

Universidad Rey Juan Carlos
Departamento de Sistemas Telemáticos y Computación

Universidad de Chile
Departamento de Ingeniería Matemática

Doctoral Thesis

Performance of scheduling policies and networks in generalized adversarial queueing models

Christopher Thraves Caro
cbthraves@gmail.com

Supervisors:

Antonio Fernández Anta
anto@gsyc.es

Marcos Kiwi
mkiwi@dim.uchile.cl

Móstoles

Dedicado a Helia Caro y Lawrence Thraves.

Esta tesis ha sido financiada por:

Beca de postgrado Mecesus, Ministerio de Educación del Gobierno de Chile a través del Departamento de Ingeniería Matemática, Universidad de Chile.

Centro de Modelamiento Matemático, Universidad de Chile.

Proyecto Anillo *Redes en Matemáticas y Ciencias de la Ingeniería*, Programa Bicentenario de Ciencia y Tecnología, Gobierno de Chile.

Proyecto *Nuevos Paradigmas Para el Diseño de Redes y Servicios de Comunicaciones*, Ministerio de Educación y Ciencia, Gobierno de España.

Proyecto *Sistemas Distribuidos Autónomos, Confiables y de Altas Prestaciones*, Comunidad Autónoma de Madrid.

Las ideas se pueden calificar de muchas formas. Pueden ser simples, bonitas, útiles, creativas, originales, pero sin duda las ideas más importantes son las geniales. El camino para llegar a una idea genial es largo, larguísimo, mucho más largo de lo que una juventud que se maravilla fácilmente te hace creer.

La calificación de una idea depende de la persona que evalúa. Una persona con experiencia en ideas puede darse cuenta fácilmente si la nueva evaluada tiene potencial o es sólo un globo lleno de aire. Por eso uno busca personas experimentadas en ideas para aprender, ya sea a distinguir las buenas de las malas, o a crear buenas ideas, mejor dicho para aprender que las buenas ideas se crean con mucho trabajo y no se tienen espontáneamente.

Yo en mi camino de aprendizaje me he maravillado fácilmente, las luces más pequeñas me hacen sentir el niño más feliz, hasta que me muestran que mis ideas se pueden apagar con un soplo y vuelta a empezar. Esta vez mis ideas no son geniales, pero conversar con mis directores de tesis me muestra formas para mejorarlas. Y vuelta a empezar. Por repetido el camino se hace familiar, pero como lo que se busca es transformar un fenómeno al lenguaje de las matemáticas es absolutamente necesario conocerlo mejor, siempre mejor.

Agradezco profundamente a Marcos Kiwi y Antonio Fernández por su experiencia compartida, por su paciencia y por el esfuerzo invertido en mi.

Agradecimientos

Después de estos cinco años hay muchas personas a las que me gustaría agradecer, personas que de una u otra forma se han hecho presentes. Por supuesto empiezo por mi hermana Cindy Thraves, es una de las más importantes por su eterna compañía, por nuestra amistad, y por el gran cariño. En la misma línea está Rafaela, su hija, que ha estado presente los últimos tres años. A mi papá y mi mamá que tienen mucho que ver con todo esto, han sido una gran influencia y determinantes en cómo soy y por qué estoy en el lugar en que estoy. A todos ustedes mil gracias de manera muy especial, las palabras siempre van a ser pocas para mostrarles todo mi cariño y agradecimiento.

Agradezco con mucho cariño a Antonio Fernández y Marcos Kiwi, mis directores de tesis. Ambos me han apoyado, acompañado y por sobre todo me han enseñado durante todo este tiempo. Cada uno en su estilo particular ha sido un gran aporte para mí, son y serán un gran ejemplo como personas y como científicos. Les agradezco por sobre todo su gran paciencia y sus grandes esfuerzos dedicados a mí.

Personas muy especiales a las que me gustaría agradecer por su amistad son todos mis primos. Con ellos he compartido desde la infancia, siempre han estado presentes y a lo largo del tiempo hemos generado lazos que a pesar de la distancia y el silencio no se rompen fácilmente. Muestra clara de esto es cada reencuentro que me llena de energía.

Me gustaría agradecer también a las personas con las que he compartido tiempo trabajando, tiempo que indudablemente a sido de gran provecho para mí. Especialmente a la gente de Madrid, ya sea de la Universidad Politécnica de Madrid o de la Universidad Rey Juan Carlos, y al grupo con el que he compartido en Barcelona de la Universitat Politècnica de Catalunya.

Les agradezco de manera efusiva a todos mis amigos, todos aquellos con los que he disfrutado ya sea en vivo o a la distancia, todos aquellos con los que he compartido interesantes proyectos como festivales de cortometrajes o fotografía. A mis amigos que me acompañan desde etapas anteriores que siguen presentes y seguirán para siempre.

Por supuesto que me gustaría agradecer a todos los miembros de los dos departamentos en los que estudié, DIM de la Universidad de Chile y GSyC de la Universidad Rey Juan Carlos, ambos lugares están llenos de personas con las que compartí mucho y de las que aprendí muchísimo más.

Finalmente hago mención a todas aquellas personas con las que me cruce durante este tiempo y de alguna manera fueron compañía y apoyo. A todos ustedes muchas gracias.

Resumen de tesis para optar al grado de Doctor en Ciencias de la Ingeniería mención Modelación Matemática. Realizada por Christopher Thraves con fecha 22/01/2008, profesores guía Marcos Kiwi y Antonio Fernández.

“Performance of scheduling policies and networks in generalized adversarial queueing models”

Esta tesis estudia problemas generados en redes de colas haciendo un análisis de peor caso. Para esto se basa en un modelo de adversario llamado *Adversarial Queueing Theory* (AQT). En esta tesis se presentan tres variantes de dicho modelo, cada una enfocada al estudio de distintos problemas generados en redes de colas; problemas motivados en redes de producción industriales o redes de comunicación similares a Internet. Además se hace una contribución directa a la versión continua del modelo AQT llamado *Continuous Adversarial Queueing Theory* (CAQT).

El modelo de AQT tiene como principales componentes una *red*, un *adversario* y una *política*. La red está representada por un grafo dirigido. El adversario inyecta carga en la red, representada por paquetes, decidiendo el momento de inyección, punto de inyección, camino que tiene que recorrer y destino de cada paquete, todo esto en el momento de la inyección. Cuando más de un paquete quiere cruzar un arco al mismo tiempo la política actúa como criterio para determinar qué paquete cruza primero, mientras que los paquetes restantes esperan en una cola asociada al arco. Todo esto ocurre en un periodo arbitrariamente largo de tiempo. Para que el adversario no sobrecargue la red trivialmente, éste es acotado en base a la capacidad de servicio de cada arco. Como parámetro de buen comportamiento para políticas y redes se define *estabilidad*, que de manera informal se puede entender como que es la propiedad de que el número de paquetes que todavía no han llegado a destino esté acotado por una constante que no depende del tiempo.

En el primer modelo presentado se analizan redes de colas generadas en servidores. En este modelo el adversario inyecta tareas representadas por un conjunto de procesos, y en donde la dependencia entre todos los procesos de cada tarea está dada por

una función Booleana, que para cada proceso que depende del estado de los mismos. De esta forma se da una condición suficiente para que dichas redes sean estables.

Un segundo modelo presentado, llamado *AQT with setups*, modela el problema cuando el adversario puede inyectar paquetes de distinto tipo, y cada arco tiene que pagar en tiempo de no trabajo cada vez que cambia el tipo de paquete que está sirviendo. Como primer resultado se presentan la equivalencia entre una versión general del modelo y una más restringida, lo que permite trabajar con la última sin perder generalidad en términos de estabilidad. Además se caracteriza el conjunto de todas las redes estables. En términos de políticas, se demuestra la existencia de una red y un adversario que impiden la existencia de políticas estables. Finalmente, se provee de una variante de fluidos que permite ocupar herramientas externas para demostrar estabilidad.

El aporte hecho a CAQT es la demostración de estabilidad para el caso en que la red es un anillo en una sola dirección. Además, apoyándose en el resultado ya probado de estabilidad para grafos dirigidos sin ciclos, se caracteriza todo el conjunto de redes estables.

El último modelo presentado, llamado *NSCAQT*, asume que cada cola tiene un reloj, hecho muy natural si se piensa que cada nodo es un computador y cada computador tiene su propio reloj, y además que dichos relojes no están sincronizados necesariamente. De esta forma, políticas que usan dicho reloj para tomar su decisión pueden cambiar su funcionamiento con respecto al probado en el modelo de CAQT. En este marco se demuestra que cuando los relojes pueden marcar horas distintas pero la diferencia entre ellos no varía, todas las políticas que son estables cuando todos los relojes marcan la misma hora siguen siendo estables. Además, para el caso en que las diferencias entre los relojes puede variar pero de manera acotada, se presentan dos familias de políticas estables que basan su decisión en el momento de inyección de cada paquete y algún parámetro del camino que tienen que recorrer. Finalmente para cuando los relojes no tienen cota para la diferencia entre ellos se presenta una nueva política estable.

Contents

1	Introduction	1
1.1	Adversarial Models	2
1.2	Definitions and Notation	4
1.3	Related Work	7
1.4	Main Contributions	12
2	Boolean Queuing Systems	17
2.1	Introduction	17
2.2	Model	18
2.3	BQS with Feed-forward Routing	21
2.4	Stability Sufficient Condition	22
2.4.1	Stability Result	23
3	AQT Model With Setups	27
3.1	Introduction	27
3.2	Equivalences between Models	30
3.2.1	Equivalence With Token Networks	32
3.3	Stability Results	35
3.3.1	Stability of Directed Acyclic Graphs	36
3.3.2	Stability of the Ring	38
3.4	Characterization of the stable networks	41
3.5	Instability Results	43
3.6	Fluid Model	48

4	Continuous AQT Model	53
4.1	Introduction	53
4.2	Universal Stability of Rings	56
4.3	Characterization of universal stability	61
5	Non Synchronized Model	65
5.1	Introduction	65
5.2	System model	67
5.3	Stability of policies for constant clock skews	70
5.4	Stability of policies for bounded clock skews	73
5.4.1	Universal stability of SISP	73
5.4.2	Universal stability of LISP	75
5.5	Universal stability of LIQ	78
6	Conclusions and Open Problems	81

List of Figures

3.1	U_1	43
3.2	U_2	43
3.3	Two concatenated gadgets.	44
3.4	Unstable network.	46
4.1	Elements involved in the nodes and links of the network in the CAQT model.	55
5.1	Basic process to transform G into G'	71

Chapter 1

Introduction

Sometimes we need a resource that is unavailable, and hence we must spend time in a queue waiting for it. A natural question is “How long are we going to wait?.” To answer this question it is best to first consider the maximum queue length possible (and, if such a maximum even exists). Queuing Theory studies the waiting times, queue lengths, and other properties associated with queueing systems. However, it relies on stochastic assumptions to model request generation and processing times.

Network of queues are used to model complex systems. For instance, many processes in industry production require completing a collection of tasks while satisfying temporal and resource constraints. Temporal constraints may imply that some tasks have to be finished before others can be started; resource constraints perhaps means that two tasks requiring the same resource cannot be done simultaneously. Typically, one can define “resources” as being machines, shops, staff, et cetera. In most cases, it is natural to consider dynamic situations where certain jobs arise over time. This gives rise to an on-line version of the job-shop scheduling problem [49].

A different framework in which networks of queues naturally appear are packet networks. Data packets must traverse a network from one point to another. When there are more packets traveling in a network than the network capacity service packets have to be enqueued. In these situations, queues play an important role in the quality of service.

In the context of network traffic and on-line job-shop scheduling (the areas of

main concern in this thesis) stochastic assumptions are often made about the process of requests generation. However, in complex networks like the Internet, typical assumptions such as, for example, exponentially and independently distributed inter-arrival times are unrealistic. In order to remove these hypotheses, an alternative elegant framework, referred to as *Adversarial Queuing Theory* (AQT), was proposed by Borodin, et al. [18]. In this model, requests are generated by an adversary rather than by a stochastic process. The AQT model was the starting point of a sequence of research that try to understand, in a worst case setting, the behavior of queueing networks with no trivial overload condition.

1.1 Adversarial Models

In the AQT model, a directed graph represents the network. The requirements are modeled by packets. A malicious adversary injects packets into the network; it decides the moment of injection, source and destination of each packet, and the path that each packet must traverse. In this model, when a packet arrives at its destination the packet is considered absorbed, and it disappears from the network.

A continuous version of the AQT model was presented in [17] by Blesa, et al., and called *Continuous AQT* (CAQT). The AQT and CAQT models are the main adversarial frameworks considered in this thesis. In the CAQT model, time runs continuously. In the AQT model, events happen at discrete moments (time steps). In both models the adversary can inject packets at any given moment, and each packet then travels through the network from source to destination.

In the AQT model, packets have the same size and, therefore, can be considered units, unlike in the CAQT model where packets are sequences of bits (basic components) and can be different sizes (in terms of bits). Henceforth, we assume that in the AQT model packets have size l in terms of bits.

A network is a directed graph. Each link of the network is associated with a positive finite bandwidth, which determines the transmission speed of the link (the capacity). Additionally, each link is associated with a finite propagation delay. The AQT model is the special case in which every edge has bandwidth equal to l (same

size as a packet) and propagation delay equal to zero.

When several packets want to cross an edge at the same time, a criterion called *policy* chooses which of them to send across the edge. The remaining packets must wait in a queue associated with the edge. It is assumed that each queue has infinite storage capacity, or infinite size.

In order to avoid trivial overloads of the network the adversary is restricted. Informally, the adversary is not allowed to inject (on average) more requests than r times the capacity of any edge (the parameter $0 \leq r \leq 1$ is referred to as the adversary's rate). Thus, there are no trivially identifiable "hot-spots" in the network. An adversary that satisfies this restriction is called a bounded adversary.

Both models, AQT and CAQT, can be seen as a game between a policy and an adversary played within a network. A completely specified game will be called a *system*. The evaluation criterion of a system is *stability*, i.e., whether the number of unserved requests remain bounded as the game runs for an arbitrarily long period of time.

Stability can be addressed from the network point of view or the policies point of view. A network is *universally stable* if the system is stable whenever any bounded adversary with rate $r < 1$ plays against any policy within the network. On the other hand, a policy is *universally stable* if whenever a bounded adversary with a rate smaller than one plays against the policy in an arbitrary network and the system is stable.

We assume that all packets' paths are edge-simple, i.e. they do not need to traverse the same edge more than once. However, they can visit the same node several times. This assumption does not decrease the generality of the model in terms of universal stability of policies, since it is known that a system in which packets may traverse the same edge several times can be simulated by another system with only edge-simple paths [10].

Formal definitions and general notation are presented in Section 1.2, (or in each chapter if it is required).

1.2 Definitions and Notation

Although each chapter of this thesis will present a slightly different model, here we introduce common notation and definitions. As a result of the differences among the models used in each chapter, some notation or definitions will be introduced specifically when required. Chapter 2 is a special case since it is self-contained. Chapter 3 has a common framework with Chapters 4 and 5, but still needs some specific definitions. Finally, Chapters 4 and 5 are closely related, hence they rely on the same conventions.

The network. A network is represented by a directed graph G , formed by a set of nodes $V(G)$, representing the hosts and routers, and a set of edges $E(G)$, representing links between the nodes. Each link e of the network is associated with a positive finite bandwidth B_e , which determines the transmission speed of the link, and a finite propagation delay $P_e \geq 0$. We use a specific notation for the largest propagation delay and smallest bandwidth as follows: $P_{\max} = \max_{e \in E(G)} \{P_e\}$ and $B_{\min} = \min_{e \in E(G)} \{B_e\}$. (Since G is finite, these values are well defined.) In the particular case of Chapter 3 $B_e = l$ and $P_e = 0$ for all links e .

Packets are sequences of bits, potentially different in size based on the number of bits. We denote by L_p the size of a packet p (in bits) and by L_{\max} the maximum size of a packet. Again in Chapter 3 we consider only the case where $L_p = l$, for all packets p .

The bounded adversary. The adversary, denoted by A , defines the network's traffic pattern, continuously deciding which packets are injected. For each packet p , the adversary chooses the moment of injection $T_0(p)$, the source node $v_0(p)$, the destination node $v_{d_p}(p)$, and the path the packet has to traverse $e_0(p), e_1(p), \dots, e_{d_p-1}(p)$. (When it is clear from the context, we can omit p from the notation.) Notice that d_p represents the length of the path packet p has to traverse. We denote by d_{\max} the length of the longest path of a packet, which is clearly bound by the length of the longest edge-simple path in the network.

Although the adversary controls the traffic arrival, it is restricted on the load that it can inject into the system. We assume that the injection of a packet is instantaneous. If $N_e(I)$ represents the number of bits of the packets injected by A which want to cross edge e during an interval I , then we require that for every edge e and for every time interval I

$$N_e(I) \leq r|I|B_e + b = (1 - \varepsilon)|I|B_e + b. \quad (1.1)$$

We denote by r , $0 < r \leq 1$, the long term rate (load) the adversary can impose on the system. For convenience we sometimes use the notation $r = 1 - \varepsilon$, for $\varepsilon \geq 0$. The parameter b , $b \geq 1$, is the overload allowed in the adversary injections, which is the excess of bits allowed to enter into the network at any time during the complete game. An adversary that satisfies this condition is called an (r, b) -adversary.

Packets, queues, and buffers. Because of the above restriction (1.1) on the adversary and the assumption of instantaneous injection of packets, it can be easily observed that $L_{\max} \leq b$. Note that b is also an upper bound on the number of packets that the adversary can inject at the same time with e as a common edge in their injection path.

Note that once a packet p starts to traverse edge e , it will spend $\frac{L_p}{B_e} + P_e$ units of time to completely cross it. Hence, the largest amount of time that a packet can spend crossing an edge denoted D_{\max} is $D_{\max} = \max_{e \in E(\mathcal{G})} \left\{ \frac{L_{\max}}{B_e} + P_e \right\} \leq \frac{b}{B_{\min}} + P_{\max}$.

Packets follow their path sequentially traversing one edge after the other towards their destination. As later detailed in Chapter 4, at every network node there is a reception buffer for each edge entering the node, and an output queue for each edge leaving the node. The output queue of an edge has an unbounded capacity and holds the packets that are ready to cross this edge. The scheduling policy of the edge's output queue chooses the next packet that will be sent across the edge among those packets in the output queue. The reception buffer is used to store the received portion of a packet until it has arrived completely. Then, a dispatcher instantaneously places the packet in the corresponding output queue or the packet disappears from the system if it has already reached its destination.

Policies A *scheduling policy* specifies, for each output queue in link e and each moment (time step in case of discrete time), which packet waiting at the queue is to be moved across the edge. In Chapter 3, due to the specific nature of the problem studied, a policy has more components; for the sake of convenience, when we arrive at Chapter 3 we will redefine the policy concept.

We say that policies are distributed if they do not depend on the state of other edges to make scheduling decisions. Policies are *work-conserving* (also called *greedy*) if they always send a packet across the link as long as the edge's output queue is not empty. In this thesis we will only consider work-conserving and distributed scheduling policies. Furthermore for the purposes of this thesis, we will assume that all the queues use the same scheduling policy. Examples of policies are:

- First in First Out (FIFO) — Gives the highest priority to the packet that has been in the queue for the longest time.
- Last in First Out (LIFO) — Gives the highest priority to the packet that has been in the queue for the shortest time.
- Nearest To Go (NTG) — Gives the highest priority to the packet that still needs to traverse the shortest path (in number of edges).
- Farthest To Go (FTG) — Gives the highest priority to the packet that still needs to traverse the longest path (in number of edges).
- Shortest In System (SIS) — Gives the highest priority to the packet that has been in the system for the shortest time.
- Longest In System (LIS) — Gives the highest priority to the packet that has been in the system for the longest time.
- Farthest from Source (FFS) — Gives the highest priority to the packet that is farthest from its source.
- Nearest from Source (NFS) — Gives the highest priority to the packet that is nearest from its source.

The *system* (G, P, A) is the complete description of a scheduling policy P and an (r, b) -adversary A injecting packets in network G . A system (G, P, A) is said to be *stable* if, for every initial configuration C_0 , there is a constant M (which may depend on the size of the network, the size of the initial configuration and parameters of the adversary) such that, the number of packets in any queue is bounded above by M .

If (G, P, A) is stable for every network G , and every (r, b) -adversary with $r < 1$, then the policy P is said to be *universally stable*. On the other hand, if (G, P, A) is stable for every greedy policy P , and every (r, b) -adversary with $r < 1$, then network G is said to be *universally stable*.

Some additional notation is needed to describe the state of the queues and the packets at a specific time step. We will use $Q_t(e)$ to denote the queue size of edge $e \in E(G)$ at time t (when convenient it could be given in number of bits or packets), and define $Q_{\max}(e) = \max_t Q_t(e)$. Similarly, we will use $R_t(e)$ to denote the number of bits that at time t are crossing link e , or have already crossed it but are still in the reception buffer at the target node of e . Let $R_{\max}(e) = \max_t R_t(e)$, and observe that $R_{\max}(e) < P_e B_e + L_{\max}$. Finally let $A_t(e)$ denote the number of bits in the system that need to traverse e and still have to be transmitted across link e at time t . The bits in $Q_t(e)$ are accounted for $A_t(e)$, but those in $R_t(e)$ are not.

1.3 Related Work

One of the earliest adversarial models was presented by Cruz in [26] [27]. In this model, each packet arrives into the network in a session, and each session has a predefined path. A session is permanent. All sessions are determined by an adversary who controls injections, it determines a session's paths, rate, and overload. The injection rate represents the mean load of a session, while the allowed overload represents the instantaneous load over the mean load of a session. This traffic model is known as *token bucket* or *leaky bucket* model. The adversary is restricted such that the total rate of all sessions using a given edge is strictly less than 1. This model received such a significant amount of attention that an entire theory, known as *Network Calculus*, was developed around it [20].

In the leaky bucket model, Cruz [27] proves the stability of every greedy policy on every layered directed acyclic graph. Furthermore, Tassiulas and Georgiadis [55] show that every greedy policy on the ring is stable. In Cruz's model, since session paths are predetermined, quality of service is obtained by a careful scheduling in the routers. It is shown that if no edge works at full injection rate (injection rate equal to its capacity), it is possible to schedule all packets in each session in such a way that packet delays are asymptotically optimal, and that these delays can be achieved with high probability by a distributed random scheduling algorithm [13] [14].

As mentioned before, Borodin et al. [18] proposed the AQT model in order to study networks under a worst case scenario. They showed that directed acyclic graphs are universally stable in the AQT model, even at injection rate equal to 1. Furthermore, they proved that stability at full injection rate ($r = 1$) does not carry over to networks with directed cycles. They demonstrated that an adversary with injection rate equal to 1 together with policy LIS makes a system unstable when the network is a unidirectional ring. Finally, they showed that FTG is stable in a unidirectional ring against every adversary, even when subject to an injection rate equal to 1.

Comparing Borodin's AQT model with Cruz's leaky bucket model, one can say that any set of k sessions in Cruz's model corresponds to an adversary of rate strictly less than 1 in Borodin et al.'s model. Hence, a stability result in the latter model implies an analogous result in the former. However, the converse direction does not hold.

In the AQT model, Andrews et al. [11] proved that scheduling policies play an important role on stability. Depending on which scheduling policy is used, the behavior of a system may change drastically. For instance, they proved universal stability for FTG, NFS, SIS, and LIS, and they built unstable systems for FIFO, LIFO, NTG and FFS. The results about instability of FIFO challenged the general belief that FIFO is a good scheduling policy. Díaz et al. also studied the performance of the FIFO policy in the AQT model [32]. Furthermore, Andrews [9] showed that FIFO is unstable even in Cruz's model. Other results showing the instability of FIFO at

arbitrarily low injection rates in the AQT model are due to Bhattacharjee et al. [16] and Koukopoulos et al. [45].

Policies with good performance guarantee were also studied by Andrews et al. [11] who proposed a random scheduling policy which ensures with high probability low end to end packet delays.

As far as specific scheduling policies are concerned, Àlvarez et al. [4] proved that the set of stable networks using FFS as the scheduling policy is equal to the set of universally stable networks. In the case of FIFO, Weinard [57] characterized stable networks under FIFO, and gave a polynomial algorithm to decide whether such property holds. Koukopoulos et al. [44] [46] studied how dynamic changes of link capacities affect the instability of greedy policies.

An important question is to decide whether a network is universally stable. This issue was first addressed by Goel [38] followed by Àlvarez et al. [6]. Finally, Àlvarez et al. [7] obtained characterization of universal stable networks under several adversarial models. Using those characterizations they provided polynomial time algorithms for testing stability under the NTG-LIS policy (with LIS used as the tiebreaker).

Specifically in directed acyclic graphs, Adler et al. [1] showed that the policy LIS require buffers of only linear size (the length of the longest path in the network). Furthermore, they showed that all of the packets incur only linear delay. These upper bounds were complemented by linear lower bounds on buffer sizes and packet delays.

Andrews et al. [14] gave a lower bound for LIS on the maximum delay of packets which is exponential on the length of the longest path followed by any packet. This lower bound is obtained on a tree of depth exponential in d . Thus, Adler's upper bounds also show that there is an exponential gap between the two measures (longest directed path in the network, and longest path followed by any packet). Finally, Rosén and Tsirkin [51] study the behavior of the AQT model when the adversaries rate is 1.

Among stable policies there are differences in packet delays and queue size. Another research direction was taken by Echagüe et al. [33]. They looked for an injection rate which ensures stability for all work conserving policies. They showed

that any greedy policy is stable against every bounded adversary with injection rate smaller than $1/d$, where d is the largest number of links crossed by any packet. For the specific case of FIFO, Lotker et al. [50] proved that there is instability at any injection rate smaller than $1/2$, and that FIFO is always stable if the injection rate is less than $1/d$, where d is the length of the longest route used by any packet.

As far as small queue size and packet delays are concerned, Weinard [56] showed that all policies without time knowledge¹ have exponential queue sizes. This result suggests that time-keeping is necessary to obtain sub-exponential performance bounds.

Routing under an adversarial model was studied by Aiello et al. [2] who considered the problem of adversarial packet injection where only destinations are specified. In this setting the policy decides the routing path. Routing when the network topology changes dynamically was considered by Awerbuch et al. [15]. Aiello et al. considered the case in which buffers have fixed sizes and packets may be dropped [3]. Andrews et al. studied source routing that guarantee the properties of an (r, b) -adversary, and provide the first polynomial delay distributed deterministic policy [12].

Seminal works for stochastic queueing networks are [52] and [48]. These works establish that load conditions are not sufficient for stability. In the specific case of FIFO, Bramson in [21] [22] [23] presents interesting results. For instance, he shows that FIFO is unstable for Poisson arrivals and exponential service times, and that FIFO is unstable under arbitrary small ratio of arrival to service rates. These results motivated the search for new tools for deciding whether specific networks were stable. So-called fluid models were developed for such purpose by Dai [28] [30].

Fluid models for stochastic multiclass queueing networks were introduced by Chen [25]. Then, Dai and Jennings [31] generalized the result of [28] to multiclass queueing networks. Finally, several papers studied the way to ensure feed-forward routing in multiclass queueing networks by prohibitions in the injected paths [54] [35] [34]. Classical texts in queueing theory and multiclass queueing networks are [41] [43] [39].

¹Time knowledge means that the policy has time-keeping information about the packets, e. g. a packets age or its current waiting time.

Analogues of fluid models for stochastic queueing networks, but for the AQT model were obtained by Gamarnik [36]. He showed how stability of a queueing network can be determined by considering an associated fluid model where the stability of the fluid model implies the stability of an underlying adversarial queueing network. This allows the possibility of analyzing stability of adversarial networks using established stability methods from continuous time processes, for example, the method of Lyapunov functions or trajectory decomposition.

The AQT model has been also generalized in order to study dynamic networks. In dynamic systems it is necessary to use scheduling and routing to ensure that packets arrive to their destination; for example, one way in which they can be dynamic is because of the continuous changes on the network topology. A first approach in this direction in which link capacity may depend on time was presented in [47] and [19]. In addition, in [8] a model in which links may fail was presented.

In order to make the AQT model more realistic, Blesa et al. [17] introduced a continuous version of it, called CAQT. In this model, time becomes continuous, packets may have different sizes (in bits), and links may have different speed and propagation delay. Blesa et al. [17] characterized the set of universal stable networks and showed that there are universal stable policies (LIS, SIS, FTG and NFS among others). Finally, they proved that in some networks there are policies based on packet length, bandwidth parameters and propagation delay which are unstable against a bounded adversary.

In [5], Àlvarez et al. proposed several variations on the AQT model, and derived stability results concerning networks and policies in such settings. The first variation they considered is to allow the adversary to set the priority of packets at injection time. Similarly, they proposed a variable priority model, in which priorities may change at each time step. Again these priority changes are under the control of the adversary. In terms of links failures they proposed two models in which the adversary may modify dynamically the network topology. Àlvarez et al. [5] showed that the set of universally stable networks in the adversarial model remains the same for the proposed models. Concerning the policies, they showed that there are no uni-

versally stable policies when priorities can change dynamically. However, most of the universally stable policies remain universally stable in most of the variations.

1.4 Main Contributions

We now discuss the main contributions of this thesis in the order in which they appear.

The main contribution of Chapter 2 is the new way of characterizing customers of a queueing system. In this chapter we consider queueing systems that are more general than those considered in AQT and CAQT. A customer is presented as a set of processes or “chunks” which helps to understand and represent formally each of its components, states, or processes depending on what a customer represents. Dependency between chunks is captured by means of Boolean functions which generalize the idea of network paths. Sufficient conditions for stability are presented even when servers are fully loaded. We then conclude that queueing networks with feed-forward routing are stable under adversaries with full rate injection.

In Chapter 3, we introduce an adversarial model that captures the natural scenarios where *distinct types* of network resource requests are generated in a non-predictable (adversarial) way but without trivially overloading the network. We call the model *AQT with setups*

We consider the case where an adversary injects packets into a network and chooses a class (or type) for each packet. The possible set of request types is denoted \mathcal{I} . The cases where \mathcal{I} contains one and more than one element will be referred to as the *single-class* (or *single-type*) and *multi-class* (or *multi-type*) case respectively. In the multi-type scenario, we will further distinguish two situations, depending on whether or not there is traversal function which is a constant function associated with each packet. The non-constant case will be called the *dynamic multi-type* case and the underlying adversary denoted as a dynamic multi-type adversary. The non-dynamic multi-type setting captures situations where the type of a packet is fixed once and for all at its injection step. Even such a restricted scenario is of major practical relevance, e.g., in TCP/IP networks like the Internet, packets may be classified according to

their data load (e.g., audio, video, et cetera.) or destination port which remain fixed throughout the packet’s life-span.

The first result of Chapter 3 shows that although constrained, the non-dynamic multi-type scenario is of key importance. Indeed, stability issues for the dynamic multi-type case may be addressed by focusing on the non-dynamic model. Specifically, in Section 3.2 we show that in the adversarial with setup model, dynamic multi-type routing networks can be simulated by similar non-dynamic networks.

It is interesting to point out that dynamic multi-type routing networks with setups are “equivalent” to the so-called token routing networks introduced by Kiwi and Russell [42]. This assertion is precisely stated and proved in Section 3.2.

The previous discussion justifies why we focus our study of the AQT with setups model to the non-dynamic multi-type case. In this latter setting, it is rather intuitive that one should prefer switching policies that avoid frequently incurring setup costs. An extensively investigated scheduling policy is exhaustive service which involves processing all packets of a given class until packets of that class are no longer available. We shall see that this alone does not insure stability. The choice of which class to service next is also important, an issue already recognized by Dai and Jennings [29] who proposed the notion of sensible policy — given a switching policy S and a parameter θ let S_θ denote the policy that picks according to S which class to service next among all the ones that have at least θ packets waiting in the queue. If no class has at least θ packets waiting at the queue, then the policy may choose whatever class it prefers. Switching policies of the form S_θ are called sensible policies.

In Section 3.3, we adapt arguments of [18] [11] and establish stability of networks whose policies rely on sensible policies, e.g., in the multi-type setting we show an analogue of Borodin, et. al’s [18, Theorem 1] result regarding universal stability of acyclic digraphs under work conserving policies.

One is tempted to conclude that if a network G under a packet scheduling policy P is stable against an adversary A , then for any Δ (setup cost or time without work) there is a packet switching policy S for which $(G, \Delta, (P, S), A)$ (system in the AQT with setup model) is also stable (The converse is easily verified.). In Section 3.5, we

show that this is not the case for various natural switching policies. In fact we prove that for every pair composed by a scheduling policy P and a sensible switching policy S_θ , and for all $r < 1$, there exists a network G and an (r, b) -adversary A , such that the system $(G, \Delta, (P, S_\theta), A)$ is unstable for all $\Delta \geq 0$.

The state of affairs described above motivates us to explore general techniques that could be useful for proving stability for given networks. Specifically, in Section 3.6 we show analogues of Gamarnik's [36] results (in the multi-type request with setups scenario). Through these results one can establish the stability of a given network by considering an associated fluid model.

In Chapter 4, we consider the continuous version of the AQT model (CAQT) presented in [17] by Blesa et al. Under this framework we show in Section 4.2, that unidirectional rings are universally stable against every adversary with rate smaller than 1. This result and the fact that networks with a directed acyclic graph topology are always stable, even if the links are fully loaded [17], allows us to identify which networks are universally stable for the CAQT model.

The main contribution of Chapter 5 is the detailed definition of a model in which clocks need not be synchronized. We call this model *Non Synchronized CAQT* (NSCAQT). This is a novel feature in the context of previously proposed adversarial models. First, we consider the NSCAQT model when clocks run at the same speed. Hence, there are constant differences among clock times. We study under this model scheduling policies that assign priorities to packets based on their injection times and their remaining edges in their paths. In Section 5.3 we show that for these policies the NSCAQT system can be transformed into a CAQT system by changing the topology of the network and the adversary. Hence, we conclude that any such policy that is universally stable under the CAQT model is also universally stable under the NSCAQT model with constant skews.

In Section 5.4 we explore universal stability under the NSCAQT model with variable differences (that are bounded) among clock times. We define two families of policies that include LIS and SIS, and show that they are universally stable in such a model. The two families of policies are referred to as *Longest in System considering*

Path (LISP) and *Shortest in System considering Path* (SISP). Policies in the LISP and SISP families assign packet priorities depending on the injection time and the number of edges already traversed by the packet.

Unfortunately, for the universally stable policies identified in Chapter 5, all the upper bounds on delays and queue sizes that we could prove depend on the differences among clock times. In fact, for several of these policies it can be easily shown that delays and queue size must grow when differences grow. Thus, the natural question is whether there are policies whose performance does not depend on the differences among clock times. We introduce a new policy in Section 5.5 called *Longest in Queues* (LIQ) which gives priority to the packet that has been waiting in queues the longest amount of time. We show that this policy is universally stable in the NSCAQT model with bounded skews. More interestingly, we show that in the NSCAQT model with constant skews this policy has an upper bound on the end-to-end delay that does not depend on the skews and is close to that of LIS in the CAQT model.

Some results of this thesis could be presented under a common and unified framework, since most of them use similar arguments, but this would make the presentation cryptic and difficult to parse. Hence, we have decided to adapt the proof arguments to each of the specific models. We feel this makes the presentation more accessible to the reader. We also hope that it makes it easier to understand how changes in the underlying model impact the phenomena under study.

Chapter 2

Boolean Queueing Systems

2.1 Introduction

Some industrial processes, or computational tasks, can be divided into different threads or subprocesses. Sometimes, these subprocesses have dependencies among them. For instance, if the industrial process is the construction of a car, then different components (wheels, different parts of the chassis, motor, etc) may be build in parallel. But, the car can not be painted before it is assembled. Hence, the processing at the assembling station must be previous to the paint station.

Hence, a task may be completely described by the set of all its subprocesses, and their characteristics, namely, the station in which the subprocess must be completed, subprocesses that must be completed previously, time that the subprocess needs to be completed, or time that it needs to start its processing.

In this chapter, we introduce an alternative model to AQT or CAQT to analyze queueing systems. The main contribution is a novel approach modeling customers (tasks). A customer is composed by a set of processes which formally represent each of the components, states, or associated subprocesses of the task. Furthermore, each of these subprocesses has associated a concatenation function which represents its dependencies of other subprocesses of the same customer.

We assume that customers (or, more precisely, processes in a customer) are processed in servers, and when more processes arrive to a server than the server's capacity, a queue is generated at the server. It is assumed that each server has an infinite

buffer to store its own queue.

We use an adversarial setting to study our model. In this setting an adversary injects customers in the system in a malicious way. Additionally, each customer may have different characteristics, which are also defined by the adversary.

A desired property is that the server's service rate matches its injection rate for an arbitrarily long period of time. This implies stability of the system, since the number of customer at any time is bounded.

We give a sufficient condition for the desired property, this condition is on the characteristics of the customers.

Structure. The structure of this chapter is the following. In Section 2.2 we introduce the notion of Boolean Queueing Systems (BQS). And, in Section 2.3 we present a sufficient condition for stability of a system.

2.2 Model

In this section we define *Boolean Queueing Systems* (BQS) in the most general way. The three main components of a BQS are:

- A set $S = \{s_1, s_2, \dots, s_n\}$ of n servers.
- A malicious *adversary* A who injects customers in servers.
- A scheduling *policy* P , which is the criteria used by servers to decide which customer to serve next among the customers waiting in their queues.

Customers. Each customer is a finite set of processes. Let k be a customer, then $\{p_1^k, p_2^k, p_3^k, \dots, p_{l_k}^k\}$ is the set of processes of k . Each p_i^k has three parameters to determine it (s_i^k, d_i^k, t_i^k) , s_i^k is the server in which p_i^k must be served (executed), d_i^k denotes its finite activation delay, and t_i^k is the time it takes server s_i^k completely serve process p_i^k . Furthermore, we associate to each p_i^k two functions:

- The first function $e_i^k(t)$ is a Boolean *state function* which indicates if p_i^k has been completely served at time t , i.e., at time instant t , if $e_i^k(t) = 1$ process p_i^k was completely served by s_i^k . Otherwise, p_i^k still needs to be served (it is possible that it is already being served, through).
- The second function $f_i^k : \{0, 1\}^{l_k} \rightarrow \{0, 1\}$ is a *feasibility function*, and it indicates whether p_i^k could be served, depending on the states of the other processes in the same customer.

If $f_i^k(e_1^k(t), e_2^k(t), \dots, e_{l_k}^k(t)) = 0$, then process p_i^k is *blocked*. Otherwise, if $f_i^k(e_1^k(t), e_2^k(t), \dots, e_{l_k}^k(t)) = 1$, then p_i^k is *feasible*. With this condition, a process cannot start being served until some given state of the processes in the same customer holds. Hence, the feasibility function can be used, for instance, to force the execution sequence of the processes of a customer.

The activation delay d_i^k of p_i^k represents a setup cost, expressed in time, that p_i^k must incur once it becomes feasible and before it starts to be processed. If t is the first instant at which p_i^k is feasible, then p_i^k will incur its activation delay during time interval $[t, t + d_i^k]$. Hence, it can not be served during such interval. Process p_i^k will be called, during such time interval, a *delayed* feasible process (or only delayed process).

When p_i^k finishes its activation delay, it can be served, and since this moment ($t + d_i^k$) will be referred to as an *active* feasible process. Equivalently, a feasible process is active if it has been feasible for at least d_i^k time.

A customer as the previously described will be referred to as a *Boolean* customer.

Remark 1 A *Boolean customer* does not follow (explicitly) a path as a customer does in classical queueing networks (like in the AQT model). However, the set of processes's feasibility functions (f) represents generalized paths. They chain processes in a customer.

If we connect the set of servers used by processes in a customer following the next condition:

- There is an arc from s_i^k to s_j^k if and only if there exists values for $e_r^k(t)$ when $r \in \{1, 2, \dots, l_k\}$ and $r \neq i$, such that

$$f_j^k(e_1^k(t), \dots, e_{i-1}^k(t), 0, e_{i+1}^k(t), \dots, e_{l_k}^k(t)) = 0$$

and

$$f_j^k(e_1^k(t), \dots, e_{i-1}^k(t), 1, e_{i+1}^k(t), \dots, e_{l_k}^k(t)) = 1$$

This construction gives us a directed graph associated to a customer referred to as the path that a customer follows.

Scheduling policy. In each server a scheduling policy P specifies which process of all customers to serve next. We assume that scheduling policies are greedy or work conserving, i.e., a server always decides to serve if there is at least one active feasible process in its queue.

Adversary. We assume that there is an adversary A who injects load (customers) into the servers. It is an adversary because A always tries to overload servers. In order to avoid trivial overloads the adversary is bounded in the following way. Let $N_s(I)$ be the total work injected by the adversary during time interval I in server s , i.e., $N_s(I) = \sum t_i^k$ over all k injected during I such that $s_i^k = s$. Then, for every server s and interval I the adversary is bounded by:

$$N_s(I) \leq r|I| + b, \quad (2.1)$$

where $0 < r \leq 1$ is called the *injection rate*, and $b \geq 1$ is called the *burstiness* allowed to the adversary. Observe that (2.1) implies $\max_{i,k} t_i^k \leq b$, since customers are injected instantaneously and $|I|$ can be very small.

An adversary that satisfies (2.1) is called a *bounded* (r, b) -adversary, or simply (r, b) -adversary. An (r, b) -adversary that injects Boolean customers is called *Boolean adversary*.

The system formed by a Boolean adversary A injecting customers in the set of servers S using the scheduling policy P is called *Boolean queueing system* and denoted by (S, P, A) .

Let $Q_s(t)$ be the number of customers with processes in server s at time t .

Definition 1 We say that a Boolean queueing system (S, P, A) is stable if there exist a value M such that $Q_s(t) \leq M$ for all t and for all $s \in S$, where M may depend on the system parameters (adversary, servers, and customers characteristics) but not on the time.

2.3 BQS with Feed-forward Routing

Multiclass queueing networks are networks of servers with different customer classes. Among these, we can identify the subclass of feed-forward networks. Roughly speaking, these are multiclass queueing networks in which customer multipaths do not create cycles of interdependence.

Multiclass queueing networks are well studied. In [24] Chang showed that multiclass queueing networks with feed-forward routing are stable when injection is not done at full rate, i.e., the rate injection is strictly smaller than 1. In [54] [34] [35] algorithms that break all the cycles in the network are provided in order to prevent any interdependence between flows, or equivalently transform any network in a network with feed-forward routing. In this section we define feed-forward routing for BQS.

Intuitively, a system will have feed-forward routing if its adversary injects customers with processes whose feasibility depends only on processes executed in a , previous server, in a total order predefined of the set of servers. Formally: let \mathcal{O} be a total order in S . A Boolean adversary A is called *feed-forward*, if all the customers injected by A have the following characteristics:

- $s_i^k \leq s_j^k$ in \mathcal{O} for all $i < j$ and for all k .
- $f_i^k : \{0, 1\}^{i-1} \rightarrow \{0, 1\}$ for all i , i.e., f_i^k only depends on the $i - 1$ first parameters.

Hence, when $f_i^k(e_1^k(t), e_2^k(t), \dots, e_{i-1}^k(t)) = 0$ process p_i^k is blocked, while if $f_i^k(e_1^k(t), e_2^k(t), \dots, e_{i-1}^k(t)) = 1$ process p_i^k is feasible. And now, the feasibility of p_i^k depends only on the status on processes with s_j^k smaller in \mathcal{O} than s_i^k . Or equivalently

the path of every customer is increasing in \mathcal{O} . Hence no set of customers's paths create a directed cycle.

Definition 2 *Let (S, P, A) be a Boolean queueing system. We say that (S, P, A) has feed-forward routing if A is a feed-forward adversary, and for all i and k , f_i^k is not decreasing and $f_i^k(1, 1, \dots, 1) = 1$.*

Notice that f_1^k is a constant, and that for every system, $f_1^k = 1$ for all k , since $f(1, 1, \dots, 1) = 1$.

By definition, feed-forward queueing networks [39, pp. 200-205] are networks of servers which can be ordered in such a way that costumers always visit them in increasing order.

Remark 2 *Notice that, a feed-forward queueing system is also a BQS with feed-forward routing.*

Our goal is to prove that every BQS with feed-forward routing is stable at full rate injection, i.e., $r = 1$.

2.4 Stability Sufficient Condition

Previously to present the stability result we introduce some notation and definitions. Let $T_{max} = \sup_{i,k} t_i^k$ be the maximum time that it takes a process to be completed. Due to (2.1), and since we assume instantaneous injection of customers, we have that $T_{max} \leq b$.

Let $T_{min} = \inf_{i,k} t_i^k$ be the minimum time that it takes a process to be completed. We assume that $t_i^k > 0$ for every i and k , otherwise we can ignore p_i^k . Hence $T_{min} > 0$.

We denote by $D_{min} = \min_{i,k} d_i^k$ and $D_{max} = \max_{i,k} d_i^k$, the minimum and maximum of d_i^k 's, respectively. Since, $d_i^k \geq 0$ it follows that $D_{min} \geq 0$. We henceforth assume that D_{max} is finite.

Finally, we use $L = \max_k l_k$ to denote the maximum length of a customer, which we henceforth assume is always finite.

2.4.1 Stability Result

In order to prove stability we use the potential method. As is usually the case for these type of proof arguments, we upper bound the potentials independently of time, hence obtaining the desired conclusions.

Definition 3 Let $D_s(t)$ be the amount of time that server s would need to completely serve (drain) all processes present in its queue at time t if they were all active and no new customer was injected:

$$D_s(t) = \sum_{K(s,t)} t_i^k$$

where $K(s, t)$ is the set of pairs (k, i) such that k was injected before time t , $s_i^k = s$ and $e_i^k(t) = 0$.

Now we define a potential function $\Phi(\cdot)$ such that for all $s \in S$ (i.e., $1 \leq s \leq n$)

$$\Phi(s) = D_s(0) + Q_s \cdot L \cdot T_{max} + b$$

Where $\Phi(0) = D_{max}$ and $Q_s = \frac{1}{T_{min}} \sum_{i=0}^{s-1} \Phi(s)$.

Lemma 1 The amount of time that a server needs to drain its queue at any time is bounded by a value that depends only on the server and the parameters of the system, i.e., for all t ,

$$D_s(t) \leq \Phi(s).$$

Proof: To prove this lemma we use induction on the server labels.

- Case $s = 1$:

Let us observe first that if $Q_1(t) > Q_1$, then there is at least one active process in server 1. This is because each customer in server 1 has at least one feasible process, and at most $\frac{D_{max}}{T_{min}}$ of those are delayed. Hence, at least one process is active.

Fix some time instant T , we prove that $D_s(T) \leq \Phi(s)$. First case: $Q_1(t) > Q_1$ for all $t \in [0, T]$. Then from the previous observation processes have been served continuously in the interval. Hence, from this and (2.1)

$$D_1(T) \leq D_1(0) + T + b - T = D_1(0) + b \leq \Phi(1).$$

Second case: There exists some moment $t \in [0, T]$ such that $Q_1(t) \leq Q_1$. Let t be the greatest such instant. Then again by previous observation and (2.1),

$$D_1(T) \leq D_1(t) + (T - t) + b - (T - t) \leq Q_1 LT_{max} + b \leq \Phi(1).$$

- Case $s = k > 1$:

Induction hypothesis: $D_s(t) \leq \Phi(s)$ for all $s < k$ and for all t .

By induction hypothesis, we have that the amount of customers in servers $s < k$ is at most

$$\sum_{i=1}^{k-1} \frac{\Phi(i)}{T_{min}},$$

which we denote by X .

Observe that if $Q_k(t) > Q_k$, then there is at least one active process in server k . Indeed, each process of a customer only depends on processes of the same customer. If all the processes of a customer in server k are blocked, it is because there is some unserved process of the same customer in some previous server (at most X). The other customers with processes in k , have at least one feasible process, among which at most Q_1 are delayed. Hence, if $Q_k(t) > Q_k = Q_1 + X$, then there is at least one active process in k .

Then we fix some time instant T , and prove that $D_s(T) \leq \Phi(s)$. The two cases are similar to those of case $s = 1$.

First case: $Q_k(t) > Q_k$ for all $t \in [0, T]$. Then

$$D_k(T) \leq D_k(0) + T + b - T = D_k(0) + b \leq \Phi(k).$$

Second case: There exists some moment $t \in [0, T]$ such that $Q_k(t) \leq Q_k$. Let t be the greatest such time. Then,

$$D_k(T) \leq D_k(t) + (T - t) + b - (T - t) \leq Q_k LT_{max} + b \leq \Phi(k).$$

■

The following theorem follows from the above lemma.

Theorem 2 *Let (S, P, A) be a BQS with feed-forward routing, then (S, P, A) is stable.*

Finally, an immediate corollary of the previous theorem is:

Corollary 3 *A feed-forward routing network is stable against an adversary with full rate injection.*

Chapter 3

AQT Model With Setups

3.1 Introduction

In this chapter we generalize the AQT model to a model in which the adversary may inject different types of packets, and each arc must spend time without working in order to change types to serve. We call this new model *AQT with setups*.

We assume that the adversary will specify, for each packet and for each edge along a packet's path, a type (or class) chosen from a fix set of classes \mathcal{I} . Formally, when a packet is injected, the adversary specifies a traversal path $P = e_0 \dots e_\ell$ and a *traversal type function* $f : P \rightarrow \mathcal{I}^\ell$. The traversal function should be interpreted as capturing the fact that as a packet progresses through the network, it might require being serviced by different resources, specifically, resource $f(e_i)$ for edge e_i . This formalization captures not only the case where a fixed type is associated to a packet at injection. Indeed, it encompasses situations where types might change as the packet moves through the network.

In a routing network, if several packets try to cross an edge during the same step, then a packet scheduling policy decides which one to send across the link. The other packets wait in the edge's queue. Our second departure from the standard AQT model is that no queued packet may be chosen by the scheduling policy if it belongs to a class different from the class of any packet sent across the link during the previous Δ time steps — Δ being a network parameter henceforth called *setup cost* or *latency*.

In the AQT model, the adversary is constrained by a pair (r, b) . The restriction

on the adversary is [18, 11], for every edge e :

$$N_e(t_1, t_2) \leq r(t_2 - t_1) + b \quad , \quad (3.1)$$

where $N_e(t_1, t_2)$ denotes the number of packets injected by the adversary during time interval $[t_1, t_2]$ ¹ that need to traverse edge e in their source to destination routes.

The existence of setup costs is captured by the following condition: For every edge e ,

$$D_{e,i}(t, t+1) > 0 \implies \forall j \neq i, D_{e,j}(t+1, t+1+\Delta) = 0 \quad , \quad (3.2)$$

where $D_{e,i}(t_1, t_2)$ denotes the number of type- i packets crossing e during interval $[t_1, t_2]$. We refer to conditions (3.1) and (3.2) as *load constraints* and *setup constraints*, respectively.

A *scheduling policy* specifies, for each network edge e and each time step, which packet waiting at the queue is to be moved (if possible). Typical scheduling policies are First In First Out (FIFO), Nearest To Go (NTG), Shortest In System (SIS), etc. A *switching policy* specifies, for each network edge e and each time step, whether to keep servicing packets of the type serviced during the preceding time step or to start servicing (if possible) a new type of packet. In other words, a scheduling policy specifies how priorities are assigned amongst packets of the same type, while a switching policy specifies priorities among distinct packet types. Examples of switching policies are:

- Largest Queue (LQ) — switch to serving the type that has the largest number of packets waiting e 's queue (ties are broken arbitrarily).
- Round Robin (RR) — types are numbered $0, \dots, |\mathcal{I}|-1$, if type i was last served, then switch to sending type $(i+1) \bmod |\mathcal{I}|$.
- First In First Out (FIFO) — switch to serving the type for which a packet first arrived at the queue (ties are broken arbitrarily).

¹Recal that in the AQT model, time is measured in discrete steps

- **PRIORITY** — types are numbered $0, \dots, |\mathcal{I}|-1$, switch to sending the type whose value is smallest among all the packet types awaiting at the queue (switch to type 0 if the queue is empty).

For the sake of preciseness, we point out that once the switching policy determines to start servicing a different packet type it can not change its decision before Δ steps have elapsed. So for example, an edge that has been idle for Δ units of time will not be readily available for servicing an arbitrary request type.

We say that a switching policy is *exhaustive*, if once it starts servicing packets of type i , it must continue serving packets of type i until none remain in the queue. In this work we only consider exhaustive switching policies. (For results concerning exhaustive policies in the standard stochastic queueing theory setting the reader is referred to [48, 37, 40]).

We say that a switching policy is *type-oblivious* if the choice of which class to serve next is independent of the type labels, i.e. for any permutation π of \mathcal{I} if the switching policy chooses to serve next type j on a given queue state, then when every packet of type i is replaced by one of type $\pi(i)$, the switching protocol will choose to serve next type $\pi(j)$. Examples of type-oblivious switching policies are LQ and FIFO defined above. In contrast, the switching policy RR is not type-oblivious. A weaker concept that encompasses all type-oblivious policies as well as policies such as RR is defined next.

We say that a switching policy is *shift-oblivious* if there exists an order of $\mathcal{I}^2 = \{0, \dots, |\mathcal{I}| - 1\}$ such that, if at a given time after serving packets of type i , the switching policy chooses to serve next packets type of j , then if every packet of type k is replaced by one of type $k + 1$ for all $k \in \mathcal{I}$ (where arithmetic on types is modulo $|\mathcal{I}|$), we will have that when a queue is serving packets type $i + 1$ the switching policy will choose to serve next packets type of $j + 1$.

Note that any type-oblivious switching policy is also shift-oblivious. Moreover, RR is an example of a switching policy that is shift-oblivious but not type-oblivious. Switching policy PRIORITY is an example of a switching policy that is not shift obli-

²Here we implicitly assume that \mathcal{I} as a set of numbers

ious.

A *policy* is one that specifies both a packet and a switching policy. The policy composed by scheduling policy P and switching policy S is denoted by (P, S) . A *greedy* or *work conserving* policy is one where, if there is at least one packet waiting to use an edge, the policy advances a packet through the edge provided it is possible.

The *system* $(G, \Delta, (P, S), A)$ is the complete description of: an (r, b) -adversary A injecting packets in the network G with Δ as setup cost, and in which policy (P, S) is used. A system $(G, \Delta, (P, S), A)$ is said to be *stable* if, for every initial configuration C_0 , there is a constant M (which may depend on the size of the network, the size of the initial configuration, and parameters of the adversary) such that the number of packets in any queue is bounded above by M .

If $(G, \Delta, (P, S), A)$ is stable for every network G , $\Delta \geq 0$, and (r, b) -adversary, $r < 1$, then policy (P, S) is said to be *universally stable*. On the other hand, if $(G, \Delta, (P, S), A)$ is stable for every policy (P, S) , $\Delta \geq 0$, and (r, b) -adversary, $r < 1$, then network G is said to be *universally stable*.

3.2 Equivalences between Models

The cases where \mathcal{I} contains one and more than one element will be referred to as the *single-class* (or *single-type*) and *multi-class* (or *multi-type*) case respectively. In the multi-type scenario, we will further distinguish two situations depending on whether or not associated to each packet there is traversal function which is a constant function. The non-constant case will be called the *dynamic multi-type* case and the underlying adversary as a dynamic multi-type adversary. The non-dynamic case is called *static*. The static setting captures situations where the type of a packet is fixed once and for all at its injection step.

In this section we first show that, in the adversarial setting, dynamic multi-type routing networks can be simulated by static multi-type networks, and vice-versa. We also describe the notion of token networks introduced in [42] and show that under some conditions it is “equivalent” to that of static multi-type routing networks.

Theorem 4 *Let $(G, \Delta, (P, S), A)$ be an arbitrary system where (P, S) is a greedy policy such that S is shift-oblivious and A is an (r, b) dynamic multi-type adversary. Then, there is a $(G', \Delta, (P, S), A')$ multi-type routing system where A' is a static adversary, and an injective mapping φ from configurations of G to configurations of G' , such that for any time step t , the configuration of G is C if and only if the configuration of G' is $\varphi(C)$.*

Proof: Without loss of generality assume the collection of types \mathcal{I} is $\{0, \dots, n-1\}$. The label of the types is chosen such that S act as shift-oblivious, i.e., when a queue finish to serve packets type i and S chooses to serve next packets type j , then if every packet of type k is replaced by a type $k+l$ packet for all $k \in \mathcal{I}$, we will have that when a queue finish to serve packets type $i+l$, S will choose to serve next type $j+l$ packets.

Let V and E be G 's node and edge sets, respectively. Network G' will also have V as node set. The edge set of G' will consist of $|\mathcal{I}|$ copies, say $e^{(0)}, \dots, e^{(n-1)}$, of each edge $e \in E$. The idea of the proof is to simulate the traversal in G of a packet p by n packets p_1, \dots, p_n going through G' . When p crosses edge e of G , each packet p_j will traverse a distinct edge $e^{(i)}$ of G' (the exact edge will depend on the type the adversary A assigns to packet p when traversing e).

The scheduling policy associated to G' is the same as the one of G . The switching policy is also the same in G and G' . The initial configuration of G' specifies which type will be served by each edge at time step $t = 0$. In our case, the packet type initially serviced by edge $e^{(j)}$ is $(l+j) \bmod n$ where l is the packet type initially serviced by edge e .

Now consider a packet p that is injected in G at time t by adversary A along path $P = e_1 \dots e_\ell$ with a traversal type function $f : P \rightarrow \mathcal{I}$. For $k = 1, \dots, \ell$, let i_k denote the type assigned to the packet when it wishes to traverse edge e_k , i.e., $i_k = f(e_k)$. Then, in G' adversary A' will inject n packets at time t , say p_0, \dots, p_{n-1} . The type of packet p_j is fixed at injection and equals i_j . Using mod n arithmetic on super-indices, the injection path of packet p_j is expressed as $P_j = e_1^{(j+i_1)} \dots e_\ell^{(j+i_\ell)}$. Observe that A' is an (r, b) static adversary in G' .

Note that, if we take a configuration in G' and we restrict it only to the edges with super-index 0 (relative to the construction of G'), then we have the same configuration in G . Similarly, if we restrict a configuration in G' to the edges with super-index l , then we have a isomorphic copy to the same configuration in G . It is isomorphic with respect to the mod n arithmetic, i.e., a type i packet in a configuration in G is a type $i + l \bmod n$ packet in a configuration in G' restricted to the edges with super index l . (We refer to a packet as a type i packet in a configuration in G when the packet is in a queue of an edge in G and needs to cross such edge under type i).

The desired conclusion follows from the observation that these isomorphic configurations move in a synchronous way, and then p crosses edge e_k of G at time step t if and only if p_j crosses $e_k^{(j-i_k)}$ of G' at time step t . ■

Corollary 5 *The shift-oblivious scheduling protocol (P, S) is universally stable in the AQT model with setups and dynamic adversaries if and only if (P, S) is universally stable in the AQT model with setups and static adversaries.*

3.2.1 Equivalence With Token Networks

We now compare the AQT with setups model with the token network model. In token networks [42], at each time step edges may possess a *token*. Only edges that hold a token in a given time step may be traversed by a packet. Thus, tokens represent scarce resources (e.g., a computational resource) for which packets compete. Edges that hold a given token are said to be *serviced* by the token.

In particular, we consider networks where tokens “move” around, i.e., a token that at time t is at a given edge e can be identified with a token that at time $t' > t$ is at some edge e' . The network is said to have *latency* Δ if $t' > t + \Delta$ whenever $e \neq e'$. In such networks, in order for a token to move from an edge e to another edge $e' \neq e$ it must be absent from the network for a time interval of length $\Delta \geq 0$.

Edges that hold a token every time step will be referred to as *permanent edges*. Non-permanent edges will be called *temporary edges*.

A *token switching policy* specifies, for each token and each time step, the subset of edges that hold tokens. For example, a FIFO token switching policy is one where priority is given to the edge (among those that can grab a token) that has not held a token for the largest amount of time. A *token scheduling protocol* is a (P, S) pair that specifies policies P and S for packet scheduling and token switching, respectively.

Conceptually, one might distinguish tokens according to the edges they might service. Call a token (and the associated token policy) *disjoint* if the collection of edges serviced by distinct tokens are disjoint.

For token networks, the adversarial model is exactly the one of AQT, but now, in order to prevent the adversary from trivially congesting the network, one imposes the following *stability constraint* on the adversary: For any time interval $[t_1, t_2)$, the number of packets injected into the network *that are destined to be serviced by a given token* is bounded by $r(t_2 - t_1) + b$. An adversary meeting such a constraint is also referred to as an (r, b) -adversary. Other standard notions, like for example stability, are easily adapted to the token network scenario.

We can now formally state our claim concerning the simulation of token networks by dynamic multi-type routing networks.

Theorem 6 *For every token network G with latency Δ , disjoint token policy (P, S) , and (r, b) -adversary A satisfying the network's stability constraint, there is a dynamic multi-type routing network $(G', \Delta, (P, S'), A')$, where A' is an (r, b) -adversary, and an injective mapping φ from configurations of G to configurations of G' , such that for any time step t , the packet configuration of G is C if and only if the configuration of G' is $\varphi(C)$. Moreover, if (P, S) is work conserving, so is (P, S') .*

Proof: The idea of the proof is to replace tokens by types in an edge. Starting with a network G with edge set E we build the network G' . The construction “collapses” all edges serviced by a given token into one edge (some care must be taken when a token services edges that belong to a path).

Consider the collection F of edges serviced by a given token. Remove F and add a directed edge (u, v) to the network. Identify all tail nodes of edges in F with u . Also, identify with v all head nodes of edges in F . Note that when F contains a

path of length at least 2, a loop is created at u . Call the resulting network G' . Now the adversary can specify the traversal type through (u, v) with a set of types where each type represents an edge in F , and it can specify the traversal type through an edge not in F with one type associated to the edge. (We abuse notation and denote the respective type by the respective edge in both cases).

A packet injected along path P in the original network will be injected along the path P' to which P is mapped G' 's construction process, i.e, if there is some edge in P that was collapsed into (u, v) , then this edge in P' is mapped to (u, v) . Other edges are mapped to the respective edges in G' . Note that P' may not be a not simple path, because P can have more than one edge in F .

Let $P = e_1 \dots e_\ell$ and $P' = e'_1 \dots e'_\ell$. The traversal function associated to the packet's injection path will be $f' : P' \rightarrow E^\ell$ such that $f'(e'_k) = e_k$. Without loss of generality, we can assume that the collection of distinct types is E . The packet scheduling policy remains the same and the token switching policy becomes G' 's switching policy.

Now, at time t a configuration in G is a set of packets waiting in their queues. The respective configuration in G' is the same set of packets but now the packets that are waiting in a queue of edge $e \in F$ in G are waiting in the queue of (u, v) with its corresponding type. The rest of other packets are waiting in the queue of the respective edge in G' . Furthermore a configuration in G' is a configuration in G where the packets in the queue of (u, v) are distributed among in the queues of F 's edges with the corresponding traversal type of each packet. ■

The converse of Theorem 6 is trivially true (simply duplicate each network edge, once for each possible type, and associate a single token to all copies of the same edge).

Corollary 7 *Using the same notation as in Theorem 6, a network G with latency Δ and disjoint token policy (P, S) is stable against a bounded adversary A if and only if the network system $(G', \Delta, (P, S'), A')$ is stable in the dynamic multi-type model.*

3.3 Stability Results

In this section we slightly modify universal stability definition for networks. We reduce the set of policies to declare a network universally stable, the reason is to use policies that avoid setups.

Intuitively, if an edge is not highly congested it does not matter much what type of packet is being serviced. However, for congested edges, a switching policy should choose to service a packet type if it allows for the efficient use of the edge (transmit 1 packet per time step) for a sufficiently long time interval. The length of the latter interval, say θ , will necessarily have to be a function of Δ . A moment of thought also allows one to see that it must depend on the adversary rate r and burst b parameters. Indeed, one expects that the larger r , b and Δ are the larger θ should be.

Dai and Jennings in [29] proposed the notion of *sensible* policy; given a switching policy S and a parameter θ , let S_θ denote the policy that picks, according to S , which class to service next among all those that have at least θ packets in the queue. If no class has at least θ packets waiting in the queue, then the policy may choose whatever class it prefers. Switching policies of the form S_θ are called sensible switching policies.

A sensible switching policy S_θ , for sufficiently large θ , guarantees that the inefficiencies entailed by setup costs will be amortized by an edge working at full capacity over a sufficiently long interval after the setup cost is incurred. A sensible switching policy can thus avoid paying unnecessary setup costs and maintain network stability against (r, b) -adversaries for rates arbitrarily close to 1.

Now, we say that G is *sensibly* universally stable if for all $\Delta \geq 0$, every work conserving policy (P, S) , and every (r, b) -adversary A where $r < 1$, there exists a $\theta > 0$ such that $(G, \Delta, (P, S_\theta), A)$ is stable.

Similar to the AQT model, directed acyclic graphs and unidirectional rings are sensibly universally stable, and the basis of the characterization of sensibly universally stable networks. First we prove that these two classes of networks are sensibly universally stable, and then we characterize the complete set of sensibly universally stable networks.

To establish the sensibly universal stability of directed acyclic graphs and directed rings we rely on potential function type arguments similar to those used to prove Theorem 1 in [18] and Theorem 3.7 in [11]. As is usually the case for this type of proof arguments, we upper bound the potentials independent of time, hence obtaining the desired conclusions. The bounds we derive take into account the setup cost Δ .

The potential functions introduced in the next two subsections are adaptations of the potential functions defined in [18] and [11]. But, their derived upper bounds now also depends explicitly on the value of θ , and hence indirectly on r , b , and Δ .

3.3.1 Stability of Directed Acyclic Graphs

Borodin et al. [18] established that acyclic networks are universally stable in the AQT model. We generalize their result to the AQT with setups scenario. As mentioned before, the argument we will use is a potential function argument. In the context of acyclic networks it is natural to consider potential functions which take into account the network state upstream of any given network edge e . Specifically, we will consider the total number of packets present in the network that want to traverse a given edge e at a given time step t . We will show that such a quantity remains bounded over time. As one might expect, the bound will depend on the structure of the network upstream of e . Specifically, we have the following result:

Theorem 8 *For every acyclic digraph G , all $\Delta \geq 0$, every work conserving policy (P, S) , and every (r, b) -adversary A where $r < 1$, there exists a $\theta > 0$ such that $(G, \Delta, (P, S_\theta), A)$ is stable.*

Proof: For each edge e of G let $Q_{e,i}(t)$ be the number of packets of type i at e 's queue at time step t . Let $Q_e(t)$ denote the total number of packets in the queue, i.e., $Q_e(t) = \sum_{i=1}^{|I|} Q_{e,i}(t)$. Furthermore, let θ be large enough so that $r(\theta + 2\Delta) + b \leq \theta$.

We inductively define the potential function $\psi(\cdot)$ as follow: For each edge e , denote by f_1, \dots, f_k the edges incident to e 's tail node and define

$$\psi(e) = \max\{(|\mathcal{I}| + 1)\theta, Q_e(0)\} + \sum_{j=1}^k \psi(f_j).$$

Denote by $A_e(t)$ the number of packets present in the network that want to traverse edge e and were injected by time t . We claim that $A_e(t) \leq \psi(e)$ for all e and for all $t = l(\theta + 2\Delta)$ with $l \in \mathbf{N}$. The desired result will follow once we prove the latter claim. Indeed, since $A_e(\cdot)$ can grow only by injection of new packets, one has $A_e(t') \leq A_e(t) + r(\theta + 2\Delta) + b$ for $t \leq t' < t + (\theta + 2\Delta)$. Since $\sum_e \psi(e)$ is an upper bound on the total number of packets in the graph, the theorem follows.

First we show that when e is source edge and $t = l(\theta + 2\Delta)$,

$$Q_e(t) \leq \max\{(|\mathcal{I}| + 1)\theta, Q_e(0)\}. \quad (3.3)$$

Indeed, consider the following two cases:

- $Q_{e,i}(t) \geq \theta$ for some $i \in \mathcal{I}$: In this case, the number of packets z that cross e during the interval $[t, t + \theta + 2\Delta)$ is at least $\theta \geq r(\theta + 2\Delta) + b$. We get that

$$Q_e(t + (\theta + 2\Delta)) \leq Q_e(t) + r(\theta + 2\Delta) + b - z \leq Q_e(t).$$

- $Q_{e,i}(t) < \theta$ for all $i \in \mathcal{I}$: Clearly, $Q_e(t) \leq |\mathcal{I}|\theta$, so by the choice of θ and the induction hypothesis,

$$Q_e(t + (\theta + 2\Delta)) \leq Q_e(t) + r(\theta + 2\Delta) + b \leq |\mathcal{I}| + \theta.$$

Now, to show that $A_e(t) \leq \psi(e)$ for all e and $t = l(\theta + 2\Delta)$ with $l \in \mathbf{N}$, we proceed by induction on the maximum distance of the tail of e to a source of G (i.e., the length of the longest directed path from any source to the tail of e). The base case of the induction is when the tail of e is a source. In this case $A_e(t) = Q_e(t)$ and $\psi(e) = \max\{(|\mathcal{I}| + 1)\theta, Q_e(0)\}$. Hence, the claim holds because of (3.3). Now,

assume the maximum distance from a source of G to the tail of e is $d > 0$. Let again f_1, \dots, f_k denote the edges incident to e 's tail node. Observe that

$$A_e(t) \leq Q_e(t) + \sum_{j=1}^k A_{f_j}(t). \quad (3.4)$$

Clearly, the maximum distance from a source of G to the tail of f_j is at most $d - 1$. Thus, by the induction hypothesis, $A_{f_j}(t) \leq \psi(f_j)$ for all t and for all j . Applying (3.3) we conclude that for $t = l(\theta + 2\Delta)$,

$$A_e(t) \leq \max\{(|\mathcal{I}| + 1)\theta, Q_e(0)\} + \sum_{j=1}^k \psi(f_j) = \psi(e).$$

■

Corollary 9 *Directed acyclic graphs are sensibly universally stable.*

3.3.2 Stability of the Ring

Andrew et al. [11] established that the simplest cyclic networks (directed rings) are universally stable in the AQT model. We extend their result to the AQT with setups scenario.

Let R_n denote the n -node directed ring. Assume that R_n 's nodes are labeled $0, 1, \dots, n - 1$ and that its edges are directed from i to $i + 1$ (all arithmetic on labels is modulo n). The edge emanating from node i will be referred in the following as edge i

Theorem 10 *For every $\Delta \geq 0$, every work conserving policy (P, S) , and every (r, b) -adversary A where $r < 1$, there exists a $\theta > 0$ such that $(R_n, \Delta, (P, S_\theta), A)$ is stable.*

Proof: The proof is by contradiction. For an adversary A we fix θ and suppose that the system is not stable, i.e., there exist an edge e such that for every positive integer Q there is a time t such that $A_e(t) > Q$.

Let $P_{e,i}(t)$ denote the set of packets of type i that want to cross e at time t , and which are in a queue together with at least θ packets of type i . Let $P_e(t) = \bigcup_{i \in I} P_{e,i}(t)$, and for each $p \in P_e(t)$, let $l_p(t)$ be the maximum positive integer such that $p \in P_e(t-j)$ for all $j \in \{0, 1, \dots, l_p(t)\}$.

Let A be an (r, b) -adversary, where $r < 1$. Fix θ such that $r = \frac{\theta}{\theta+2\Delta} - \varepsilon$, where $\varepsilon > 0$. We take Q' large enough and fix its exact value later. Let T' be the first moment at which $|P_e(T')| > Q'$, for some e in G . Let $p \in P_e(T')$ be such that $l_p(T') \geq l_q(T')$ for all $q \in P_e(T')$.

Without loss of generality, we assume that p was at the queue of edge 0 in the moment $T_0 := T' - l_p(T')$. Let $0, 1, \dots, s$ be the nodes visited by p during the interval $[T_0, T']$, and let T_i denote the time at which p arrives at the queue of the edge i . By abuse of notation, we also write $T_{s+1} = T'$.

Notice that for all $k \in \{0, 1, \dots, s\}$ and $t \in (T_k, T_{k+1}]$ edge k works as best as possible, i.e., it works at rate $r + \varepsilon$. Furthermore notice that for all e in G , and for all $t' \leq t$ we have that

$$|P_e(t)| \leq |P_e(t')| + r(t' - t) + b + |\mathcal{I}|\theta n - z \quad (3.5)$$

where z is the number of packets sent across edge e during the interval $[t', t)$.

We define

$$Q = \max_{e \in G, t \in [T_0, T']} P_e(t). \quad (3.6)$$

Notice that $Q \leq Q'$.

For e and $t \geq T_0$ as before, we now define the function f as $f(e, T_0) = Q + e(b + 2|\mathcal{I}|\theta n)$, and $f(e, t) = Q - \varepsilon(t - T_0) + (b + 2|\mathcal{I}|\theta n)(e + 1)$, for $t > T_0$. We note the following properties of this function f hold:

$$(i) \quad f(e, t) = f(e, T_0) - \varepsilon(t - T_0) + (b + 2|\mathcal{I}|\theta n), \quad \forall t > T_0. \quad (3.7)$$

$$(ii) \quad f(e, t) = f(e, t') - \varepsilon(t - t'), \quad \forall t > t' > T_0. \quad (3.8)$$

$$(iii) \quad f(e, t) = f(e - 1, t) + (b + 2|\mathcal{I}|\theta n), \quad \forall t > T_0, \quad \forall e > 0. \quad (3.9)$$

If e is an edge of G and t is a time step, we say that the pair (e, t) is *applicable* if either $e = i \in \{0, 1, \dots, s\}$ and $t \in [T_0, T_e)$, or $e > s$ and $t \in [T_0, T']$.

Remark 3 Note that if (e, t) is applicable and $(e - 1, t)$ is not, then $t \in [t_e, T_e + 1)$.

The crux of our analysis is the following lemma.

Remark 4 For all applicable pairs (e, t) , we have $|P_e(t)| \leq f(e, t)$.

Remark 4 can be proved by induction on $e \geq 0$, and for fixed e for all $t \geq T_0$. First, establish the result for $e = 0$ and for all t , which is the first step of the induction. Notice that $(0, t)$ is applicable if and only if $t \in [T_0, T_1]$. Moreover, we have that $f(0, T_0) = Q \geq |P_0(T_0)|$. Also note that, since the packet p arrives to $e = 0$ at time T_0 and p leaves it at time T_1 , there is a continuous flow of packets sent across $e = 0$ during the interval $[T_0, T_1)$. Then, by Equations (3.5) and (3.7) - (3.9), and the induction hypothesis,

$$\begin{aligned} |P_0(t)| &\leq |P_0(T_0)| + r(t - T_0) + b + |\mathcal{I}|\theta n - (r + \varepsilon)(t - T_0) \\ &\leq f(0, T_0) - \varepsilon(t - T_0) + b + |\mathcal{I}|\theta n \\ &\leq f(0, t). \end{aligned}$$

Now we want to prove the claim for all $e > 0$, for this we use induction on e . We assume, by induction hypothesis, that $|P_{e-1}(t)| \leq f(e - 1, t)$ for all applicable $(e - 1, t)$. For T_0 we have that (e, T_0) is applicable, and that $f(e, T_0) = Q + e(b + 2|\mathcal{I}|\theta n) \geq Q \geq |P_e(T_0)|$.

Now, consider any applicable pair (e, t) , with $t > T_0$. If for all $t' \in [T_0, t)$ there is $i \in \mathcal{I}$ such that $Q_{e,i}(t') \geq \theta$, again, by Equations (3.5) and (3.7) - (3.9), and the induction hypothesis, we have

$$\begin{aligned} |P_e(t)| &\leq |P_e(T_0)| + r(t - T_0) + b + |\mathcal{I}|\theta n - (r + \varepsilon)(t - T_0) \\ &\leq f(e, T_0) - \varepsilon(t - T_0) + b + |\mathcal{I}|\theta n \\ &\leq f(e, t). \end{aligned}$$

Otherwise, there is some step $t' \in [T_0, t)$ such that for all $i \in \mathcal{I}$ $Q_{e,i}(t') < \theta$. Let us take the largest such $t' < t$. Notice that $|P_e(t')| \leq |P_{e-1}(t')| + |\mathcal{I}|\theta$. Again, applying

Equations (3.5) and (3.7) - (3.9), and the induction hypothesis, we have

$$\begin{aligned}
|P_e(t)| &\leq |P_e(t')| + r(t - t') + b + |\mathcal{I}|\theta n - (r + \varepsilon)(t - t') \\
&\leq |P_{e-1}(t')| + |\mathcal{I}|\theta - \varepsilon(t - t') + b + |\mathcal{I}|\theta n \\
&\leq f(e - 1, t') - \varepsilon(t - t') + b + |\mathcal{I}|\theta n + |\mathcal{I}|\theta \\
&= f(e - 1, t) + b + |\mathcal{I}|\theta n + |\mathcal{I}|\theta \\
&\leq f(e, t).
\end{aligned}$$

Now we fix

$$Q' = \left(\frac{1}{\varepsilon} \cdot (e + 1)(b + 2|\mathcal{I}|\theta n) + 1\right)r + b + |\mathcal{I}|\theta n.$$

Notice that since the amount of packets that want to use e at time T' and that are in a queue of size bigger than θ are bounded by $Q' < |P_e(T')| \leq r(T' - T_0 + 1) + b + |\mathcal{I}|\theta n$, then $T' - T_0 \geq \frac{Q' - b - |\mathcal{I}|\theta n}{r} - 1$. Using the Remark 4 we have

$$\begin{aligned}
Q' < |P_e(T')| &\leq f(e, T') \\
&= Q - \varepsilon(T' - T_0) + (e + 1)(b + 2|\mathcal{I}|\theta n) \\
&\leq Q' - \varepsilon\left(\frac{Q' - b - |\mathcal{I}|\theta n}{r} - 1\right) + (e + 1)(b + 2|\mathcal{I}|\theta n) \\
&\leq Q'
\end{aligned}$$

which is a contradiction. ■

Corollary 11 *Unidirectional rings are sensibly universally stable.*

3.4 Characterization of the stable networks

In this section we characterize networks which are sensibly universally stable under the AQT with setups model.

First we present a result presented in [7] used in the characterization. Note that in our characterization we are interested sensibly universally stable networks and the next result is for universal stability. Nevertheless universal stability is more general than sensible universal stability.

Lemma 12 ([7]) *If digraphs G_1 and G_2 are universally stable, then so is any graph formed by joining them with edges that go only from G_1 to G_2 .*

As a consequence of the previous lemma we have the sensible version of the result presented in [7] for universal stable networks, following theorem.

Theorem 13 *A digraph G is sensible universally stable if and only if all its strongly connected components are sensible universally stable.*

In [7], a characterization of universally stable networks in the AQT model is given. In order to such characterization, Álvarez et al. detect the simplest non universally stable networks. Then, define subdivision operations, such that applying those operations iteratively to the simplest non universally stable networks, a family of non universally stable networks is created. Finally, it is proved that this family is the set of non universally stable networks.

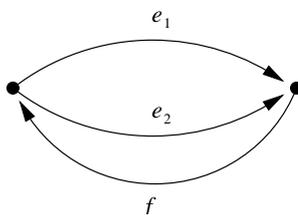
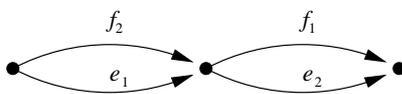
Definition 4 (Subdivision operations [7])

- *The subdivision of an arc (u, v) in a digraph G consists in the addition of a new vertex w and the replacement of (u, v) by (u, w) and (w, v) .*
- *The subdivision of a 2-cycle $(u, v), (v, u)$ in a digraph G consists in the addition of a new vertex w and the replacement of (u, v) and (v, u) by the arcs $(u, w), (w, u), (v, w)$ and (w, v) .*

Given a digraph G , let us denote by $\varepsilon(G)$ the family of digraphs formed by G and all the digraphs obtained from G by successive arc or 2-cycle subdivisions.

Since the AQT with setups model is an extension of the AQT model, then if G is not universally stable in the AQT model implies that it is not universally stable under the AQT with setups model model, so we can apply all the unstable results that are in [7].

Theorem 14 ([7]) *All the digraphs in $\varepsilon(U_1) \cup \varepsilon(U_2)$ are not stable under the NTG-LIS protocol.*

Figure 3.1. U_1 Figure 3.2. U_2

Then the characterization of sensible universally stable networks is stated in the following theorem:

Theorem 15 *A digraph is sensibly universal stable if and only if it does not contain as a subgraphs any of the digraphs in $\varepsilon(U_1) \cup \varepsilon(U_2)$.*

Proof: If G contains as a subgraph some digraph in $\varepsilon(U_1) \cup \varepsilon(U_2)$, then by Theorem 14, and by the fact that if a subgraph of G is not stable then G is not stable, we have that G is not stable. Now if G does not contain as a subgraph a digraph in $\varepsilon(U_1) \cup \varepsilon(U_2)$, then all strongly connected components must consist of simple cycles. By the fact that acyclic digraphs and directed cycles on any number of vertices are sensibly universally stable, and Theorem 13 we have that G is sensibly universally stable. ■

3.5 Instability Results

In this section we prove that stability does not trivially carry over from the single-class (i. e., the AQT model) to the multi-class setting. We show that universally stable policies do not exist by proving the following theorem:

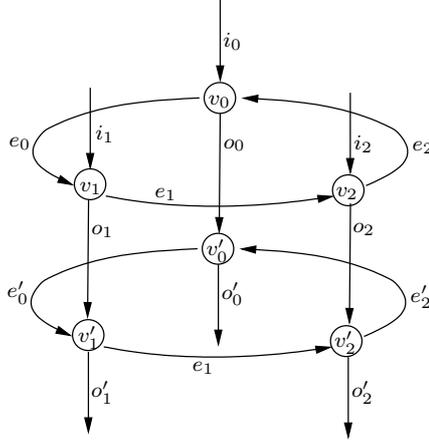


Figure 3.3. Two concatenated gadgets.

Theorem 16 For all $0 < r < 1$, $\Delta \geq 0$, $\theta > 0$, and every greedy policy (P, S) where S is shift-oblivious, there exists a network G (whose size depends on r , Δ , and θ), and an (r, b) -adversary A , such that $(G, \Delta, (P, S_\theta), A)$ is unstable.

The proof is a construction of a network and an adversary that create instability under every policy. The network and the adversary are simplifications of the construction presented in [16]. The network is built from a basic gadget which “slows” packets that need to traverse it. The delays allow the adversary to inject packets so as to make them arrive simultaneously at some other network spot. Without loss of generality we fix the setup cost Δ to 0.

The Network: The basic structure of the network is a *gadget* (Figure 3.3). A gadget has nodes v_0, v_1, v_2 , and its edges are of three distinct types; (1) input edges i_0, i_1, i_2 , (2) output edges o_0, o_1, o_2 , and (3) load edges e_0, e_1, e_2 where e_j goes from v_j to v_{j+1} (arithmetic on subindices will henceforth be mod 3). Note that load edges form a unidirectional ring.

We say that a gadget G_2 is concatenated to gadget G_1 if the output edges of G_1 are the inputs edges of G_2 . One gadget can be concatenated to several gadgets and more than one gadget can be concatenated to another gadget. This can be done by

having several input or output edges. We say that $C = \langle H_1, H_2, \dots, H_n \rangle$ is a *chain of length n* if H_{i+1} is concatenated to H_i . A chain $B = \langle G_1, H_1, \dots, H_m, G_2 \rangle$ will be called a *bridge of length m* between G_1 and G_2 .

The main network components are: columns, connectors and shortcuts. There are two columns, each one is a chain of length α , namely,

$$C_1 = \langle C_{1,1}, C_{1,2}, \dots, C_{1,\alpha} \rangle$$

and

$$C_2 = \langle C_{2,1}, C_{2,2}, \dots, C_{2,\alpha} \rangle.$$

Connectors are bridges of length β between each gadget of a column and the other column's first gadget; say for each $C_{1,i}$ there is a bridge

$$\langle C_{1,i}, D_{1,i,1}, D_{1,i,2}, \dots, D_{1,i,\beta}, C_{2,1} \rangle$$

and similarly, for each $C_{2,i}$ there is another bridge

$$\langle C_{2,i}, D_{2,i,1}, D_{2,i,2}, \dots, D_{2,i,\beta}, C_{1,1} \rangle.$$

The values of α and β will be fixed later. Finally, shortcuts are bridges of length 1 from each connector gadget $D_{1,i,j}$ to $C_{2,1}$ and they are denoted by $E_{1,i,j}$. Similar shortcuts exist from $D_{2,i,j}$ to $C_{1,1}$ and they are denoted by $E_{2,i,j}$. A sketch of the network can be seen in Figure 3.4

The Adversarial Injection: Informally, we can describe the adversarial injection in the following way: first, we assume that there are l packets going down through a column. While these packets are crossing a gadget of that column, the adversary injects packets that want to cross through its respective bridge to the first gadget of the other column, and then go down through it. Finally, using the shortcuts we obtain that all these packets arrive at the same time to the first gadget of the other column. Since there are more than l of these packets we can conclude instability.

For each gadget we define 3 basic injection routes referred to as *gadget-traversing* packets. These packets enter the gadget through the input edge i_j , traverse the load

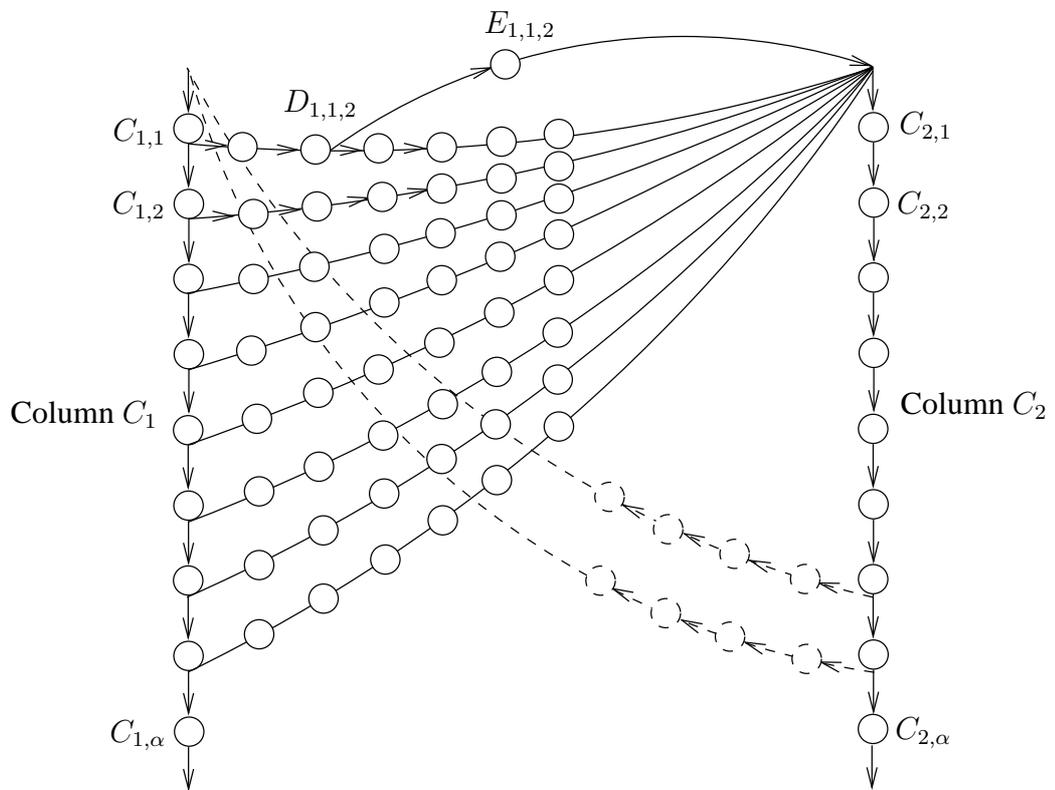


Figure 3.4. Unstable network.

edges e_j, e_{j+1} , and exit the gadget through edge o_{j+2} . A packet is called *chain-traversing* for a chain $\langle H_1, H_2, \dots, H_k \rangle$ if its route is gadget-traversing for each H_j .

To describe the packet injection pattern we introduce the notion of *gadget activation*, which encompasses three phases: precondition, injection and postcondition. In the precondition phase, we assume that for l consecutive time steps there will be a gadget-traversing packet at the queue of each gadget input edges. Moreover, the packets are chain-traversing for the chain $\langle C_{\gamma,i}, C_{\gamma,i+1}, \dots, C_{\gamma,\alpha} \rangle$ where γ is the column to which the gadget belongs. The injection phase starts when the first packet of the precondition uses its respective load edge and lasts l time steps. During the injection phase, the adversary injects packets in the load edges at rate $r/3$. The postcondition phase corresponds to the precondition assumption for the activation of the next concatenated gadget.

The adversary injects 6 different packet classes: $1, 1', 2, 2', 3$ and $3'$; we associate types without prima to column C_1 and types with prima to column C_2 . Time steps are grouped into stages. At the beginning of an even stage there is a guarantee that for l consecutive times steps a chain-traversing (for column C_1) packet of type p will be at i_p 's queue, where i_p is an input edge of gadget $C_{1,1}$. We show that at the beginning of the next odd stage there is a guarantee that for more than l consecutive times steps a chain-traversing (for column C_2) packets of type p' will be at each of the input edges $i_{p'}$ of gadget $C_{2,1}$. Each stage is divided into α sub-stages. During the j -th sub-stage, gadget $C_{1,j}$ is activated.

During the j -th sub-stage, a total of $rl/3$ packets of type p are injected along the path whose first edge is the load edge e_p of gadget $C_{1,j}$. The routes of these packets are chosen so that, for some $m < \beta$, they are chain-traversing for the chain $\langle D_{1,j,1}, D_{1,j,2}, \dots, D_{1,j,m}, E_{1,j,m}, C_{2,1}, C_{2,2}, \dots, C_{2,\alpha} \rangle$. We will see that using shortcuts one can have all these packets simultaneously arrive at the queues of the input edges of gadget $C_{2,1}$.

Instability of the System: We set $\alpha = 3/l$, $\beta > (1 + 3/r)\alpha$, and l such that $r^2l/9 \geq \theta$.

During the first l time steps of gadget activation, load edges are used by column traversing packets, and the injection of new packets is performed. After this, the switching protocol is which decides if the load edges will serve new injected packets or column traversing packets (both satisfies the sensible threshold).

If new packets are served, gadget activation will last $l + lr/3$. Otherwise it will take l time steps to reach the postcondition. We can conclude this, since S is a shift-oblivious policy, in no case one of the edges e_0 , e_1 or e_2 will chose to serve column traversing packets and other edge will chose to serve newly injected packets.

For connector gadgets the argument is the same but substituting l by $lr/3$, observing that, by hypothesis, $(lr/3)r/3 > \theta$ and recalling that no packet is injected in internal connector gadgets. Thus, their activation period lasts $lr/3$.

Activation of gadget $D_{1,j,\beta}$ is completed before $lr\beta/3$ time steps. It is after activation of gadget $C_{1,\alpha}$, which is completed at most in $(1 + 3/r)\alpha$ time steps. Since β is large enough, it is possible to route all the connector chain-traversing packets so they will simultaneously arrive to $C_{2,1}$. Finally, note that when the stage finishes, precondition for $C_{2,1}$ can be accomplished for each of the input edge's queue with at least $\alpha lr/3 = 2l$ packets.

3.6 Fluid Model

In this section we show that a result of Gamarnik presented in [36] can be extended to the multi-type with setups scenario.

Consider an adversarial queuing system with setups $(G, \Delta, (P, S), A)$. Let \mathcal{R} denote the set of simple paths in G . For each path $P \in \mathcal{R}$ of length $\ell(P)$ let $e_0^P \dots e_{\ell(P)}^P$ be the set of consecutive edges in P . Let $N_{P,i}(t_1, t_2)$ be the total number of packets of type i that the adversary injects along path P during the interval $[t_1, t_2)$ and let $B_{e,i}(t_1, t_2)$ be the total number of packets of type i that arrive at the tail node of edge e during $[t_1, t_2)$. Also, define $D_{P,e,i}(t_1, t_2)$ as the total number of type i packets following P that cross e during $[t_1, t_2)$. Let $Q_{e,i}(t)$ be the total number of type i packets that are waiting at e 's queue at time t and let $S_{e,i}(t_1, t_2)$ be the total setup time incurred by e during $[t_1, t_2)$. Finally, let $Y_e(t_1, t_2)$ be the idle time of e during $[t_1, t_2)$.

For the sake of simplicity we denote $N_{P,i}(0, t)$ by $N_{P,i}(t)$ and adopt a similar convention for B , D , S , and Y . When a sub-index is missing for some previously defined quantity, it is to be understood that summation is over the range of the missing index, e.g., $N_{e,i}(t) = \sum_{P: e \in P} N_{P,i}(t)$ and $Q_e(t) = \sum_{i \in \mathcal{I}} Q_{e,i}(t)$.

Discrete Model Equations: Let \mathcal{R}_e denote the set of paths starting with edge e and $\mathcal{R}_{f,e}$ the set of paths where e immediately follows f . Up to time step t , the number of type i packets that are injected by the adversary (respectively, arrive from some other node in the network) at e 's queue is $\sum_{P \in \mathcal{R}_e} N_{P,i}(t)$ ($\sum_{f \in T(e)} \sum_{P \in \mathcal{R}_{f,e}} D_{P,f,i}(t)$, where $T(e)$ denotes the set of edges incident to the node tail of e , respectively). Thus, it is easy to see that the dynamics of the adversarial network is fully described by the load and setup constraints (equations (3.1) and (3.2) respectively) and the following equations are valid for each $i \in \mathcal{I}$, $P \in \mathcal{R}$, and $e \in E$:

$$B_{e,i}(t) = \sum_{P \in \mathcal{R}_e} N_{P,i}(t) + \sum_{f \in T(e)} \sum_{P \in \mathcal{R}_{f,e}} D_{P,f,i}(t), \quad (3.10)$$

$$Q_{e,i}(t) = Q_{e,i}(0) + B_{e,i}(t) - D_{e,i}(t), \quad (3.11)$$

$$t = S_e(t) + D_e(t) + Y_e(t). \quad (3.12)$$

Let $\vec{N}(t) = (N_{P,i}(t))_{P,i}$, $\vec{B}(t) = (B_{e,i}(t))_{e,i}$, $\vec{Y}(t) = (Y_e(t))_e$, $\vec{Q}(t) = (Q_{e,i}(t))_{e,i}$, $\vec{D}(t) = (D_{P,e,i}(t))_{P,e,i}$ and $\vec{S}(t) = (S_{e,i}(t))_{e,i}$. The time dependent vector sequence $(\vec{N}, \vec{B}, \vec{D}, \vec{Q}, \vec{S}, \vec{Y})$ is called a *realization* of the discrete model if it satisfies (3.1)-(3.2), (3.10) - (3.12).

For multi-type queuing networks, we say that a realization is a *greedy* or *work-conserving realization* if no edge is idle when its queue is non-empty. Formally, if $Q_e(t) > 0$ for all $t \in [t_1, t_2)$, then $Y_e(t_1) = Y_e(t_2)$. Moreover, a realization is called *stable* if the number of packets in the system stays bounded over time, i.e., $\sup_t \sum_{e \in E} Q_e(t) < \infty$. Finally, we say that an adversarial multi-type queuing network G with setup cost Δ is *universally stable* against adversary A if it is stable for every work conserving realization.

Fluid Model Equations: Fluid models of queuing networks correspond to continuous approximation of queuing networks. In these models one associates to relevant

network parameters, time-dependent, continuous, non-decreasing and non-negative real-valued functions. For the routing network of the preceding section, we consider for each edge e , path P and type i , the following functions: $\widehat{N}_{P,i}(t)$, $\widehat{B}_{e,i}(t)$, $\widehat{D}_{P,e,i}(t)$, $\widehat{Q}_{e,i}(t)$, $\widehat{S}_{e,i}(t)$, and $\widehat{Y}_e(t)$. We also define the vectors $\widehat{N}(t)$, $\widehat{B}(t)$, $\widehat{D}(t)$, $\widehat{Q}(t)$, $\widehat{S}(t)$ and $\widehat{Y}(t)$ as in the discrete model. A continuous mapping from t to $(\widehat{N}(t), \widehat{B}(t), \widehat{D}(t), \widehat{Q}(t), \widehat{S}(t), \widehat{Y}(t))$ is called a *fluid solution* for the adversarial multi-type model if it satisfies the following set of equations: For each $i \in \mathcal{I}$, $P \in \mathcal{R}$, and $e \in E$,

$$\widehat{N}_P(t) \leq r \cdot t, \quad (3.13)$$

$$\widehat{B}_{e,i}(t) = \sum_{P \in \mathcal{R}_e} \widehat{N}_{P,i}(t) + \sum_{f \in T(e)} \sum_{P \in \mathcal{R}_{f,e}} \widehat{D}_{P,f,i}(t), \quad (3.14)$$

$$\widehat{Q}_{e,i}(t) = \widehat{Q}_{e,i}(0) + \widehat{N}_{e,i}(t) - \widehat{D}_{e,i}(t), \quad (3.15)$$

$$t = \widehat{S}_e(t) + \widehat{D}_e(t) + \widehat{Y}_e(t). \quad (3.16)$$

A fluid solution is called *work conserving fluid solution* if $\widehat{Q}_e(t) > 0$ for all $t \in [t_1, t_2]$ implies that $\widehat{Y}_e(t_1) = \widehat{Y}_e(t_2)$. A fluid solution $(\widehat{N}(t), \widehat{B}(t), \widehat{D}(t), \widehat{Q}(t), \widehat{S}(t), \widehat{Y}(t))$ is called *stable* if there is some time τ after which the fluid network stays empty, i.e., $\widehat{Q}(t) = 0$ for all $t \geq \tau$. A fluid network is called *globally stable* if for any fluid solution with initial vector of queue lengths $\widehat{Q}(0)$ having $\sum_e \widehat{Q}_e(0) = 1$, there is a time τ such that $\widehat{Q}(t) = 0$ for all $t \geq \tau$.

This section's main result is an analogue of the one due to Gamarnik [36, Theorem 1] concerning the single-type case. Specifically we prove the following:

Theorem 17 *In the AQT model with setups, if $(G, \Delta, (P, S); A)$ is a stable network system whose associated fluid network is globally stable, then $(G, \Delta, (P, S); A)$ is stable.*

Proof: Consider a realization $(\vec{A}, \vec{B}, \vec{D}, \vec{Q}, \vec{S}, \vec{Y})$ that satisfies (3.1) - (3.2) and (3.10) - (3.12). From Gamarnik [36, Proposition 1] we deduce that for a non-decreasing sequence of integer times $t_1, t_2, \dots, t_k, \dots$ there exists a sequence of positive integers $k_1, k_2, \dots, k_n, \dots$ such that for any non-negative real number t the following quanti-

ties converge as n goes to infinity:

$$\begin{aligned}\widehat{Q}_e(t) &= \lim_{n \rightarrow \infty} \frac{1}{k_n} Q_e(t_{k_n} + tk_n), \\ \widehat{A}_e(t) &= \lim_{n \rightarrow \infty} \frac{1}{k_n} (A_e(t_{k_n} + tk_n) - A_e(t_{k_n})), \\ \widehat{D}_e(t) &= \lim_{n \rightarrow \infty} \frac{1}{k_n} (D_e(t_{k_n} + tk_n) - D_e(t_{k_n})), \\ \widehat{B}_e(t) &= \lim_{n \rightarrow \infty} \frac{1}{k_n} (B_e(t_{k_n} + tk_n) - B_e(t_{k_n})).\end{aligned}$$

Moreover, the limits satisfy (3.13)-(3.15). Denote each of these limits by $\widehat{Q}_e(t)$, $\widehat{A}_e(t)$, $\widehat{D}_e(t)$ and $\widehat{B}_e(t)$ respectively. We now show that we can associate to $S_e(\cdot)$ and $Y_e(\cdot)$ fluid limits which together with $\widehat{A}(\cdot)$, $\widehat{B}(\cdot)$, $\widehat{D}(\cdot)$, and $\widehat{Q}(\cdot)$ yield a work conserving fluid solution if $(\vec{A}, \vec{B}, \vec{D}, \vec{Q}, \vec{S}, \vec{Y})$ is a work conserving realization. Indeed, note that from (3.12) we get that for any non-decreasing integer sequence t_1, t_2, \dots and positive rational number q ,

$$\frac{1}{k} (S_e(t_k + qk) - S_e(t_k)) \leq \frac{qk}{k} = q.$$

Thus, again by [36, Proposition 1], we conclude that the fluid limit $\widehat{S}_e(\cdot)$ exists. A similar argument shows that the fluid limit $\widehat{Y}_e(\cdot)$ exists. It is easy to check that \widehat{A} , \widehat{B} , \widehat{D} , \widehat{Q} , \widehat{S} , and \widehat{Y} must satisfy (3.13)-(3.16), i.e. $(\widehat{A}, \widehat{B}, \widehat{D}, \widehat{Q}, \widehat{S}, \widehat{Y})$ is a fluid solution of the fluid model.

We now want to show that the fluid solution is work conserving. For the sake of contradiction, suppose that $\widehat{Q}_e(t) > 0$ for all $t \in [t_1, t_2]$. Since $\widehat{Q}_e(\cdot)$ is continuous and $[t_1, t_2]$ is a compact set, there is a $\gamma > 0$ such that $\widehat{Q}_e(t) > \gamma$ for all $t \in [t_1, t_2]$. But $\widehat{Q}_e(t) = \lim_{n \rightarrow \infty} Q_e(t_{k_n} + tk_n)/k_n$, hence $Q_e(t_{k_n} + tk_n) > \gamma k_n$ for sufficiently large n . This implies, since the discrete realization is work conserving, that $Y_e(t_{k_n} + t_1 k_n) = Y_e(t_{k_n} + t_2 k_n)$ for sufficiently large n , hence $\widehat{Y}_e(t_1) = \widehat{Y}_e(t_2)$.

Because of the global stability of G 's associated fluid network, we have that there is a time τ such that $\widehat{Q}(t) = 0$ for all $t \geq \tau$ for any fluid solution with initial vector of queue lengths $\widehat{Q}(0)$ for which $\sum_e \widehat{Q}_e(0) = 1$. However, this is impossible if $(\vec{A}, \vec{B}, \vec{D}, \vec{Q}, \vec{S}, \vec{Y})$ is an unstable realization [36, Lemma 3]. \blacksquare

Chapter 4

Continuous AQT Model

4.1 Introduction

In this chapter we present our contributions to the CAQT model. For convenience of the reader, we explain again the model and explain in detail some ideas presented in Chapter 1.

The CAQT model represents a network as a finite directed graph G in which the set of nodes $V(G)$ represent the hosts, and the set of edges $E(G)$ represent the links between those hosts. Each link $e \in E(G)$ in this graph has associated a positive but not infinite transmission speed (a bandwidth), denoted as B_e . The bandwidth of a link establishes how many bits per second can be transmitted by the link. Instead of considering the bandwidth as a synonym for parallel transmission, we relate the bandwidth to the transmission velocity. We consider that only one bit can be put in a link $e \in E(G)$ at each time, and that conceptually the sender puts the associated signal level to the corresponding bit for $1/B_e$ seconds for each bit. This means that a bit can be partially transmitted or partially received at a given time. Let us henceforth denote as $B_{\min} = \min_{e \in E(G)} B_e$ and as $B_{\max} = \max_{e \in E(G)} B_e$ the minimum and maximum bandwidth, respectively, of the edges in G .

Each link $e \in E(G)$ has associated a propagation delay, denoted here as P_e , where $P_e \geq 0$. This delay, measured in seconds, establishes how long it takes for a signal (the start of a bit, for instance) to traverse the link. This parameter is related to the propagation speed of the changes in the signal that carry the bits along the physical

medium used for the transmission. We will denote as $P_{\min} = \min_{e \in E(G)} P_e$ and $P_{\max} = \max_{e \in E(G)} P_e$ the minimum and maximum propagation delay, respectively, of the edges in G .

We assume the existence of an adversary that defines the traffic pattern of the system by choosing when and where to inject packets into the system, and the path to be followed by each of them. We assume that a packet path is edge-simple, in the sense that it does not contain the same edge more than once (it can visit the same vertex several times, though). Again, we restrict the adversary so that it can not trivially overload any link. To do so, we also define two system-wide parameters: the *injection rate* r (with $0 < r \leq 1$), and the *burstiness* b (with $b \geq 1$). For every link $e \in E(G)$, if we denote by $N_e(I)$ the total size (in bits) of the packets injected by the adversary in the interval I whose path contains link e , it must be satisfied that

$$N_e(I) \leq r|I|B_e + b.$$

We call an adversary A that satisfies this restriction an (r, b) -adversary. The injection rate r is sometimes expressed alternatively as $(1 - \varepsilon)$, with $\varepsilon \geq 0$.

Regarding packet injections, we assume that the adversary injects packets instantaneously. From the above restriction, this implies that packets have a maximum size of b bits. In general, we will use L_p to denote the length (in bits) of a packet p , and $L_{\max} = \max_p L_p \leq b$ to denote the maximum packet length. Observe that, once a packet p starts being transmitted through a link $e \in E(G)$, it will only take $P_e + L_p/B_e$ units of time more until it crosses it completely.

Let us now look at the packet switching process. We assume that each link has associated an *output queue*, where the packets that have to be sent across the link are held. The still unsent portion of a packet that is being transmitted is also held in this queue. In fact, if a bit has only been partially sent, we assume that the still unsent portion of the bit still resides in this queue. A packet can arrive to a node either by direct injection of the adversary or by traversing some incoming link. In the latter case we assume that only full packets are dispatched (moved to an output queue). Hence, we assume that each link has a *reception buffer* in the receiving node where the portion of a partially received packet is held. As soon as the very last bit of a

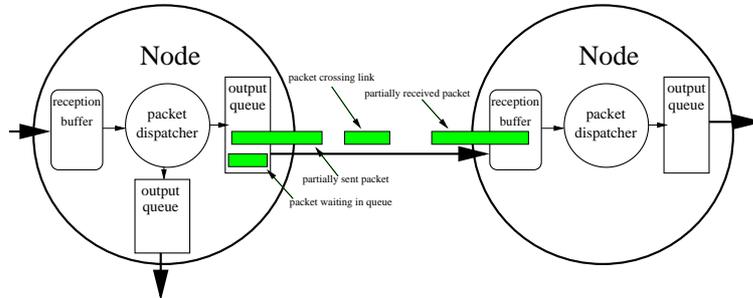


Figure 4.1. Elements involved in the nodes and links of the network in the CAQT model.

packet is completely received, the packet is dispatched instantaneously (by a packet dispatcher) to the corresponding output queue (or removed, if this is the final node of the packet). Figure 4.1 shows the previously described network elements.

The definition of *stability* in the CAQT model is analogous to the definitions stated under other adversarial models.

Definition 5 Let G be a network with a bandwidth B_e and a propagation delay P_e associated to each link e , P be a scheduling policy, and A an (r, b) -adversary, with $0 < r \leq 1$ and $b \geq 1$. The system (G, P, A) is stable if at every moment, the total number of packets (or, equivalently, the total number of bits) in the system is bounded by a value C that can depend on the system parameters, but is independent of time.

We also use standard definitions of *universal stability*. We say that a scheduling policy P is universally stable if the system (G, P, A) is stable for each network G and each (r, b) -adversary A , with $0 < r < 1$ and $b \geq 1$. Similarly, we say that a network G is universally stable if the system (G, P, A) is stable for each *greedy* scheduling policy¹ P and each (r, b) -adversary A , with $0 < r < 1$ and $b \geq 1$.

¹Three types of packets may wait to traverse a link in a particular instant of time: the incoming packets arrived from adjacent links, the packets injected directly into the link, and the packets that could not be forwarded in previous steps. At each moment, only one packet from those waiting is forwarded through the link; the rest are kept in the output queue.

4.2 Universal Stability of Rings

We study now the simplest networks which include some cyclicity: directed rings. Let G denote the n -node ring with $V(G) = \{0, 1, \dots, n-1\}$, $E(G) = \{(i, (i+1) \bmod n) : i \in V(G)\}$ and A any (r, b) -adversary with $r = 1 - \varepsilon < 1$ and $b \geq 1$. We show that the rings are universally stable in the CAQT model.

The proof follows the lines of the proof of a similar result for the AQT model [11]. To end up with such result, we need to previously derive some partial results in the form of lemmas. Let us consider some packet p in the system (G, A, P) . We suppose that it was injected in instant T_0 at the node 0 with some node d as destination. Let $T' > T_0$ be some time at which it was not yet absorbed. Let $0, 1, \dots, s$ be the nodes visited by p in the interval $[T_0, T']$. The edge $(i, i+1)$ will be named edge i for simplicity (where the sum is mod n). For $l = 0, \dots, s$, let T_l denote the time at which p arrives at the output buffer of the edge l ; by abuse of notation, we will also write $T_{s+1} = T'$.

For j an edge of G , and $t \in [T_0, T']$, we define $A_t(j)$ as in Section 1.2, again the bits in $Q_t(j)$ are included in $A_t(j)$ but those in $R_t(j)$ are not. We have the following basic property of $A_t(j)$.

Lemma 18 *Let t and t' be such that $t' \leq t$. Then*

$$A_t(j) \leq A_{t'}(j) + (1 - \varepsilon)(t - t')B_j + b - z,$$

where z is the number of bits sent across edge j in the interval $[t', t]$.

We define

$$Q = \max_{j \in G, t \in [T_0, T']} A_t(j). \quad (1)$$

For j and t as before, we now define the function f as $f(j, T_0) = Q + bj$, and $f(j, t) = Q - \varepsilon(t - T_0)B_{min} + (b + 2P_{max}B_{max} + L_{max})(j + 1)$, for $t > T_0$. Note that f satisfies the following properties:

Lemma 19

- (i) $f(j, t) = f(j, T_0) - \varepsilon(t - T_0)B_{min} + b + (1 + j)(2P_{max}B_{max} + L_{max}), \forall t > T_0.$
(ii) $f(j, t) = f(j, t') - \varepsilon(t - t')B_{min}, \quad \forall t > t' > T_0.$
(iii) $f(j, t) = f(j - 1, t) + b + 2P_{max}B_{max} + L_{max}, \quad \forall t > T_0 \quad j \geq 1.$

Definition 6 *If j is an edge of G and t is a time step, we say that the pair (j, t) is applicable if either*

$$(j \in \{0, 1, \dots, s\} \text{ and } t \in [T_0, T_{j+1}]) \text{ or } (j > s \text{ and } t \in [T_0, T']).$$

Note the following basic property of applicability.

Lemma 20 *If j 's output buffer is empty at a time $t' \in [T_0, T_{j+1} - P_j]$, then $(j - 1, t')$ is applicable.*

Proof: First we can notice that t' can not be in $[T_j, T_{j+1} - P_j]$ because the packet p arrived to j 's output buffer at time T_j and the last bit of p fully leaves it at time $T_{j+1} - P_j$, then j 's output buffer can not be empty during this interval. Now, if $t' \in [T_0, T_j]$ then the pair $(j - 1, t')$ is applicable by Definition 6. ■

The crux of our analysis relies in the following lemma.

Lemma 21 *For all applicable pairs (j, t) , we have $A_t(j) \leq f(j, t)$.*

Proof: We prove the lemma by induction on $j \geq 0$, and for fixed j for all t . First, we show that the claim holds for $j = 0$ and for all t . This is the first step of the induction. Notice that $(0, t)$ is applicable if and only if $t \in [T_0, T_1]$. In the case $(0, T_0)$ we have $f(0, T_0) = Q \geq A_{T_0}(0)$. Observe that, since the packet p arrives to $j = 0$ at time T_0 and the last bit of p completely leaves it at time $T_1 - P_0$, there is a continuous flow of bits sent across $j = 0$ during the interval $[T_0, T_1 - P_0]$. Then, from Lemmas 18 and 19, and the induction hypothesis,

$$\begin{aligned}
A_t(0) &\leq A_{T_0}(0) + (1 - \varepsilon)(t - T_0)B_0 + b - (t - P_0 - T_0)B_0 \\
&= A_{T_0}(0) - \varepsilon(t - T_0)B_0 + b + P_0B_0 \\
&\leq f(0, T_0) - \varepsilon(t - T_0)B_{min} + b + P_{max}B_{max} \\
&= f(0, t) - P_{max}B_{max} - L_{max} \\
&\leq f(0, t).
\end{aligned}$$

Now we want to prove the claim for all $j > 0$, for this we use induction on j . We assume by induction hypothesis that $A_t(j - 1) \leq f(j - 1, t)$ for all applicable $(j - 1, t)$. For T_0 we have that (j, T_0) is applicable, and we have $f(j, T_0) = Q + bj \geq Q \geq A_{T_0}(j)$.

Now, consider any applicable pair (j, t) , with $t > T_0$. If for all $t' \in [T_0, t - P_j)$ j 's output buffer is not empty at time t' , then by Lemmas 18 and 19 and the induction hypothesis, we have

$$\begin{aligned}
A_t(j) &\leq A_{T_0}(j) + (1 - \varepsilon)(t - T_0)B_j + b - (t - P_j - T_0)B_j \\
&= A_{T_0}(j) - \varepsilon(t - T_0)B_j + b + P_jB_j \\
&\leq f(j, T_0) - \varepsilon(t - T_0)B_{min} + b + P_{max}B_{max} \\
&= f(j, t) - (2P_{max}B_{max} + L_{max})j - (P_{max}B_{max} + L_{max}) \\
&\leq f(j, t).
\end{aligned}$$

Otherwise, there is some step $t' \in [T_0, t - P_j)$ in which j 's output buffer is empty. Let us take the largest such $t' < t - P_j$. Recall that; by definition, (j, t) is applicable when $t \in [T_0, T_{j+1}]$. Then, in this case $t' \in [T_0, T_{j+1} - P_j)$. Now by Lemma 20 we have that the pair $(j - 1, t')$ is also applicable. Since the output buffer of j is empty at time t' , then $A_{t'}(j) \leq A_{t'}(j - 1) + P_{j-1}B_{j-1} + L_{max}$. Again, applying Lemmas 18 and 19 and the induction hypothesis we have

$$\begin{aligned}
A_i(j) &\leq A_{t'}(j) + (1 - \varepsilon)(t - t')B_j + b - (t - P_j - t')B_j \\
&\leq A_{t'}(j - 1) + P_{j-1}B_{j-1} + L_{max} + b + P_jB_j - \varepsilon(t - t')B_j \\
&\leq f(j - 1, t') + b + 2P_{max}B_{max} + L_{max} - \varepsilon(t - t')B_{min} \\
&= f(j, t') - \varepsilon(t - t')B_{min} \\
&= f(j, t).
\end{aligned}$$

■

Applying the previous lemma, we obtain the following results:

Theorem 22 *Let G be a unidirectional ring, A an (r, b) -adversary with $r < 1$ and $b \geq 1$ and let P be a greedy policy, then the system (G, A, P) is stable, and there are never more than*

$$(1 - \varepsilon) \frac{(b + 2P_{max}B_{max} + L_{max})n}{\varepsilon B_{min}} B_{max} + b$$

bits in the system that require any given edge.

Proof: The second statement implies the first, so we will concentrate on proving the second statement. Set $Q' = ((1 - \varepsilon)(b + 2P_{max}B_{max} + L_{max})nB_{max}/\varepsilon B_{min}) + b$, and suppose that the theorem is not true. Let T' be the first time at which $Q' + 1$ bits in the system require any edge. Let $T_0 < T'$ denote the time at which the oldest of these bits, which belongs to a packet p , was injected. Relabel the nodes to make 0 the node at which p was injected. The link of interest will become $k \leq n - 1$. Since, at time T' packet p has not crossed edge k yet, we have that (k, T') is applicable. Note that for Q as defined in (4.2) we have $Q \leq Q'$. During interval $[T_0, T']$ at most $(T' - T_0)(1 - \varepsilon)B_{max} + b$ bits can be injected requiring any edge. Therefore, $Q' \leq (T' - T_0)(1 - \varepsilon)B_{max} + b$, and hence

$$\begin{aligned}
\varepsilon(T' - T_0)B_{min} &\geq \varepsilon \frac{Q' - b}{(1 - \varepsilon)B_{max}} B_{min} \\
&= \varepsilon \frac{(b + 2P_{max}B_{max} + L_{max})n}{\varepsilon B_{min}} B_{min} \\
&= (b + 2P_{max}B_{max} + L_{max})n.
\end{aligned}$$

By our assumption we have $Q' < A_{T'}(k)$. By Lemma 21, we get that $A_{T'}(k) \leq f(k, T')$.

Then, we obtain the following contradiction

$$\begin{aligned}
Q' &< f(k, T') \\
&= Q - \varepsilon(T' - T_0)B_{min} + (b + 2P_{max}B_{max} + L_{max})(1 + k) \\
&\leq Q' - \varepsilon(T' - T_0)B_{min} + (b + 2P_{max}B_{max} + L_{max})n \\
&\leq Q' - (b + P_{max}B_{max} + L_{max})n + (b + P_{max}B_{max} + L_{max})n \\
&= Q'.
\end{aligned}$$

■

Theorem 23 *The maximum number of steps a packet spends in the system is*

$$\frac{B_{max}((b + 2P_{max}B_{max} + L_{max})n)}{\varepsilon^2 B_{min}^2} + \frac{b}{\varepsilon B_{min}} + P_{max}.$$

Proof: Without loss of generality, Suppose that a packet p is injected at time T_0 at the origin 0 with destination $k + 1 \pmod{n}$. Then, k is the last edge it has to traverse. Suppose that p did not completely leave the output buffer of edge k at time T' , where

$$T' = T_0 + \frac{B_{max}((b + 2P_{max}B_{max} + L_{max})n)}{\varepsilon^2 B_{min}^2} + \frac{b}{\varepsilon B_{min}}.$$

Then by Theorem 22 we can apply Lemma 21 with

$$Q = (1 - \varepsilon) \frac{(b + 2P_{max}B_{max} + L_{max})n}{\varepsilon B_{min}} B_{max} + b,$$

and obtain the following contradiction

$$\begin{aligned}
A_{T'}(k) &\leq f(k, T') \\
&= Q - \varepsilon(T' - T_0)B_{min} + (b + 2P_{max}B_{max} + L_{max})(1 + k) \\
&\leq Q - \frac{B_{max}((b + 2P_{max}B_{max} + L_{max})n)}{\varepsilon B_{min}} - b + (b + 2P_{max}B_{max} + L_{max})n \\
&= (1 - \varepsilon) \frac{(b + 2P_{max}B_{max} + L_{max})n}{\varepsilon B_{min}} B_{max} + b - \\
&\quad \frac{B_{max}((b + 2P_{max}B_{max} + L_{max})n)}{\varepsilon B_{min}} - b + (b + 2P_{max}B_{max} + L_{max})n \\
&= - \frac{(b + 2P_{max}B_{max} + L_{max})n}{B_{min}} B_{max} + (b + 2P_{max}B_{max} + L_{max})n \\
&= ((b + 2P_{max}B_{max} + L_{max})n) \left(1 - \frac{B_{max}}{B_{min}}\right) \\
&\leq 0.
\end{aligned}$$

■

4.3 Characterization of universal stability

In this section we give a characterization of universal stability under the CAQT model. Our characterization relies on the the universal stability of rings and DAGs presented in [17]. In general, we proceed analogously as in the Chapter 3, and [7]. In the following we will consider bounded (r, b) -adversaries with $r = 1 - \varepsilon < 1$.

First, notice that Lemma 12 obtained in the AQT with setups setting holds also in the CAQT model.

Lemma 24 *If digraphs G_1 and G_2 are universally stable in the CAQT model, then so is any graph formed by joining them with edges that go only from G_1 to G_2 .*

Proof: Assume that we are working against $(1 - \varepsilon, b)$ -adversary. Since G_1 is universally stable, any bit which is injected in G_1 gets out of G_1 within T_1 times steps, where T_1 is some constant that depends on $1 - \varepsilon$ and b . Some of these bits may then enter G_2 . Now consider a time window T_2 units long. Any new bit that enters G_2

during this window must have been introduced during an interval of length at most $T_1 + T_2$. For a given edge, the number of bits injected during such interval can be at most $(T_1 + T_2)(1 - \varepsilon)B_{max} + b = T_2B_{max}(1 - \varepsilon) + T_1B_{max}(1 - \varepsilon) + b$. Thus we can say that these bits could have been injected by $(1 - \varepsilon, b')$ -adversary, where $b' = b + T_1B_{max}(1 - \varepsilon)$. By definition of universal stability, G_2 is stable against such adversary, therefore the traversal time for bits, and then for packets, as well as the queue lengths are bounded in G . ■

We can now conclude the following result:

Theorem 25 *A network G is universally stable in the CAQT model if and only if all its strongly connected components are universally stable.*

Observe that previous theorem, together with Lemma 24, guarantee the universal stability of unicyclic graphs, i.e., those digraphs with only one cycle and possibly, directed subgraphs rooted in the nodes of that unique cycle.

Once all these types of graphs (namely directed acyclic digraphs, rings and unicyclic digraphs) have been identified as universally stable, the next step towards the characterization of the property is to detect the simplest digraphs that are not universally stable in the CAQT model. Let U_1 and U_2 be as Figures 3.1 and 3.2 of Chapter 3. Observe that U_1 is the smallest non-unicyclic digraph whose cycles share an edge, and U_2 is the smallest non-unicyclic digraph whose cycles share a vertex.

Again using these two digraphs U_1 and U_2 , we want to consider the whole family of digraphs that contain the essence of their structure, i.e., those digraphs which are defined by iteratively extending their topologies.

Given a digraph G , we use the subdivision operations defined in [7] to build that family. We will denote as $\varepsilon(G)$ the family of digraphs formed by G and all the digraphs obtained from G by successive arc or 2-cycle subdivisions (subdivisions presented in Section 3.4).

In the AQT model, we know that neither the digraphs U_1, U_2 nor the families that they define by subdivision operations are universally stable. Even more, we know that

these digraphs characterize the property of universal stability of networks. These two results are stated in [7] and used in Section 3.4 for Theorems 14 and 15.

Since the instability results from the AQT model holds in the CAQT model, we have that all digraphs in $\varepsilon(U_1) \cup \varepsilon(U_2)$ are not universally stable under the CAQT model. Moreover, since the previously considered digraphs (namely acyclic digraphs, rings and unicyclic digraphs) are universally stable under the CAQT model, we get the same family of universally stable networks as in Theorem 15.

Corollary 26 *A digraph is universally stable in the CAQT model if and only if it does not contain as a subdigraphs any of the digraphs in $\varepsilon(U_1) \cup \varepsilon(U_2)$.*

Chapter 5

Non Synchronized Model

5.1 Introduction

Stability is a desirable property of a packet switched network in order to be able to guarantee some quality of service. It is well known that an appropriate scheduling of packets is fundamental in order to achieve stability [11]. A fundamental question is to identify scheduling policies that are able to guarantee stability, and among these, policies that guarantee good quality of service (e.g., latency).

Time-based scheduling policies. From all the policies that have been shown to be stable in all networks (which we call *universally stable*) under the AQT and CAQT models, those that seem to provide the lowest end-to-end packet delays are based on timing. In fact, it has been shown by Weinard [56] that, for any policy in the family of Without-Time-Stamping strategies, there are n -node under-loaded networks in which the delays and queue sizes are $2^{\Omega(\sqrt{n})}$. A policy of this family is assumed to know the network topology, and it assigns the priority of a packet as a function of its path and the number of edges it already crossed. This implies that, in general, it is convenient to use some timing information for scheduling. Unfortunately, the simplest time-based policies, can also suffer of large delays. For instance, for Shortest in System (SIS), the policy that gives the highest priority to the newest packet in the network, there are networks in which the delays and queue sizes are $2^{\Omega(\sqrt{n})}$ [11]. Similarly, for Longest in System (LIS), the policy that gives the highest priority to the oldest packet

in the network, there are networks with diameter d in which the delays and queue sizes are $2^{\Omega(d)}$ [14].

The good news are that time-based policies can in fact provide low delay guarantees. For instance, a randomized scheduling algorithm that guarantees delays polynomial on the network parameters is described in [11]. The deterministic scheduling algorithm with polynomial delays presented in [12] uses a similar approach. Additionally, there are simulation studies [53] which show that LIS may in fact behave much better in practice than one may expect from the lower bounds mentioned above.

The Non Synchronized CAQT model. The above mentioned results for time-based scheduling policies are obtained in network models in which it is implicitly assumed that each node in the network has a local clock to provide the time, and that all these clocks are synchronized and provide the same time. However, this latter assumption is not realistic in practice, since the oscillation frequency of each timer is different, and thus produces different *clock drifts*. A consequence of these drifts is that different clocks often provide different times. The differences between the clock times is what we call *clock skews*. In practice, in order to limit the effect of clock drifts, and to bound the clock skews, there are mechanisms like the Network Time Protocol (NTP), that allow the resynchronization of clocks.

In this chapter we propose a model of adversary in which clocks do not need to be synchronized. We call the new model *Non Synchronized CAQT* (NSCAQT), since it is basically the CAQT model with this additional generalization. Under this model we will study the behavior of different queue scheduling policies which depend on time and can be affected both by clock skews and clock drifts. To study these policies, we need to make assumptions on how they use the local clocks. For instance, for LIS and SIS we will assume that packets are assigned the local clock value of their injection node at their injection time, value that they carry with them and is used for scheduling.

In this chapter we will study two main variations of the NSCAQT model. In the first one we assume that the system has clock skews but no clock has drifts. Hence,

in this model clocks do not have the same time, but their time differences remain constant. We call this the *NSCAQT model with constant skews*. The second model we will study is a model in which skews can vary over time, but there is a bound on the maximum difference between the time of the clocks. We call this model the *NSCAQT model with bounded skews*, it is well adapted to model a network in which a protocol like NTP is used to periodically resynchronize all the clocks. A third natural model which we do not explore here is one in which clock drifts are present and no resynchronization mechanism guarantees that the skews are bounded. This model is left for future study.

Structure. The structure of the rest of the chapter is the following. In Section 5.2 we define the NSCAQT model in detail. In Section 5.3 we study stability under the NSCAQT model with constant skews. In Section 5.4 we introduce and study two families of policies under the NSCAQT model with bounded skews. In Section 5.5 we explore the performance of LIQ under both NSCAQT models.

5.2 System model

Like most previous adversarial network models, the NSCAQT system model has three major elements: an underlying network G , a scheduling policy P , and an adversary A . With these elements, the evolution of the system can be seen as a game between the adversary, which injects packets in the network trying to create instability, and the scheduling policy, that decides which packets move along their paths in the network, trying to prevent instability. The model of the systems considered in this chapter is a direct extension of that presented in Chapter 4.

Clocks. As said before, the main difference between the NSCAQT model we propose and previous models [18, 20, 11, 17] is that we consider here the impact on the performance of clocks not being synchronized. In order to make the model as general as possible, we assume that the output queue of each edge has its own internal clock, called *edge clock* (this is clearly more general than assuming one clock per node).

Additionally, we assume there is an external reference clock which is always on time. We refer to this clock as the *real clock* and we say that it provides the *real time*. We assume that the adversary has access to both the edge clocks and the real clock, while the scheduling policy at a given edge has only access to the clock of that edge.

The difference between the real clock and the edge clock of e at real time t is what we call the *clock skew* of e 's edge clock at time t , and is denoted by $\phi_e(t)$. Then, if t_e denotes the value of the edge clock of e at real time t , we have $t_e = t - \phi_e(t)$. If this value changes over time, we say that the edge clock has a *drift*. If an edge clock has no drift we omit the time and denote its skew by ϕ_e . Note that, at any given time, the skew of an edge clock can be positive or negative. However, for convenience we assume that these skews are all non-negative if all edge-clock skews are lower bounded. We can do this freely since the real clock is not available to the scheduling policies and does not interfere in the relation between edge clocks.

We denote by $T_i(p)$, $0 \leq i < d_p$, the real time at which a packet p arrives to the output queue of the edge $e_i(p)$. Due to clock skews, according to edge $e_i(p)$'s clock, the instant when packet p arrives to the output queue is $T_i(p) - \phi_{e_i(p)}(T_i(p))$. Additionally, we denote by $T_{d_p}(p)$ the time at which p is completely received at its destination and leaves the system.

Scheduling policies. As we said above, the scheduling policy is in charge of deciding, whenever a link e is available, which packet from those in the output queue of e must be sent next across e . We only consider distributed work-conserving time-based scheduling policies, which are policies that use the edge clocks for scheduling. Note that policies that are not time-based are not affected by clock skews and drifts.

Two of the most studied distributed work-conserving time-based scheduling policies are Longest in System (LIS) and Shortest in System (SIS). The LIS policy gives the highest priority to the packet that has been in the system for the longest time, while the SIS policy gives the highest priority to the packet that has been in the system for the shortest time. These definitions do not clearly show the use these policies make of the edge clocks. For that, we need to look at the natural implementation of

these policies: upon arrival of a packet p into the system, it is assigned a time-stamp, $TS(p)$, which p carries with it. Then, the LIS and SIS policies only compare the time-stamps of the packets to decide which one to schedule next. The edge scheduler in LIS gives the highest priority to the packet with the smallest time-stamp, while in SIS it gives the highest priority to the packet with the largest time-stamp. Note that when clocks are not synchronized, these time-stamps are not accurate, since the time-stamp for a packet p is $TS(p) = T_0(p) - \phi_{e_0(p)}(T_0(p))$. These two policies have been proved to be universally stable in [17] for the CAQT model, where $\phi_e(t) = 0$ for all $e \in E(G)$ and for all $t \geq 0$.

In addition to LIS and SIS, we will study two families of policies derived from SIS and LIS, that we call *Shortest in System considering Path* (SISP) and *Longest in System considering Path* (LISP), respectively. To implement policies belonging to these families, packets carry their time-stamp, and (when needed) the packet path and the number of links traversed, so that at its edge $e_i(p)$, packet p is assigned a priority label of the form

$$PL(p, i) = TS(p) + f(\Pi(p), i),$$

where $\Pi(p) = (e_0(p), e_1(p), \dots, e_{d_p-1}(p))$ denotes the path the packet has to traverse, and $f(\Pi(p), i)$ is a function which assigns a real number to each pair formed by a packet path $\Pi(p)$ and a value $i \in \{0, 1, \dots, d_p - 1\}$, being i the number of crossed edges by the packet in its path. A policy \mathcal{P}_f is in SISP (respectively LISP) if at each queue it gives the highest priority to the packet p with the largest (respectively smallest) value $PL(p, i)$. Notice that when $f(\Pi(p), i) = 0$ for all $\Pi(p)$ and i , \mathcal{P}_f is equivalent to SIS (respectively LIS). Since the function f is defined over the finite sets of paths and number of edges crossed by a packet, it has a maximum and a minimum value, that we denote by f_{\max} and f_{\min} , respectively.

Finally, we will consider a new policy named *Longest in Queues* (LIQ). In this policy the highest priority is assigned to the packet that has been waiting the longest in all the output queues it has visited. In our model NSCAQT (where clocks are not synchronized) we assume that the time in queues is measured locally at each output queue. The time a packet p waits at an edge's queue is the difference between the

value of the edge clock when the packet arrives and the value when it starts being transmitted, or the current value of the edge clock if it is still waiting. The time used to schedule p is the sum of these waiting times in all the visited queues.

System stability. To study stability and performance of packet switching networks, we use the concept of a (G, P, A) system. Here again we say that a system (G, P, A) is stable if the maximum number of packets (or bits) present in the system is bounded at any time by a constant that may depend on system parameters: the network, the adversary or the policy. A policy P is *universally stable* if the system (G, P, A) is stable on every network G and against every (r, b) -adversary A with $r < 1$ and $b \geq 1$.

5.3 Stability of policies for constant clock skews

In this section we study the case in which all the edge clocks have zero drift, so that ϕ_e is constant. This framework allows to assure the stability in a non synchronized system if there is stability in a synchronized system for many policies, in particular for those policies that only depend on the injection time and the remaining path of the packets.

We present a proof by transformation of this case. We start from a (G, P, A) system with non synchronized clocks, where A is an (r, b) -adversary, $r \leq 1$. Then, we vary the network G and the adversary A to obtain a synchronized system (G', P, A') , where A' is an (r, b') -adversary, so that if (G', P, A') is stable, then (G, P, A) is also stable. First we explain the constructions and then we state the result.

Construction of G' . As we explained in the previous section, since we arbitrarily fix the real clock, we can do it so that all the clock skews are non-negative. Then, let $G = (V, E)$ be a directed graph in the NSCAQT model with constant skew $\phi_e \geq 0$ for each edge clock.

We construct G' starting from G as shown in Figure 5.1. For each edge $e \in E(G)$ with $\phi_e > 0$, let $K_e = \lceil b + rB_e\phi_e/2 \rceil$. Then, we add $L_{\max} \times K_e$ edges and $L_{\max} \times K_e$ nodes. New edges and nodes are denoted $e^{l,j}$ and $v^{l,j}$, respectively, where

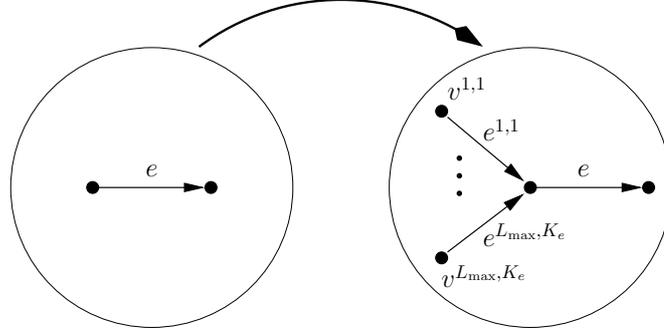


Figure 5.1. Basic process to transform G into G' .

$l \in \{1, 2, \dots, L_{\max}\}$ and $j \in \{1, 2, \dots, K_e\}$. For all l and for all j , we add edge $e^{l,j}$ going from node $v^{l,j}$ to the tail node of edge e . For every edge $e^{l,j}$, we set the bandwidth to $B_{e^{l,j}} = \frac{2l}{\phi_e}$ and the propagation delay to $P_{e^{l,j}} = \frac{\phi_e}{2}$.

By construction of G' , a packet p of size L_p takes $\frac{\phi_e}{2}$ units of time to be fully sent and ϕ_e units of time to be fully received across edge $e^{L_p,j}$, $j \in \{1, 2, \dots, K_e\}$.

Construction of A' . We now construct the adversary A' from A . A packet that is injected by A at edge e of G with $\phi_e = 0$ is injected by A' exactly in the same edge and at the same time in G' . Now, let p be a packet of size L_p that was injected in G by A at time t , such that $\phi_{e_0(p)} > 0$. Then, A' will inject a packet p' in G' at time $t - \phi_{e_0(p)}$. The size of p' will be $L_{p'} = L_p$ and its path will be

$$\Pi(p') = (e_0^{L_p,j}(p), e_0(p), e_1(p), \dots, e_{d_p}(p)),$$

where $e_0^{L_p,j}(p)$ is an edge corresponding to the construction described above. This edge must satisfy that no other packet has been injected in it in the previous $\phi_e/2$ time. Since K_e is clearly an upper bound on the number of packets of length L_p that can be injected in any interval of $\phi_e/2$ time, and there are K_e edges for each L_p , there is always a suitable edge to be used.

Under these circumstances, if a packet is injected by A in time t in the queue of edge e , and labeled with a time-stamp of $t - \phi_e$, a corresponding packet injected by A' will arrive to the same queue at the same time t and labeled with the same time-stamp. Let us now bound the parameters of the adversary A' .

Note that, the packets injected by A during an interval of size $|I| + \phi_{\max}$ are injected by A' during an interval I' with maximum size $|I|$. So, since A is an (r, b) -adversary, A' is an (r, b') -adversary, where $b' = b + \phi_{\max}B_{\max}$, and $\phi_{\max} = \max_{e \in E(G)} \{\phi_e\}$.

We can now state the main result of this section.

Theorem 27 *Let (G', P, A') be the synchronized system in the CAQT model obtained from the non synchronized system (G, P, A) through the above process. Let P be a scheduling policy which considers only the time of injection of the packets and the paths that the packets still have to traverse. Then, (G, P, A) is stable if and only if (G', P, A') is stable.*

Proof: Note that the stability of (G, P, A) must occur for any value of the clock skews, including the case in which all skews are zero. Then, it trivially follows that if (G, P, A) is stable, (G', P, A') is stable as well.

In the other direction, we first have that the queues of the new edges $e^{l,j}$ in system (G', P, A') never present contention since, as already mentioned, and by the construction of the adversary A' , by the time a packet arrives to the network the previous packet (if any) was already sent. Then, if we observe the output queues of the edges that G' and G have in common, we find that similar sets of packets arrive at the same times, with the same time-stamps, and with the same remaining paths to cross in both systems (G, P, A) and (G', P, A') . Since these are the parameters used by P to schedule the packets, we have that the behavior of these queues in systems (G, P, A) and (G', P, A') is exactly the same. Hence, we have that if (G, P, A) is unstable, then (G', P, A') is also unstable or, equivalently, if (G', P, A') is stable, then (G, P, A) is stable. ■

Corollary 28 *The scheduling policies that are universally stable in CAQT and only consider the times of injection and the paths that the packets still have to traverse are universally stable in the NSCAQT model with constant clock skews.*

5.4 Stability of policies for bounded clock skews

In this section we will study the case in which clocks may experience drifts. Hence, we assume here that the clock skews are not necessarily constant. However, the maximum difference between real time and any edge clock is bounded. As we said, this model fits naturally with a system in which edge clocks are periodically resynchronized, for instance via NTP.

Again, we will adapt the real time reference clock, in order to simplify the analysis and the presentation. Like in the previous section, we will assume that all clock skews are non-negative, i.e., for any edge e and any time t , $\phi_e(t) \geq 0$. Additionally, since we assume that skews are bounded, we can safely define $\phi_{\max} = \max_{e,t} \{\phi_e(t)\}$.

5.4.1 Universal stability of SISP

In this subsection we explore the stability of the new family of policies SISP defined in Section 5.2, which is based on the injection time, and the path and number of edges already crossed by a packet. As described there, a policy \mathcal{P}_f in SISP assigns to each packet p at the queue of its edge $e_i(p)$ a priority label $PL(p, i) = TS(p) + f(\Pi(p), i)$, and gives the highest priority to the packet with the largest label.

Now we prove that every policy in SISP is universally stable in the NSCAQT model when the clock skew is bounded by ϕ_{\max} . We start with the following simple lemma.

Lemma 29 *Let p and q be two packets. If $T_0(q) > T_0(p) + f_{\max} - f_{\min} + \phi_{\max}$, then p never has higher priority than q in any queue.*

Proof: Let us assume that p and q meet at the output queue of edge $e_i(p) = e_j(q)$. Note that $\phi_{\max} - \phi_{e_0(q)} \geq 0$ and that $f(\Pi(q), j) - f_{\min} \geq 0$. Hence,

$$\begin{aligned} PL(q, j) &= T_0(q) - \phi_{e_0(q)} + f(\Pi(q), j) \\ &> T_0(p) + f_{\max} - f_{\min} + \phi_{\max} - \phi_{e_0(q)} + f(\Pi(q), j) \\ &\geq T_0(p) + f_{\max} \geq PL(p, i), \end{aligned}$$

and then $PL(q, j) > PL(p, i)$ for all i and j . ■

The proof of universal stability of the policies in SISP we present is very similar to that of SIS presented in [17] for the CAQT model. We first recall a lemma proved there, which limits the time spent by a packet in the queue of an edge e if there are $k - 1$ bits in the system with higher priority to cross that edge. The assumptions and the proof of this lemma do not depend on whether the clocks are synchronized. Hence, it can be applied directly to our model.

Lemma 30 ([17]) *Let p be a packet that, at real time t , is waiting in the queue of edge e . At that instant, let $k - 1$ be the total size in bits of the packets in the system that also want to cross e and that may have priority over p . Then, p will start crossing e in at most $(k + b)/(\varepsilon B_e)$ units of time.*

Recall that, when a packet p starts crossing an edge e , it spends $P_e + L_p/B_e \leq D_{\max}$ units of time until it crosses it completely. Then, using this and the previous lemma recursively we can prove the following result.

Lemma 31 *Let*

$$k_0 = r(\phi_{\max} + f_{\max} - f_{\min})B_{\max} + b,$$

and

$$k_i = k_{i-1} + r \left(\frac{k_{i-1} + b}{\varepsilon B_{\min}} + D_{\max} \right) B_{\max} + b,$$

for $0 < i < d_{\max}$. *When a packet p arrives to the output queue of edge $e_i(p)$, no more than $k_i - 1$ bits can have priority over it in any edge $e_j(p)$, for $j \geq i$.*

Proof: Let us first consider the case $i = 0$. When a packet p arrives into the system, by Lemma 29, the packets that may have priority over it have been injected at most $\phi_{\max} + f_{\max} - f_{\min}$ units of time earlier than p , because, although they may have arrived to the system before p , they have a greater timestamp due to their initial edge's clock skew or the value of the function f . The total size of these packets is at most $r(\phi_{\max} + f_{\max} - f_{\min})B_{\max} + b - 1 = k_0 - 1$ bits (since $L_p \geq 1$).

Let us now assume as induction hypothesis that the claim holds for $0 \leq i < d_p - 1$. Then, from Lemma 30, p will arrive at the output queue of edge $e_{i+1}(p)$ at most $(k_i + b)/(\varepsilon B_{\min}) + D_{\max}$ time units after arriving at the output queue of edge $e_i(p)$. The packets that can block p at any edge injected during this time have at most $r((k_i + b)/(\varepsilon B_{\min}) + D_{\max})B_{\max} + b$ bits. Hence packets with at most $(k_i - 1) + r((k_i + b)/(\varepsilon B_{\min}) + D_{\max})B_{\max} + b = k_{i+1} - 1$ bits can block p in any edge $e_j(p)$, $j \geq i + 1$. ■

Using these definitions of k_i and the previous lemma, we can limit the size of the queues and the amount of time that a packet spends in the network as it was done in [17]. The proof of the theorem is verbatim to the the final part of the corresponding theorem in [17] and is hence omitted.

Theorem 32 *Let G be a network and d_{\max} the length of its longest edge-simple directed path, let A be an (r, b) -adversary with $r = 1 - \varepsilon < 1$ and $b \geq 1$, and let \mathcal{P}_f be a policy in SISP. Then the system (G, \mathcal{P}_f, A) is stable under the NSCAQT model with bounded clock skews, no queue ever contains $k_{d_{\max}-1} + L_{\max}$ bits, and no packet spends more than*

$$\frac{d_{\max}b + \sum_{i=0}^{d_{\max}-1} k_i}{\varepsilon B_{\min}} + d_{\max}D_{\max}$$

units of time in the system.

The main result of this section follows.

Corollary 33 *Any protocol in SISP, and in particular SIS, is universally stable under the NSCAQT model with bounded clock skew, and hence under the CAQT model.*

5.4.2 Universal stability of LISP

As defined in Section 5.2, LISP is a family of policies based on the injection time and path, and the number of edges already crossed by a packet. A policy \mathcal{P}_f in LISP assigns to each packet p at the queue of its edge $e_i(p)$ a priority label $PL(p, i) = TS(p) + f(\Pi(p), i)$, and gives the highest priority to the packet with the smallest label.

Now we prove that every policy in LISP is universally stable in the NSCAQT model when the clock skew is bounded by ϕ_{\max} . We start with the following simple lemma, which is similar to Lemma 29 in the previous section.

Lemma 34 *Let p and q be two packets. If $T_0(q) > T_0(p) + f_{\max} - f_{\min} + \phi_{\max}$, then q never has higher priority than p in any queue.*

Proof: Let us assume that p and q meet at the output queue of edge $e_i(p) = e_j(q)$. Note that $\phi_{\max} - \phi_{e_0(q)} \geq 0$ and that $f(j) - f_{\min} \geq 0$. Hence,

$$\begin{aligned} PL(q, j) &= T_0(q) - \phi_{e_0(q)} + f(j) \\ &> T_0(p) + f_{\max} - f_{\min} + \phi_{\max} - \phi_{e_0(q)} + f(j) \\ &\geq T_0(p) + f_{\max} \geq PL(p, i), \end{aligned}$$

and then $PL(q, j) > PL(p, i)$ for all i and j . ■

Let p be a packet in the system and let t denote some real time in $[T_0(p), T_{d_p}(p)]$. We denote by $g(t)$ the real injection time of the oldest packet in the system at time t . We define $C_p = \max_{t \in [T_0(p), T_{d_p}(p)]} \{t - g(t)\}$. Notice that C_p represents the age of the oldest packet in the system while p is present. For convenience we use the abbreviation $K = f_{\max} - f_{\min} + \phi_{\max}$.

In the following lemma we start by bounding the amount of time, $T_{i+1}(p) - T_i(p)$, that p takes to move from the queue of e_i to the queue of e_{i+1} , which allows us to bound the time p is in the system.

Lemma 35 *The time packet p is in the system is at most*

$$T_{d_p}(p) - T_0(p) \leq (1 - \varepsilon^d) \left(K + C_p + \frac{D_{\max}}{1 - \varepsilon} \right).$$

Proof: Observe that the oldest packet in the system when p arrives at the queue of e_i at time $T_i(p)$ was injected at most at time $T_i(p) - C_p$. Then, from Lemma 34 we have that p , and all the packets with higher priority than p in the queue of e_i , were injected

during the interval $[T_i(p) - C_p, T_0(p) + K]$. Since the packets injected in this interval can have at most

$$(1 - \varepsilon)(T_0(p) + K - T_i(p) + C_p)B_{e_i} + b$$

bits, we know that all of them cross e_i in at most

$$\begin{aligned} & (1 - \varepsilon)(T_0(p) + K - T_i(p) + C_p) + \frac{b}{B_{e_i}} + P_{e_i} \\ & \leq (1 - \varepsilon)(T_0(p) + K - T_i(p) + C_p) + D_{\max} \end{aligned}$$

units of time. Then, we have that

$$\begin{aligned} T_{i+1}(p) & \leq T_i(p) + (1 - \varepsilon)(T_0(p) + K - T_i(p) + C_p) + D_{\max} \\ & = \varepsilon T_i(p) + (1 - \varepsilon)(T_0(p) + K + C_p) + D_{\max}, \end{aligned}$$

and solving the recurrence for $T_{d_p}(p)$ we get

$$T_{d_p}(p) - T_0(p) \leq (1 - \varepsilon^{d_p})(K + C_p + D_{\max}/(1 - \varepsilon)).$$

■

Now, we have bounded the time that a packet spends in the system, but our bound depends on C_p . To finish the proof we need the following lemma.

Lemma 36 *For any packet p we have that*

$$C_p \leq \frac{1 - \varepsilon^{d_{\max}}}{\varepsilon^{d_{\max}}} \left(K + \frac{D_{\max}}{1 - \varepsilon} \right) = C.$$

Proof: By contradiction, let us assume there are packets that spend in the system more than C time. Let t be the first time at which a packet has been in the system more than C time, and let p be one packet that satisfies this. Then $T_{d_p}(p) - T_0(p) \geq t - T_0(p) > C$. Note that before t no packet in the system has age older than C . Hence, by Lemma 35 with $C_p = C$ we have that

$$\begin{aligned} T_{d_p}(p) - T_0(p) & \leq (1 - \varepsilon^{d_p}) \left(K + C + \frac{D_{\max}}{1 - \varepsilon} \right) \leq (1 - \varepsilon^{d_{\max}}) \left(K + C + \frac{D_{\max}}{1 - \varepsilon} \right) \\ & = C - \varepsilon^{d_{\max}} C + (1 - \varepsilon^{d_{\max}}) \left(K + \frac{D_{\max}}{1 - \varepsilon} \right) = C, \end{aligned}$$

which is a contradiction. ■

Now we can enunciate the final theorem of this section. The proof follows from Lemmas 35 and 36, and the definition of K .

Theorem 37 *Let G be a network and d_{\max} the length of its longest edge-simple directed path, let A be an (r, b) -adversary, with $r = 1 - \varepsilon < 1$ and $b \geq 1$, and let \mathcal{P}_f be a policy in LISP. Then the system (G, \mathcal{P}_f, A) is stable under the NSCAQT model with bounded clock skew, and no packet spends more than*

$$\frac{1 - \varepsilon^{d_{\max}}}{\varepsilon^{d_{\max}}} \left(f_{\max} - f_{\min} + \phi_{\max} + \frac{D_{\max}}{1 - \varepsilon} \right)$$

units of time in the system.

Corollary 38 *Any protocol in LISP, and in particular LIS, is universally stable under the NSCAQT model with bounded clock skew, and hence under the CAQT model.*

5.5 Universal stability of LIQ

In previous sections we have shown how several policies are universally stable in the NSCAQT model with constant and bounded clock skews, respectively. Unfortunately, the bounds on end-to-end packet latencies we derived were dependent on the maximum clock skew that can occur in the system. This means that in a system with high maximum skew, the latencies can be very high.

In this section we study a new policy named LIQ, which gives the highest priority to the packet that has been waiting in output queues for the longest time. We prove that LIQ is universally stable in the NSCAQT model with bounded clock skews. The bad news is that in this case the end-to-end latency bound we obtain depends also on the maximum skew. The good news is that for the NSCAQT model with constant clock skews LIQ is universally stable, and the bound does not depend on the maximum skew, and it is similar to that obtained with LIS in a synchronized system.

As in previous sections, we assume that $\phi_e(t) \geq 0$ for all e and t . Then, we define $\phi_{\min}(e) = \min_t \{\phi_e(t)\}$ and $\phi_{\max}(e) = \max_t \{\phi_e(t)\}$. Finally, let $\Delta\phi = \max_e \{\phi_{\max}(e) - \phi_{\min}(e)\}$. Observe that in the model of constant skews, $\Delta\phi = 0$.

Let $W_{i,t}(p)$ be the amount of real time which packet p has waited in output queues when at time t it is at the queue of edge $e_i(p)$. We have that

$$t = T_0(p) + W_{i,t}(p) + \sum_{k=0}^{i-1} \left(\frac{L_p}{B_{e_k(p)}} + P_{e_k(p)} \right)$$

and, hence,

$$t - T_0(p) - d_{\max} D_{\max} \leq W_{i,t}(p) \leq t - T_0(p). \quad (5.1)$$

Recall that measuring the waiting time of a packet at a queue is done by taking the local time when the packet arrives and the local time when it leaves (or the current local time if it is still in the queue). Then the measured time at one queue can have an error of up to $\pm\Delta\phi$. Hence, the measured waiting time of a packet p that at time t is in queue $e_i(p)$, denoted $M_{i,t}(p)$, is a value in the interval $[W_{i,t}(p) - i\Delta\phi, W_{i,t}(p) + i\Delta\phi]$.

The proof we have for LIQ is similar to the proof for the LISP family of policies. We first prove a lemma analogous to Lemma 34.

Lemma 39 *Let p and q be two packets. If $T_0(q) > T_0(p) + d_{\max}(2\Delta\phi + D_{\max})$, then q never has higher priority than p in any queue.*

Proof: Let us assume that p and q meet at the output queue of edge $e_i(p) = e_j(q)$ at time t . We need to show that $M_{j,t}(q) < M_{i,t}(p)$ is satisfied for any i, j , and t . The largest value $M_{j,t}(q)$ can take is

$$M_{j,t}(q) \leq W_{j,t}(q) + j\Delta\phi \leq W_{j,t}(q) + d_{\max}\Delta\phi.$$

Similarly, $M_{i,t}(p) \geq W_{i,t}(p) - d_{\max}\Delta\phi$, and therefore we are left with the problem of showing that $W_{j,t}(q) < W_{i,t}(p) - 2d_{\max}\Delta\phi$. By using Equation (5.1) and the assumption of the lemma, we have

$$\begin{aligned} W_{j,t}(q) &\leq t - T_0(q) \\ &< t - T_0(p) - d_{\max}(2\Delta\phi + D_{\max}) \\ &\leq W_{i,t}(p) - 2d_{\max}\Delta\phi \end{aligned}$$

which completes the proof. ■

Now, defining $K = d_{\max}(2\Delta\phi + D_{\max})$, we have that Lemmas 35 and 36 are also valid for LIQ. Then, we can enunciate the following theorem.

Theorem 40 *Let G be a network with d_{\max} the length of its longest edge-simple directed path, let A be an (r, b) -adversary with $r = 1 - \varepsilon < 1$. Then, (1) the system (G, LIQ, A) is stable under the NSCAQT model with bounded clock skews, and no packet spends more than*

$$\frac{1 - \varepsilon^{d_{\max}}}{\varepsilon^{d_{\max}}} \left(d_{\max}(2\Delta\phi + D_{\max}) + \frac{D_{\max}}{1 - \varepsilon} \right)$$

units of time in the system, and (2) the system (G, LIQ, A) is stable under the NSCAQT model with constant clock skews, and no packet spends more than

$$\frac{1 - \varepsilon^{d_{\max}}}{\varepsilon^{d_{\max}}} \left(d_{\max}D_{\max} + \frac{D_{\max}}{1 - \varepsilon} \right)$$

units of time in the system.

Corollary 41 *LIQ is universally stable under the NSCAQT model with bounded clock skews, and hence under the CAQT model.*

Chapter 6

Conclusions and Open Problems

The main contribution of this thesis is the definition of three new adversarial models. Each one is specially well adjusted to analyze different kinds of network problems. Mainly, these models attempt to represent real communication and production networks, and more precisely attempt to capture issues concerning the queues generated in them.

The first model, presented in Chapter 2, is a new approach to model classical queueing systems. The other two of these are extensions of the well known AQT model (described in Chapter 3), and the CAQT model (described in Chapter 5). Additionally, a direct contribution to CAQT is done in Chapter 4.

The model presented in Chapter 2 gives a framework to study queueing systems under an adversarial setting. The main contribution of this model is the way in which a customer is represented. A customer is a set of processes whose sequence is determined by boolean functions, which allow (or not) a process to be serviced. Each process has an associated function, and these functions depend on the state of all the processes of the same customer. In this way, a customer is divided in several parts where each part represents components, or states depending on what a customer is. In this model, a sufficient condition for stability is proved. This condition reduces to the property of feed-forward routing in classical queueing systems. Hence, stability can be established under full rate injection in an adversarial setting for multiclass queueing networks with feed-forward routing.

The AQT with setups model is presented in Chapter 3. It models production networks (or packets in communication networks), with different types of customers, and rescues the fact that, when a server switches between types of customers, it must pay a switching (setup) cost, i.e., it must spend time without serving customers. We prove that dynamic multi-type routing networks with setups are “equivalent” to the so-called token routing networks. In order to establish network stability, sensible policies are used, and the set of universal stable networks using greedy sensible policies is characterized. Furthermore, it is shown that for every $0 < r < 1$ and $\Delta \geq 0$ and every shift-oblivious greedy sensible policy, it is always possible to construct a network and a bounded adversary that makes the network system unstable. Finally, we provide fluid type models and arguments through which stability of special network systems can be established.

A model to analyze communication networks in which router clocks are not necessarily synchronized is presented in Chapter 5. This new model (named NSCAQT) is an extension of the CAQT model. It is shown that, if all clocks run at the same speed, all universally stable policies in CAQT that only depend on the injection time and the remaining path to schedule packets remain universally stable. Then, two universal stable families are defined. In these families, scheduling depends on the injection time and the characteristics of the path that a packet must traverse. Particular instances of these families are the well known policies LIS and SIS. The universal stability of these policies is shown when time differences vary but remain bounded over time. Finally, a new policy is presented which gives priority to the packet that has been waiting the longest in edge queues (longest in queue LIQ). We show that LIQ is universally stable and, if clocks maintain constant differences, the delay and queue size bounds do not depend on them.

Finally, in terms of the contribution to the CAQT model. Universal stability for the simplest cyclic network (unidirectional ring) is proved. Using this result, the family of universal stable networks is characterized.

Some interesting problems that this thesis leaves open are the following:

- In the case of Boolean Queueing Systems, the complete characterization of

stable systems would increase the interest of the model. It would be interesting to find the set of boolean functions that customers may use in order to guarantee stability.

- In the AQT with setups model, an important question to answer is to find conditions for which there is a canonical adaptation of a packet scheduling policy P and a way of choosing a switching policy S such that a stable network G remains stable when subject to an injection pattern under the control of an (r, b) -adversary A , but in the multi-class scenario where setup costs equal Δ .
- In the case of NSCAQT, we would like to find policies with good behavior in the case in which skews are not bounded, hence clocks may provide arbitrarily different times.
- In case of the CAQT model, we would like to know how stability of networks and policies depends on the bandwidth configuration (bandwidth of every edge in the network). Given a system with stability threshold¹ equal to r . Under which conditions there exists a new bandwidth configuration for the network, such that the new system has stability threshold equal to $r' > r$?
- In general, for any adversarial model, an interesting challenge is to find deterministic policies that guarantee polynomial queue size against any (r, b) -adversary, where the polynomial is on the parameters of the system.
- To find a unstable policy for r close to 1, but stable for constant r is an open problem for AQT.
- FIFO, for obvious reasons, is the most studied scheduling policy. It would be interesting to experimentally evaluate the behavior of other policies such as LIS or SIS and compare it against with that of FIFO, in terms of throughput, queue sizes, dropped packets.

¹Stability threshold of a system is equal to r , if the system is stable for all adversarial injection rate $r' < r$ and unstable for all adversarial injection rate $r'' \geq r$.

- It is an interesting challenge to propose a bounded buffer model susceptible of being analyzed mathematically. Bounded buffers are realistic assumptions, and must be dealt with in practice [3].
- The adversarial model is based on the restricted adversary, but in practice it is difficult to decide whether this restriction is satisfied in a real network. An interesting problem is to develop packet admission control mechanisms for real systems in order to ensure that this restriction is satisfied.
- Generalize the CAQT model to dynamic settings, for instance when bandwidth change over time, is an interesting challenge.

To conclude we would like to stress the fact further elicited by this work that adversarial models provide flexible settings in which to address different types of problems that naturally arise queueing networks. These models' appeal and the great interest they have attracted is partly explained by the fine balance between the predictability of traditional injection processes (as in classical queueing theory) and on the other hand, unconstrained adversaries which can easily overload the system (as in competitive analysis). We believe that the application of similar adversarial type approaches might turn out productive in order to address issues that arise when dealing with other classical algorithmic problems.

Bibliography

- [1] ADLER, M., AND ROSEN, A. Tight bounds for the performance of longest-in-system on DAGs. *Journal of Algorithms* 55, 2 (2005), 101–112.
- [2] AIELLO, W., KUSHILEVITZ, E., OSTROVSKY, R., AND ROSEN, A. Adaptive packet routing for bursty adversarial traffic. *Journal of Computer and System Sciences* 60, 3 (2000), 482–509.
- [3] AIELLO, W., OSTROVSKY, R., KUSHILEVITZ, E., AND ROSÉN, A. Dynamic routing on networks with fixed-size buffers. In *SODA* (2003), pp. 771–780.
- [4] ÀLVAREZ, C., BLESÁ, M., DÍAZ, J., FERNÁNDEZ, A., AND SERNA, M. The complexity of deciding stability under ffs in the adversarial queueing model. *Inf. Process. Lett.* 90, 5 (2004), 261–266.
- [5] ÀLVAREZ, C., BLESÁ, M., DÍAZ, J., FERNÁNDEZ, A., AND SERNA, M. Adversarial models for priority-based networks. *Networks* 45 (2005), 23–35.
- [6] ÀLVAREZ, C., BLESÁ, M., AND SERNA, M. Universal stability of undirected graphs in the adversarial queueing model. In *SPAA* (2002), pp. 183–197.
- [7] ÀLVAREZ, C., BLESÁ, M., AND SERNA, M. A characterization of universal stability in the adversarial queueing model. *SIAM Journal on Computing* 34 (2004), 41–66.
- [8] ÀLVAREZ, C., BLESÁ, M., AND SERNA, M. The impact of failure management on the stability of communication networks. In *10th International Conference on Parallel and Distributed Systems* (2004), IEEE Computer Society Press, pp. 153–160.
- [9] ANDREWS, M. Instability of FIFO in session-oriented networks. *Journal of Algorithms* 50, 2 (2004), 232–245.
- [10] ANDREWS, M. Instability of FIFO in the permanent sessions model at arbitrarily small network loads. In *SODA* (2007), ACM Press.
- [11] ANDREWS, M., AWERBUCH, B., FERNÁNDEZ, A., KLEINBERG, J., LEIGHTON, T., AND LIU, Z. Universal stability results and performance bounds for greedy contention resolution protocols. *Journal of the ACM* (2001).

- [12] ANDREWS, M., FERNÁNDEZ, A., GOEL, A., AND ZHANG, L. Source routing and scheduling in packet networks. *J. ACM* 52, 4 (2005), 582–601.
- [13] ANDREWS, M., FERNÁNDEZ, A., HARCHOL-BALTER, M., LEIGHTON, T., AND ZHANG, L. General dynamic routing with per-packet delay guarantees of $o(\text{distance} + 1/\text{session rate})$. *SIAM Journal on Computing* 30, 5 (2000), 1594–1623.
- [14] ANDREWS, M., AND ZHANG, L. The effects of temporary sessions on network performance. *SIAM Journal on Computing* 33, 3 (2004), 659–673.
- [15] AWERBUCH, B., BERENBRINK, P., BRINKMANN, A., AND SCHEIDELER, C. Simple routing strategies for adversarial systems. In *Proc. of the 42nd IEEE Symposium on Foundations of Computer Science* (2001), pp. 158–167.
- [16] BHATTACHARJEE, R., GOEL, A., AND LOTKER, Z. Instability of FIFO at arbitrarily low rates in the adversarial queueing model. *SIAM Journal on Computing* 34, 2 (2004), 318–332.
- [17] BLESÁ, M., CALZADA, D., FERNÁNDEZ, A., LÓPEZ, L., MARTÍNEZ, A., SANTOS, A., SERNA, M., AND THRIVES, C. Adversarial queueing model for continuous network dynamics. *Theory of Computing Systems* (2007), in press.
- [18] BORODIN, A., KLEINBERG, J., RAGHAVAN, P., SUDAN, M., AND WILLIAMSON, D. Adversarial queueing theory. *Journal of the ACM* (1996).
- [19] BORODIN, A., OSTROVSKY, R., AND RABANI, Y. Stability preserving transformations: Packet routing networks with edge capacities and speeds. *Journal of Interconnection Networks* 5 (2004), 1–12.
- [20] BOUDEC, J.-Y. L., AND THIRAN, P. *Network Calculus: A Theory of Deterministic Queueing Systems for the Internet*. Springer-Verlag New York, Inc., New York, NY, USA, 2001.
- [21] BRAMSON, M. Instability of FIFO queueing networks. *Ann. Appl. Prob.* (1994), 414–431.
- [22] BRAMSON, M. Instability of FIFO queueing networks. *Ann. Appl. Prob.* (1994), 693–718.
- [23] BRAMSON, M. Convergence to equilibria for fluid models of FIFO queueing networks. *Queueing Syst.* (1996), 5–45.
- [24] CHANG, C.-S. Stability, queue length, and delays of deterministic and stochastic queueing networks. *IEEE Transactions on Automatic Control* 39, 5 (1994), 913–931.
- [25] CHEN, H. Fluid approximations and stability of multi class queueing networks: Work conserving disciplines. *Ann. Appl. Prob.* (1995).

- [26] CRUZ, R. A calculus for network delay. Part I: Network elements in isolation. *IEEE Transactions on Information Theory* 37 (1991), 114–131.
- [27] CRUZ, R. A calculus for network delay. Part II: Network analysis. *IEEE Transactions on Information Theory* 37 (1991), 132–141.
- [28] DAI, J. On positive Harris recurrence of multiclass queueing networks: A unified approach via fluid limit models. *Ann. Appl. Prob.* (1995), 49–77.
- [29] DAI, J., AND JENNINGS, O. Stabilizing queueing networks with setups. *Mathematics of Operations Research* (2004).
- [30] DAI, J., AND MEYN, S. Stability and convergence of moments for networks and their fluid models. *IEEE Trans. on Automatic Control* (1995), 1889–1904.
- [31] DAI, J. G., AND JENNINGS, O. B. Stabilizing queueing networks with setups. *Math. Oper. Res.* 29, 4 (2004), 891–922.
- [32] DÍAZ, J., KOUKOPOULOS, D., NIKOLETSEAS, S., SERNA, M., SPIRAKIS, P., AND THILIKOS, D. Stability and non-stability of the FIFO protocol. In *Proc. of the 13th Annual ACM Symposium on Parallel Algorithms and Architectures* (2001), pp. 48–52.
- [33] ECHAGÜE, J., CHOLVI, V., AND FERNÁNDEZ, A. Universal stability results for low rate adversaries in packet switched networks. *IEEE Communication Letters* 7 (2003), 578–580.
- [34] ECHAGÜE, J., PRIETO, M., VILLADANGOS, J., AND CHOLVI, V. Distributed algorithm to provide qos by avoiding cycles in routes. In *Quality of Service in the Emerging Networking Panorama* (2004), vol. 3266 of *Incs*, pp. 224–236.
- [35] FIDLER, M., AND EINHO, G. Routing in turn-prohibition based feed-forward networks. In *Proceedings of IFIP-TC6 Networking* (2004), vol. 3042 of *Incs*, pp. 1168–1179.
- [36] GAMARNIK, D. Stability of adversarial queues via fluid model. In *Proc. of the 39th Annual Symposium on Foundations of Computer Science* (1998), pp. 60–70.
- [37] GERSHWIN., S. Stochastic scheduling and set-ups in manufacturing systems. *Internat. J. Production Res.* 33 (1995), 1849–1870.
- [38] GOEL, A. Stability of networks and protocols in the adversarial queueing model for packet routing. *Networks* (2001).
- [39] HAVERKORT, B. R. *Performance of Computer Communication System. A Model-Based Approach*. John Wiley and Sons, 2003.

- [40] JENNINGS., O. *Multiclass queueing networks with setup delays: Stability analysis and heavy traffic approximation*. PhD thesis, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA., 2000.
- [41] KELLY, F. P. *Reversibility and Stochastic Systems*. John Wiley & Sons, 1979.
- [42] KIWI, M., AND RUSSELL, A. The chilean highway problem. *Theory of Computing Systems* (2004), 329–342.
- [43] KLEINROCK, L. *Queueing Systems, Volume 1: Theory*. John Wiley & Sons, 1975.
- [44] KOUKOPOULOS, D., MAVRONICOLAS, M., NIKOLETSEAS, S. E., AND SPIRAKIS, P. G. The impact of network structure on the stability of greedy protocols. *Theory Comput. Syst.* 38, 4 (2005), 425–460.
- [45] KOUKOPOULOS, D., MAVRONICOLAS, M., AND SPIRAKIS, P. FIFO is unstable at arbitrarily low rates (even in planar networks). *Electronic Colloquium on Computational Complexity* 10 (2003).
- [46] KOUKOPOULOS, D., MAVRONICOLAS, M., AND SPIRAKIS, P. Instability of networks with quasi-static link capacities. In *10th International Colloquium on Structural Information Complexity* (2003), vol. 17 of *Proceedings in Informatics*, Carleton Scientific, pp. 179–194.
- [47] KOUKOPOULOS, D., MAVRONICOLAS, M., AND SPIRAKIS, P. Performance and stability bounds for dynamic networks. In *7th International Conference on Parallel Architectures, Algorithms and Networks* (2004), IEEE Computer Society Press, pp. 239–246.
- [48] KUMAR, P., AND SEIDMAN, T. Dynamic instabilities and stabilization methods in distributed real-time scheduling of manufacturing systems. *IEEE Trans. on Automatic Control* (1990), 289–298.
- [49] LEIGHTON, F. T., MAGGS, B. M., AND RAO, S. Packet routing and job-shop scheduling in (congestion + dilation) steps. *Combinatorica* 14, 2 (1994), 167–186.
- [50] LOTKER, Z., PATT-SHAMIR, B., AND ROSEN, A. New stability results for adversarial queueing. *SIAM Journal on Computing* 33, 2 (2004), 286–303.
- [51] ROSÉN, A., AND TSIRKIN, M. On delivery times in packet networks under adversarial traffic. In *16th ACM Symposium on Parallel Algorithms and Architectures* (2004), ACM Press, pp. 1–10.
- [52] RYBKO, A., AND STOLYAR, A. Ergodicity of stochastic processes describing the operation of open queueing networks. *Prob. Inf. Trans.* 28 (1996), 366–375.

- [53] SANTOS, A., FERNÁNDEZ, A., AND LÓPEZ, L. Evaluation of packet scheduling policies with application to real-time traffic. In *Actas de las V Jornadas de Ingeniería Telemática, JITEL 2005* (Vigo, Spain, 2005).
- [54] STAROBINSKI, D., KARPOVSKY, M., AND ZAKREVSKI, L. Application of network calculus to general topologies using turn-prohibition. *IEEE/ACM Transaction on Networking* 11, 3 (2003), 411–421.
- [55] TASSIULAS, L., AND GEORGIADIS, L. Any work-conserving policy stabilizes the ring with spatial re-use. *IEEE/ACM Transactions on Networking* 4 (2001).
- [56] WEINARD, M. The necessity of timekeeping in adversarial queueing. In *4th International Workshop on Efficient and Experimental Algorithms* (2005), vol. 3503 of *LNCS*, Springer-Verlag, pp. 440–451.
- [57] WEINARD, M. Deciding the fifo stability of networks in polynomial time. In *6th International Conference on Algorithms and Complexity* (2006), vol. 3998 of *lncs*, pp. 81–92.