anta, L. Zhang, and W. Zhao. Routing and scheduling for energy and delay minimization in the powerdown model. *Networks*, 61(3):226–237, 2013. [23]

[AC15] L. Anantharamu and B. S. Chlebus. Broadcasting in ad hoc multiple access channels. *Theoretical Computer Science*, 584:155–176, 2015. [21]

[ACK19] B. A. Aldawsari, B. S. Chlebus, and D. R. Kowalski. Broadcasting on adversarial multiple access channels. In *2019 IEEE 18th International Symposium on Network Computing and Applications (NCA)*, pages 1–4, 2019. doi:10.1109/NCA.2019.8935052. [22, 107]

[ACKR11] L. Anantharamu, B. S. Chlebus, D. R. Kowalski, and M. A. Rokicki. Medium access control for adversarial channels with jamming. In A. Kosowski and M. Yamashita, editors, *Structural Information and Communication Complexity*, pages 89–10, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. [23]

[ACKR19] L. Anantharamu, B. S. Chlebus, D. R. Kowalski, and M. A. Rokicki. Packet latency of deterministic broadcasting in adversarial multiple access channels. *Journal of Computer and System Sciences*, 99:27–52, 2019. doi:https://doi.org/10.1016/j.jcss.2018.07.001. [21, 22, 76, 77, 100, 103, 106, 108, 111, 112, 114, 116]

[ACR09] L. Anantharamu, B. S. Chlebus, and M. A. Rokicki. Adversarial multiple access channel with individual injection rates. In T. Abdelzaher, M. Raynal, and N. Santoro, editors, *Principles of Distributed Systems*, pages 174–188, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. [21]

[ACR17] L. Anantharamu, B. S. Chlebus, and M. A. Rokicki. Adversarial multiple access channels with individual injection rates. *Theory of Computing Systems*, 61(3):820–850, 2017. [21, 23, 27]

[AFZZ10] M. Andrews, A. Fernández, L. Zhang, and W. Zhao. Routing for energy minimization in the speed scaling model. In *Proceedings of the 29th IEEE International Conference on Computer Communications (INFOCOM)*, pages 2435–2443. IEEE, 2010. [23]

[AHJ20] B. A. Aldawsari and J. Haadi Jafarian. Improved deterministic broadcasting for multiple access channels. In K. Arai, S. Kapoor, and R. Bhatia, editors, *Intelligent Computing*, pages 645–660, Cham, 2020. Springer International Publishing. [22]

[Alb10] S. Albers. Energy-efficient algorithms. *Communications of the ACM*, 53(5):86–96, 2010. [23]

[AS19] K. L.-M. Ang and J. K. P. Seng. Application specific internet of things (asiots): Taxonomy, applications, use case and future directions. *IEEE Access*, 7:56577–56590, 2019. doi:10.1109/ACCESS.2019.2907793. [26]

[AWW05] I. F. Akyildiz, X. Wang, and W. Wang. Wireless mesh networks: a survey. *Computer Networks*, 47(4):445–487, 2005. doi:https://doi.org/10.1016/j.comnet.2004.12.001. [20]

[BCF+09] M. Blesa, D. Calzada, A. Fernández, L. López, A. L. Martínez, A. Santos, M. Serna, and C. Thraves. Adversarial queueing model for continuous network dynamics. *Theory of Computing Systems*, 44(3):304–331, Apr 2009. doi:10.1007/s00224-007-9046-1. [21]

[BFCH+05] M. A. Bender, M. Farach-Colton, S. He, B. C. Kuszmaul, and C. E. Leiserson. Adversarial contention resolution for simple channels. In *Proceedings of the 17th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 325–332. ACM, 2005. [21, 22, 107]

[BFGY16] M. A. Bender, J. T. Fineman, S. Gilbert, and 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, pages 636–654, Philadelphia, PA, USA, 2016. Society for Industrial and Applied Mathematics. URL http://dl.acm.org/citation.cfm?id=2884435.2884482. [23]

[BG92] D. P. Bertsekas and R. G. Gallager. *Data Networks, Second Edition*. Prentice Hall, 1992. [111]

[BGT+04] P. Bergamo, A. Giovanardi, A. Travasoni, D. Maniezzo, G. Mazzini, and M. Zorzi. Distributed power control for energy efficient routing in ad hoc setworks. *Wireless Networks*, 10(1):29–42, 2004. [24]

[BJKK18] M. Bienkowski, T. Jurdzinski, M. Korzeniowski, and D. R. Kowalski. Distributed online and stochastic queueing on a multiple access channel. *ACM Trans. Algorithms*, 14(2), May 2018. doi:10.1145/3182396. [21, 23]

[BKPY18] M. A. Bender, T. Kopelowitz, S. Pettie, and M. Young. Contention resolution with constant throughput and log-logstar channel accesses. *SIAM Journal on Computing*, 47(5):1735–1754, 2018. [107]

[BMÖG21] L. Beltramelli, A. Mahmood, P. Österberg, and M. Gidlund. Lora beyond aloha: An investigation of alternative random access protocols. *IEEE Transactions on Industrial Informatics*, 17(5):3544–3554, 2021. doi:10.1109/TII.2020.2977046. [26, 30]

[Bru71] S. L. Brumelle. On the relation between customer and time averages in queues. *Journal of Applied Probability*, 8(3):508–520, 1971. doi:10.2307/3212174. [108]

[Cap79] J. Capetanakis. Tree algorithms for packet broadcast channels. *IEEE Transactions on Information Theory*, 25(5):505–515, 1979. doi:10.1109/TIT.1979.1056093. [27]

[CCK20] B. S. Chlebus, V. Cholvi, and D. R. Kowalski. Universal stability in multi-hop radio networks. *Journal of Computer and System Sciences*, 114:48–64, 2020. doi:https://doi.org/10.1016/j.jcss.2020.05.009. [21, 22, 23]

[CDGZ20] T. Chen, J. Diakonikolas, J. Ghaderi, and G. Zussman. Hybrid scheduling in heterogeneous half- and full-duplex wireless networks. *IEEE/ACM Transactions on Networking*, 28(2):764–777, 2020. [26]

[CDH+18] Y. Chang, V. Dani, T. P. Hayes, Q. He, W. Li, and S. Pettie. The energy complexity of broadcast. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing (PODC)*, pages 95–104. ACM, 2018. [24]

[CGJK20] V. Cholvi, P. Garncarek, T. Jurdziński, and D. R. Kowalski. Optimal packet-oblivious stable routing in multi-hop wireless networks. In A. W. Richa and C. Scheideler, editors, *Structural Information and Communication Complexity*, pages 165–182, Cham, 2020. Springer International Publishing. [22]

[CGR02] M. Chrobak, L. Gasieniec, and W. Rytter. Fast broadcasting and gossiping in radio networks. *Journal of Algorithms*, 43(2):177 – 189, 2002. doi:https://doi.org/10.1016/S0196-6774(02)00004-4. [58]

[CHJ+19] B. S. Chlebus, E. Hradovich, T. Jurdziński, M. Klonowski, and D. R. Kowalski. Energy efficient adversarial routing in shared channels. In *The 31st ACM Symposium on Parallelism in Algorithms and Architectures*, pages 191–200, 2019. [12, 13, 75]

[CHJ+21] B. S. Chlebus, E. Hradovich, T. Jurdzinski, M. Klonowski, and D. R. Kowalski. Energy efficient adversarial routing in shared channels, 2021, 1810.11441. [13]

[Chl18] B. S. Chlebus. Randomized communication in radio networks. *CoRR*, abs/1801.00074, 2018, 1801.00074. URL `http://arxiv.org/abs/1801.00074`. [21]

[Cho20] J. Choi. On throughput of compressive random access for one short message delivery in iot. *IEEE Internet of Things Journal*, 7(4):3499–3508, 2020. doi:10.1109/JIOT.2020.2972519. [26]

[CK05] B. S. Chlebus and D. R. Kowalski. Almost optimal explicit selectors. In *FCT*, pages 270–280, 2005. [22, 57]

[CKP+17] Y.-J. Chang, T. Kopelowitz, S. Pettie, R. Wang, and 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, pages 771–783, New York, NY, USA, 2017. ACM. doi:10.1145/3055399.3055481. [24]

[CKR06] B. S. Chlebus, D. R. Kowalski, and M. A. Rokicki. Adversarial queuing on the multiple-access channel. In *Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*, pages 92–101, 2006. [22, 107]

[CKR09] B. S. Chlebus, D. R. Kowalski, and M. A. Rokicki. Maximum throughput of multiple access channels in adversarial environments. *Distributed Computing*, 22(2):93–116, 2009.

doi:10.1007/s00446-009-0086-4. [15, 17, 21, 22, 23, 27, 29, 30, 34, 55, 57, 75, 76, 79, 105, 107, 112]

[CKR12] B. S. Chlebus, D. R. Kowalski, and M. A. Rokicki. Adversarial queuing on the multiple access channel. *ACM Transactions on Algorithms (TALG)*, 8(1):1–31, 2012. [17, 21, 22, 27, 30, 107, 111, 114]

[CSB+08] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, and S. J. Wright. Power awareness in network design and routing. In *Proceeding of the 27th IEEE International Conference on Computer Communications (INFOCOM)*, pages 457–465. IEEE, 2008. [23]

[DBV17] A. De Bonis and U. Vaccaro. -almost selectors and their applications to multiple-access communication. *IEEE Transactions on Information Theory*, 63(11):7304–7319, 2017. doi:10.1109/TIT.2017.2750178. [22]

[De 19] A. De Bonis. New selectors and locally thin families with applications to multi-access channels supporting simultaneous transmissions. *Theoretical Computer Science*, 796:34–50, 2019. doi:https://doi.org/10.1016/j.tcs.2019.08.020. [22]

[DEA06] I. Demirkol, C. Ersoy, and F. Alagoz. Mac protocols for wireless sensor networks: a survey. *IEEE Communications Magazine*, 44(4):115–121, 2006. doi:10.1109/MCOM.2006.1632658. [20]

[DG20] M. Dibaei and A. Ghaffari. Full-duplex medium access control protocols in wireless networks: a survey. *Wireless Networks*, 26(4):2825–2843, May 2020. doi:10.1007/s11276-019-02242-w. [20]

[DV20] A. De Bonis and U. Vaccaro. A new kind of selectors and their applications to conflict resolution in wireless multichannels networks. *Theoretical Computer Science*, 806:219–235, 2020. doi:https://doi.org/10.1016/j.tcs.2019.03.034. [22]

[dVLR19] R. M. de Oliveira, A. B. Vieira, H. A. Latchman, and M. V. Ribeiro. Medium access control protocols for power line communication: A survey. *IEEE Communications Surveys Tutorials*, 21(1):920–939, 2019. [20]

[DWD+18] L. Dai, B. Wang, Z. Ding, Z. Wang, S. Chen, and L. Hanzo. A survey of non-orthogonal multiple access for 5g. *IEEE Communications Surveys Tutorials*, 20(3):2294–2323, 2018. doi:10.1109/COMST.2018.2835558. [20]

[DZ83] J. Day and H. Zimmermann. The osi reference model. *Proceedings of the IEEE*, 71(12):1334–1340, 1983. doi:10.1109/PROC.1983.12775. [19]

[FGKN16] J. T. Fineman, S. Gilbert, F. Kuhn, and C. Newport. Contention resolution on a fading channel. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*, PODC '16, pages 155–164, New York, NY, USA, 2016. ACM. doi:10.1145/2933057.2933091. [23]

[FKAS82] P. Franken, D. König, U. Arndt, and V. Schmidt. Queues and point processes. *JOHN WILEY & SONS, INC., ONE WILEY DR., SOMERSET, N. J. 08873, 1982, 230*, 1982. [108]

[GCNS08] C. Gunaratne, K. J. Christensen, B. Nordman, and S. Suen. Reducing the energy consumption of Ethernet with adaptive link rate (ALR). *IEEE Transactions on Computers*, 57(4):448–461, 2008. [23]

[GFK+17] J.-M. Gorce, Y. Fadlallah, J.-M. Kelif, H. V. Poor, and A. Gati. Fundamental limits of a dense iot cell in the uplink. In *2017 15th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, pages 1–6, 2017. doi:10.23919/WIOPT.2017.7959936. [22]

[GJK18] P. Garncarek, T. Jurdzinski, and D. R. Kowalski. Local Queuing Under Contention. In U. Schmid and J. Widder, editors, *32nd International Symposium on Distributed Computing (DISC 2018)*, volume 121 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 28:1–28:18, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.DISC.2018.28. [107]

[GJK19] P. Garncarek, T. Jurdzinski, and D. R. Kowalski. Stable memoryless queuing under contention. In *33rd International Symposium on Distributed Computing (DISC 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019. [22]

[GJKP00] L. A. Goldberg, M. Jerrum, S. Kannan, and M. Paterson. A bound on the capacity of backoff and acknowledgement-based protocols. In *International Colloquium on Automata, Languages, and Programming*, pages 705–717. Springer, 2000. [21]

[GJKP04] L. A. Goldberg, M. Jerrum, S. Kannan, and M. Paterson. A bound on the capacity of backoff and acknowledgment-based protocols. *SIAM Journal on Computing*, 33(2):313–331, 2004, https://doi.org/10.1137/S0097539700381851. doi:10.1137/S0097539700381851. [27]

[GKK+08] L. Gąsieniec, E. Kantor, D. R. Kowalski, D. Peleg, and C. Su. Time efficient *k*-shot broadcasting in known topology radio networks. *Distributed Computing*, 21(2):117–127, 2008. [24]

[Gol00] L. A. Goldberg. Notes on contention resolution, 2000. URL `http://www.cs.ox.ac.uk/people/leslieann.goldberg/contention.html`. [21, 27, 29]

[GRV20] L. Gargano, A. A. Rescigno, and U. Vaccaro. Low-weight superimposed codes and related combinatorial structures: Bounds and applications. *Theoretical Computer Science*, 806:655–672, 2020. doi:https://doi.org/10.1016/j.tcs.2019.10.032. [22]

[GS07] M. Gupta and S. Singh. Dynamic Ethernet link shutdown for energy conservation on Ethernet links. In *Proceedings of IEEE International Conference on Communications (ICC)*, pages 6156–6161. IEEE, 2007. [23]

[HK11] M. Herlich and H. Karl. Reducing power consumption of mobile access networks with cooperation. In *Proceedings of the 2nd International Conference on Energy-E cient Computing and Networking (e-Energy)*, pages 77–86. ACM, 2011. [24]

[HKK20] E. Hradovich, M. Klonowski, and D. R. Kowalski. Contention resolution on a restrained channel. In *2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 89–98. IEEE, 2020. [12, 17, 33, 56]

[HKK21a] E. Hradovich, M. Klonowski, and D. R. Kowalski. New view on adversarial queueing on mac. *IEEE Communications Letters*, 25(4):1144–1148, 2021. doi:10.1109/LCOMM.2020.3047997. [11, 25, 27, 111]

[HKK21b] E. Hradovich, M. Klonowski, and D. R. Kowalski. Restrained medium access control on adversarial shared channels, 2021, 1808.02216. [12]

[HLR96] J. Hästad, T. Leighton, and B. Rogoff. Analysis of backoff protocols for multiple access chan-

nels. *SIAM Journal on Computing*, 25(4):740–774, 1996, https://doi.org/10.1137/S0097539792233828. doi:10.1137/S0097539792233828. [15, 21, 27, 65, 67]

[HS80] D. P. Heyman and S. Stidham. The relation between customer and time averages in queues. *Operations Research*, 28(4):983–994, 1980, https://doi.org/10.1287/opre.28.4.983. doi:10.1287/opre.28.4.983. [108]

[IEEE802.11] Ieee standard for information technology–telecommunications and information exchange between systems local and metropolitan area network–specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications amendment 5: Preassociation discovery. *IEEE Std 802.11aq-2018 (Amendment to IEEE Std 802.11-2016 as amended by IEEE Std 802.11ai-2016, IEEE Std 802.11ah-2016, IEEE Std 802.11aj-2018, and IEEE Std 802.11ak-2018)*, pages 1–69, 2018. doi:10.1109/IEEESTD.2018.8457463. [20]

[IEEE802.15] Ieee recommended practice for information technology– telecommunications and information exchange between systems– local and metropolitan area networks– specific requirements part 15.5: Mesh topology capability in wireless personal area networks (wpans). *IEEE Std 802.15.5-2009*, pages 1–166, 2009. doi:10.1109/IEEESTD.2009.4922106. [20]

[IEEE802.16] Ieee standard for wirelessman-advanced air interface for broadband wireless access systems. *IEEE Std 802.16.1-2012*, pages 1–1090, 2012. doi:10.1109/IEEESTD.2012.6297413. [20]

[IEEE802.1D] Ieee standard for local and metropolitan area networks: Media access control (mac) bridges. *IEEE Std 802.1D-2004 (Revision of IEEE Std 802.1D-1998)*, pages 1–281, 2004. doi:10.1109/IEEESTD.2004.94569. [20]

[IEEE802] Ieee standard for local and metropolitan area networks: Overview and architecture. *IEEE Std 802-2014 (Revision to IEEE Std 802-2001)*, pages 1–74, 2014. doi:10.1109/IEEESTD.2014.6847097. [19, 20]

[IEEE802.22] Ieee recommended practice for information technology - telecommunications and information exchange between systems wireless regional area networks (wran) - specific requirements - part 22.2: Installation and deployment of ieee

802.22 systems. *IEEE Std 802.22.2-2012*, pages 1–44, 2012. doi:10.1109/IEEESTD.2012.6317267. [20]

[IEEE802.3] Ieee standard for management information base (mib) definitions for ethernet. *IEEE Std 802.3.1-2013 (Revision of IEEE Std 802.3.1-2011)*, pages 1–415, 2013. doi:10.1109/IEEESTD.2013.6572796. [20]

[Ind02] 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, page 697–704, USA, 2002. Society for Industrial and Applied Mathematics. [21, 22]

[IP05] S. Irani and K. Pruhs. Algorithmic problems in power management. *ACM SIGACT News*, 36(2):63–76, 2005. [23]

[Jia18] S. Jiang. State-of-the-art medium access control (mac) protocols for underwater acoustic networks: A survey based on a mac reference model. *IEEE Communications Surveys Tutorials*, 20(1):96–131, 2018. [20, 26]

[JKZ02] T. Jurdzinski, M. Kutylowski, and J. Zatopianski. Efficient algorithms for leader election in radio networks. In *PODC 2002*, pages 51–57, 2002. doi:10.1145/571825.571833. [24]

[JLB04] R. Jurdak, C. V. Lopes, and P. Baldi. A survey, classification and comparative analysis of medium access control protocols for ad hoc networks. *IEEE Communications Surveys Tutorials*, 6(1):2–16, 2004. doi:10.1109/COMST.2004.5342231. [26]

[Kel85] F. P. Kelly. Stochastic models of computer communication systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 47(3):379–395, 1985, https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/j.2517-6161.1985.tb01367.x. doi:https://doi.org/10.1111/j.2517-6161.1985.tb01367.x. [27]

[KG85] J. Komlós and A. Greenberg. An asymptotically fast nonadaptive algorithm for conflict resolution in multiple-access channels. *IEEE Transactions on Information Theory*, 31(2):302–306, 1985. [22]

[KKP13] M. Kardas, M. Klonowski, and D. Pajak. Energy-efficient leader election protocols for single-hop radio networks. In *ICPP 2013*, pages 399–408, 2013. doi:10.1109/ICPP.2013.49. [24]

[KKPS17] S. Karmakar, P. Koutris, A. Pagourtzis, and D. Sakavalas. Energy-efficient broadcasting in ad hoc wireless networks. *Journal of Discrete Algorithms*, 42:2–13, 2017. [24]

[KKZ12] M. Klonowski, M. Kutylowski, and J. Zatopianski. Energy efficient alert in single-hop networks of extremely weak devices. *Theor. Comput. Sci.*, 453:65–74, 2012. doi:10.1016/j.tcs.2012.01.044. [24]

[KL76] T. Kasami and S. Lin. Coding for a multiple-access channel. *IEEE Transactions on Information Theory*, 22:129–137, Mar. 1976. [21]

[KP16] E. Kantor and D. Peleg. Efficient k-shot broadcasting in radio networks. *Discrete Applied Mathematics*, 202:79–94, 2016. [24]

[KP18] M. Klonowski and D. Pajak. Brief announcement: Broadcast in radio networks, time vs. energy tradeoffs. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing (PODC)*, pages 115–117. ACM, 2018. [24]

[KS64] W. H. Kautz and R. C. Singleton. Nonrandom binary superimposed codes. *IEEE Trans. Inf. Theory*, 10(4):363–377, 1964. doi:10.1109/TIT.1964.1053689. [60]

[Lit61] J. D. C. Little. A proof for the queuing formula: L = λw. *Operations Research*, 9(3):383–387, 1961, https://doi.org/10.1287/opre.9.3.383. doi:10.1287/opre.9.3.383. [22, 108]

[Lit11] J. D. C. Little. Or forum—little's law as viewed on its 50th anniversary. *Operations Research*, 59(3):536–549, 2011, https://doi.org/10.1287/opre.1110.0940. doi:10.1287/opre.1110.0940. [108]

[MF21] M. D. Mendy and Y. Faye. A comparative study of wireless technologies coexistence mechanisms in iot: A survey. In S. Fong, N. Dey, and A. Joshi, editors, *ICT Analysis and Applications*, pages 173–181, Singapore, 2021. Springer Singapore. [20]

[Miy77] M. Miyazawa. Time and customer processes in queues with stationary inputs. *Journal of Applied Probability*, 14(2):349–357, 1977. URL `http://www.jstor.org/stable/3213005`. [108]

[MK10] G. D. Marco and D. R. Kowalski. Towards power-sensitive communication on a multiple-access channel. In *2010 IEEE 30th International Conference on Distributed Computing Systems*, pages 728–735, 2010. doi:10.1109/ICDCS.2010.50. [27]

[MS17] G. D. Marco and G. Stachowiak. Asynchronous shared channel. In E. M. Schiller and A. A. Schwarzmann, editors, *Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2017, Washington, DC, USA, July 25-27, 2017*, pages 391–400. ACM, 2017. doi:10.1145/3087801.3087831. [24]

[NPI+08] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, and D. Wetherall. Reducing network energy consumption via sleeping and rate-adaptation. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design & Implementation (NSDI)*, pages 323–336. USENIX Association, 2008. [24]

[ORS+18] A. Ogierman, A. Richa, C. Scheideler, S. Schmid, and J. Zhang. Sade: competitive mac under adversarial sinr. *Distributed Computing*, 31(3):241–254, Jun 2018. doi:10.1007/s00446-017-0307-1. [23]

[PS95] M. Paterson and A. Srinivasan. Contention resolution with bounded delay. In *Proceedings of IEEE 36th Annual Foundations of Computer Science*, pages 104–113, 1995. doi:10.1109/SFCS.1995.492467. [27]

[RDM83] I. Rubin and L. De Moraes. Message delay analysis for polling and token multiple-access schemes for local communication networks. *IEEE Journal on Selected Areas in Communications*, 1(5):935–947, 1983. doi:10.1109/JSAC.1983.1145983. [21]

[RLNJ06] Y. K. Rana, B. H. Liu, A. Nyandoro, and S. Jha. Bandwidth aware slot allocation in hybrid mac. In *Proceedings. 2006 31st IEEE Conference on Local Computer Networks*, pages 89–96, 2006. [19, 26]

[Ros02] A. Rosén. A note on models for non-probabilistic analysis of packet switching networks. *Information Processing Letters*, 84(5):237–240, 2002. [23]

[SLW18] M. F. Sohail, C. Y. Leow, and S. Won. Non-orthogonal multiple access for unmanned aerial vehicle assisted communication. *IEEE Access*, 6:22716–22727, 2018. doi:10.1109/ACCESS.2018.2826650. [20]

[SMK18] V. Sundararaj, S. Muthukumar, and R. Kumar. An optimal cluster formation based energy efficient dynamic scheduling hybrid mac protocol for heavy traffic load in wireless sen-

sor networks. *Computers & Security*, 77:277 – 288, 2018. doi:https://doi.org/10.1016/j.cose.2018.04.009. [26, 30]

[SMM20] E. Shayo, P. Mafole, and A. Mwambela. A survey on time division multiple access scheduling algorithms for industrial networks. *SN Applied Sciences*, 2(12):2140, Dec 2020. doi:10.1007/s42452-020-03923-4. [20]

[SNK+16] M. Sami, N. K. Noordin, M. Khabazian, F. Hashim, and S. Subramaniam. A survey and taxonomy on medium access control strategies for cooperative communication in wireless networks: Research issues and challenges. *IEEE Communications Surveys Tutorials*, 18(4):2493–2521, 2016. doi:10.1109/COMST.2016.2569601. [26]

[TDL+20] S. A. Tegos, P. D. Diamantoulakis, A. S. Lioumpas, P. G. Sarigiannidis, and G. K. Karagiannidis. Slotted aloha with noma for the next generation iot. *IEEE Transactions on Communications*, 68(10):6289–6301, 2020. doi:10.1109/TCOMM.2020.3007744. [20]

[TNS+19] A. B. Tambawal, R. M. Noor, R. Salleh, C. Chembe, M. H. Anisi, O. Michael, and J. Lloret. Time division multiple access scheduling strategies for emerging vehicular ad hoc network medium access control protocols: a survey. *Telecommunication Systems*, 70(4):595–616, Apr 2019. doi:10.1007/s11235-018-00542-8. [26, 30]

[TUZ01] A. Ta-Shma, C. Umans, and D. Zuckerman. Loss-less condensers, unbalanced expanders, and extractors. In J. S. Vitter, P. G. Spirakis, and M. Yannakakis, editors, *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 143–152. ACM, 2001. doi:10.1145/380752.380790. [59, 60]

[UAK+17] F. Ullah, A. H. Abdullah, O. Kaiwartya, S. Kumar, and M. M. Arshad. Medium access control (mac) for wireless body area network (wban): Superframe structure, multiple access technique, taxonomy, and challenges. *Human-centric Computing and Information Sciences*, 7(1):34, Dec 2017. doi:10.1186/s13673-017-0115-4. [26]

[Whi91] W. Whitt. A review ofl=λw and extensions. *Queueing Systems*, 9(3):235–268, Sep 1991. doi:10.1007/BF01158466. [108]

[Wol10] R. W. Wolff. Little's law and related results. *Wiley encyclopedia of operations research and management science*, 2010. [108]

[YAL+19] X. Yan, K. An, T. Liang, G. Zheng, Z. Ding, S. Chatzinotas, and Y. Liu. The application of power-domain non-orthogonal multiple access in satellite communication networks. *IEEE Access*, 7:63531–63539, 2019. doi:10.1109/ACCESS.2019.2917060. [20]

[YMG08] J. Yick, B. Mukherjee, and D. Ghosal. Wireless sensor network survey. *Computer Networks*, 52(12):2292–2330, 2008. doi:https://doi.org/10.1016/j.comnet.2008.04.002. [20]