Taller de sistemas Cíber–Físicos

Curso 2021

Primer Semestre

Implementación de un sistema de QoS Slicing para WiFi en NS3

Informe

Autores:
Nicolás Cámera
Diego Sellanes

Dirigido a:
Matías Richart

23 de Agosto de 2021

Índice

1. Introducción

El objetivo de este proyecto, se basa en tomar un modelo de solución a un problema cíber – físico, que solamente está explicado en la tesis de Matías Richart [1] de manera teórica y mediante un pseudocódigo, e implementar ese modelo en el simulador NS3. El modelo es como solucionar *Slicing* con calidad de servicio en WiFi.

El por qué el modelo a tratar es un sistema cíber – físico, se podrá observar más en detalle en las siguientes secciones, pero, para tener una pequeña introducción, en el modelo se toman medidas del medio, como lo son la capacidad de transmisión de cada cliente de un AP.

1.1. Conceptos clave

El objetivo de esta subsección, es introducir los conceptos claves que van a estar presentes a lo largo de este informe.

1.1.1. *Slicing*

El *Slicing* consiste en particionar una red física en varias partes lógicas auto contenidas (llamadas *slices*). De este modo, las *slices* se pueden adaptar para ofrecer requisitos de rendimiento o funcionales. Mas aún, una característica clave del paradigma de *Slicing* es el de proveer aislamiento de los recursos, así como permitir un uso eficiente de los mismos.

En definitiva, una *slice* puede ser considerada como una red virtual de punta a punta.

1.1.2. *Slicing* con calidad de servicio (QoS *Slicing*)

QoS *Slicing* es una variante de *Slicing*, la cual especifica que las *slices* tienen requerimientos de rendimiento determinados. Una forma de modelar estos requisitos en las *slices*, es clasificar en el tipo de tráfico existente en la red. Una abstracción de esto, se presenta en la figura 1.1.2.1.

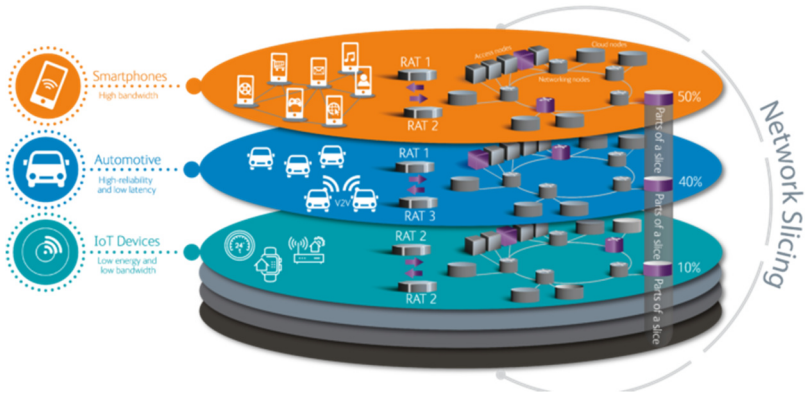


Figura 1.1.2.1. *Slicing* con calidad de servicio.

1.1.3. *Slicing* con calidad de servicio en WiFi

Para poder garantizar calidad de servicio, hay que poder reservar recursos en cada *slice*. Esto no es algo innato de WiFi, ya que cuando un dispositivo quiere transmitir, si el canal está libre, lo ocupa durante todo el tiempo que pueda. Entonces, no es posible dividir el ancho de banda de transmisión. Una alternativa para lidiar con esto, es gestionar el uso del tiempo de transmisión (*airtime*) de cada *slice*.

Finalmente, dados requerimientos de QoS en un *slice* y la capacidad del medio de transmisión (la cual, es variable en el tiempo), se tiene que poder lograr decidir cuánto *airtime* necesita reservarse para ese *slice*. Esto es lo que soluciona el modelo planteado en la tesis de Matías Richart [1].

1.1.4. Simulador NS3

En este simulador, es en donde se implementará la solución del modelo cíber – físico propuesto en la tesis de Matías Richart.

NS3 [2] es un simulador de redes basado en eventos discretos, esto es, cada acción es modelada con un evento, que podrá disparar ciertas acciones en otras entidades. Además, está implementado en `C++` y de código abierto. Al ser de código abierto, permite modificar su implementación con facilidad, por lo que es utilizado mayoritariamente de forma educacional y para investigación.

Adicionalmente, cabe mencionar que el simulador NS3 posee escenarios y ejemplos muy avanzados que se ajustan a la realidad, los cuales permitirán estudiar a fondo la evaluación el funcionamiento y comportamiento de la solución a implementar.

Finalmente, esta herramienta es actualizada constantemente a los requerimientos de las redes modernas, siendo mantenida voluntariamente por personas en todo el mundo.

1.2. Solución del problema planteado

Como ya se mencionó en la sección 1.1.3., la solución modelada a implementar, es cómo resolver *Slicing* con calidad de servicio en WiFi. A modo de resumen de todo el proceso de modelado que se abarco en la tesis de Matías Richart, se presenta esta sección.

Primero, se parte de lo que garantiza el modelado de la solución propuesta en la tesis de Matías Richart. En la misma, se propone un algoritmo, el cuál funciona en cada AP y provee las siguientes garantías a todos los clientes que estén en las *slices* de ese AP:

- Garantías mínimas de *bit rate* de transmisión.
- Garantías de un *delay* de cola acotado.

Para llegar a esto, se tiene lo siguiente:

- El escenario inicial se plantea como un problema de reserva de recursos dinámica, la cual puede ser optimizada.
- Este problema, se modela como un proceso estocástico desconocido, dado que se desconocen las tasas de arribo de tráfico de los diferentes *slices* y clientes, así como la capacidad del canal de transmisión.
- Así, se considera que el AP está operando en un entorno estocástico.
- En el problema, se modelan los requisitos para que se cumplan las garantías de *bit rate* de transmisión y de *delay* de cola acotado.
- Finalmente, se obtiene una solución aproximada al problema por medio de la aplicación de la Teoría de Estabilidad de Lyapunov.

Está solución aproximada, transforma el problema estocástico en un problema de toma de decisiones, en el cual, en cada instante de tiempo en donde el AP tiene el canal libre para transmitir, se tiene que decidir:

- Qué cliente va a transmitir.
- Qué paquetes se van a descartar para cada cliente.

En la siguiente sección se presenta el algoritmo que plantea esta toma de decisiones.

1.3. Algoritmo

A continuación, se cita el algoritmo de la tesis de Matías Richart que soluciona QoS *Slicing* en WiFi.

Algorithm 3: QoS Slicing Scheduler Pseudocode.

function *Scheduler()* **is**

input : $V, Q_{n_s}, H_s, K_s, D_{n_s}^{max}, \gamma_{n_s}^{max}, \epsilon_{n_s}, \omega \quad \forall s \in [1,S],\ \forall n_s \in [1,C_s]$

output: Scheduling and dropping decisions on each slot t

/* Initialize queues */

foreach $n_s \mid s \in [1,S], n_s \in [1,C_s]$ **do**

  $Z_{n_s} \leftarrow 0;\ G_{n_s} \leftarrow 0;\ Y_{n_s} \leftarrow 0;\ U_s \leftarrow 0;$

end

while *true* **do**

  /* Observe the current channel capacity and arrivals */

  foreach $n_s \mid s \in [1,S], n_s \in [1,C_s]$ **do**

    $\mathbb{C}_{n_s} \leftarrow getCapacity(n_s);$

    $A_{n_s} \leftarrow getArrivals(n_s);$

  end

  /* Compute the vector of benefits B using only clients with queued packets */

  $\boldsymbol{B} = \mathbb{C} * (\boldsymbol{G} + \boldsymbol{Q} + \boldsymbol{Z}) - \boldsymbol{U};$

  /* Find the client i with the maximum benefit */

  $i \leftarrow arg\,max\ \boldsymbol{B};$

  /* Get the queue of client i */

  $queue \leftarrow GetQueue(queueList, i);$

  /* Transmit packets for a quantum period */

  while $queue.excess < 0$ **do**

    $airtime \leftarrow transmitPacket(queue);$

    $queue.excess \leftarrow queue.excess + airtime;$

  end

  $queue.excess \leftarrow queue.excess - QUANTUM;$

  foreach $n_s \mid s \in [1,S], n_s \in [1,C_s]$ **do**

    if $Q_{n_s} + Z_{n_s} > Y_{n_s}$ **then**

      $dropPackets(n_s, D_{n_s}^{max});$

    end

    /* Calculate the auxiliary variable values */

    $\gamma_{n_s} \leftarrow \min\left\{\frac{V}{Y_{n_s}} - \frac{1}{\omega},\ \gamma_{n_s}^{max}\right\}$

    /* Update queue backlogs */

    $Z_{n_s} \leftarrow [Z_{n_s} + \epsilon_{n_s} - R_{n_s} - D_{n_s}]^+$

    $G_{n_s} \leftarrow [G_{n_s} - R_{n_s} + K_s]^+$

    $U_s \leftarrow [U_s + X_s - H_s]^+$

    $Y_{n_s} \leftarrow [Y_{n_s} + \gamma_{n_s} - u_{n_s}]^+$

  end

end

end

El algoritmo se ejecuta en cada AP y su ejecución se da siempre que el canal de transmisión esté libre. En este punto, se puede decir que las instrucciones 3 a 5 se ejecutan cada vez que se agrega un cliente en un *slice* y, la porción del algoritmo real, está en las instrucciones 7 a 36.

Se puede observar que el algoritmo utiliza un conjunto de variables, las cuales, en su mayoría, llamaremos variables virtuales (o también, colas virtuales). Estas variables comprenden un papel clave en el funcionamiento del algoritmo y, su semántica, se puede encontrar en la sección 1.3.1.

Se identifican las siguientes fases en el algoritmo:

- Toma de medidas del medio; identificado por las instrucciones 8 a 11. En estas instrucciones, se censa del medio la capacidad de transmisión de los clientes, $\mathbb{C}_{n_s}$ (instrucción 9) y se mide la cantidad de nuevos paquetes que arribaron para los clientes, A_{n_s} (instrucción 10).

- Decidir cuál es el próximo cliente que tiene que transmitir; identificado por las instrucciones 12 a 23. En estas instrucciones, se obtiene el cliente de máximo beneficio y se le permite transmitir durante una cierta duración (QUANTUM), la cual, comúnmente llamaremos *slot* de transmisión.

  El cliente de máximo beneficio, es aquel que tiene esos valores más grandes en las variables. Esto indica, que es el que está más lejos de que se garanticen sus requisitos.

  La obtención del cliente de máximo beneficio está representada por las instrucciones 12 a 17, mientras que las instrucciones 18 a 23 representan la transmisión de paquetes en el *slot* de transmisión.

- Tomar decisiones de descartes de paquetes y actualizar variables virtuales; identificado por las instrucciones 24 a 36. Una vez que el cliente de máximo beneficio transmitió, se toman decisiones de descartes de paquetes para todos los clientes (instrucciones 25 a 27), asegurando así, el *delay* de cola acotado; y se actualizan las variables virtuales de los clientes (instrucciones 28 a 34), las cuales, permiten al algoritmo saber cuál es el cliente de máximo beneficio y tomar decisiones de descartes de paquetes.

Como se puede apreciar, el funcionamiento del algoritmo es lo que ya se vio en la sección 1.2. En cada *slot* de tiempo (o transmisión), el algoritmo decide qué cliente tiene que transmitir (para garantizar los requisitos de *bit rate* de transmisión o, también llamado, *throughput*) y toma decisiones de descartes de paquetes en todos los clientes (para asegurar el *delay* de cola acotado).

1.3.1. Semántica de cada variable

A continuación, se presenta el significado de la mayoría de las variables utilizadas en el algoritmo:

- $\mathbb{C}_{n_s}$: Capacidad de transmisión del cliente n_s en cada *slot* de transmisión.
- A_{n_s}: Arribos de cada cliente n_s en cada *slot* de transmisión.
- G_{n_s}: Modela las garantías de *bit rate* que aún hacen falta cumplir para el cliente n_s.
- Q_{n_s}: Modela la cantidad de paquetes que tiene para transmitir el cliente n_s.
- Z_{n_s}: Modela una prioridad dinámica en cada cliente n_s, la cual, adicionalmente ayuda tomar decisiones de descartes de paquetes.
- Y_{n_s}: Ayuda a modelar un THRESHOLD para saber cuándo en necesario descartar paquetes.
- D_{n_s}: Cantidad de paquetes descartados de cada cliente n_s en cada *slot* de transmisión.
- R_{n_s}: *Bit rate* de cada cliente n_s en cada *slot* de transmisión.
- ϵ_{n_s}: Es el parámetro de cada cliente n_s que ayuda a ajustar la prioridad dinámica, Z_{n_s}.
- u_{n_s}: Función del rendimiento de cada cliente n_s, $u_{n_s} = A_{n_s} - D_{n_s}$.
- K_s: Garantías mínimas de *bit rate* que asegura el *slice s*.
- X_s: Proporción del *slot* de tiempo en la cual transmitió el *slice s*.
- H_s: Límite de capacidad del *slice s* en cada *slot* de transmisión.
- U_s: Mide cuánto un *slice s* ha transmitido por sobre su límite de capacidad H_s.
- V, ω: Parámetros de ajuste.

2. Implementación de QoS Slicing en el simulador NS3

Para poder llevar acabo la implementación del algoritmo presentado en la sección 1.3., primero se tuvo que entender bien a fondo su funcionamiento, así como el significado de cada variable. A continuación, se tuvo que entender la implementación antigua que estaba de WiFi en NS3. En este punto, se podría decir que se llevó a cabo un proceso de ingeniería inversa.

Cabe destacar que, la implementación del algoritmo fue lo que llevó más tiempo realizar en todo este proyecto.

2.1. Cambios realizados en el simulador NS3

Para implementar el algoritmo en NS3, se tuvo que:

- Hacer una cola distinta para cada cliente de cada *slice*. Esto es debido a que el protocolo WiFi maneja una única cola para todos los paquetes, pero hacía falta poder identificar de manera rápida que paquetes pertenecía a la dupla de cliente y *slice*.
- Implementar todas las estructuras necesarias para manejar todas las variables que usa el algoritmo. En este ámbito, algunas de las variables se implementaron en la estructura de cada cola de los clientes, y otras variables se implementaron en la estructura de cada *slice*.
- Modelar funciones auxiliares necesarias para poder manejar flujos principales y alternos que maneja el algoritmo.
- Modelar todas las funciones necesarias para poder configurar los parámetros del escenario desde un archivo de configuración. En este ámbito, se logró hacer *getters* y *setters* para poder configurar todos los escenarios de prueba desde un archivo externo de configuración.
- Se tuvo que expresar todas las variables en la misma unidad. Algo que puede no apreciarse a primera vista en el algoritmo, es el tipo de las variables. Se optó porque todas las variables estuvieran medidas en paquetes por *slot* de tiempo, dado que era la mejor convención a usar según el planteamiento del algoritmo.
- Se tuvo que idear un mecanismo para hacer frente a la variación de la duración de los *slots* de transmisión. En WiFi, para hacer más eficientes las transmisiones, el contenido de varios paquetes se encapsula dentro de un único cabezal y se forma un *agreement*. Esto permite transmitir varios paquetes a la vez sin tener que realizar muchos cálculos intermedios en el AP. El problema que conlleva esto, es que los *agreements* formados son de gran tamaño, lo que provoca que el canal este ocupado por mucho tiempo, haciendo que el AP esté transmitiendo a un cliente por más tiempo (QUANTUM) del estipulado. Para lidiar con este problema, en la actualización de las variables del algoritmo, las cuales suponen una duración del *slot* de

transmisión fija, se ideó un mecanismo basado en la duración del *slot* de transmisión, para ponderar la actualización de aquellas variables que dependieran de la duración del *slot* de transmisión.

2.1.1. Cambios realizados en el Scheduler

Cuando el AP tiene el canal libre para transmitir, en la implementación de NS3, se llama a una función que decide cuál va a ser el próximo cliente que ocupe el canal. Esta función se conoce como el *Scheduler*.

Para poder implementar el algoritmo, se tuvo que actualizar el antiguo *Scheduler* para que soporte toda la lógica del algoritmo presentado en la sección 1.3. En la figura 2.1.1.1 se pueden ver un diagrama de flujo de cómo funciona el nuevo *Scheduler*.

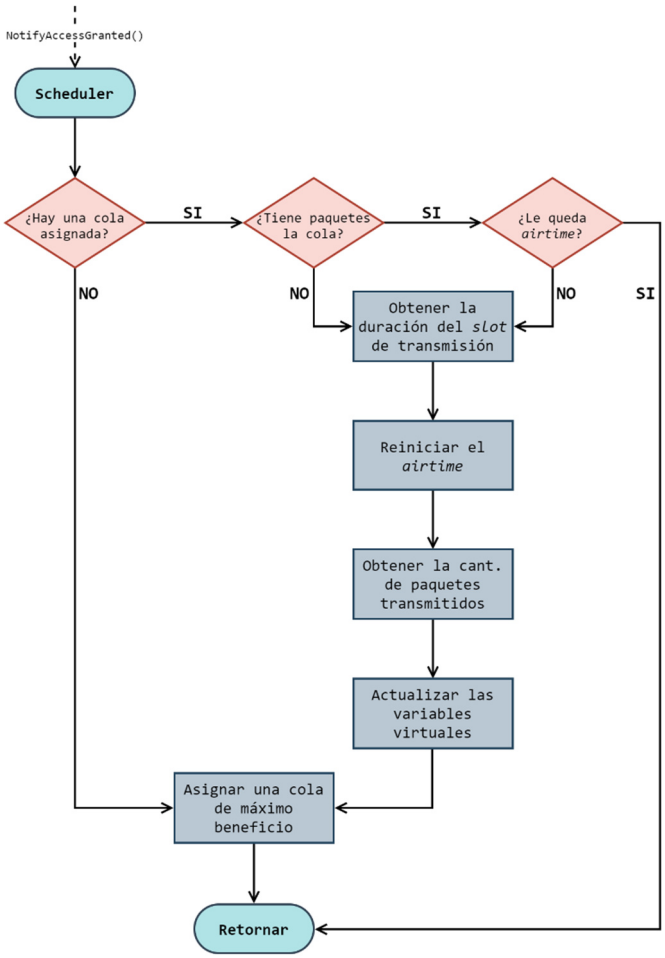


Figura 2.1.1.1. Diagrama de flujo sobre el funcionamiento del nuevo *Scheduler*.

2.1.2. Implementación del cálculo de los arribos

En la instrucción 10 del algoritmo presentado en la sección 1.3, se puede ver como siempre se toma del medio la cantidad de nuevo arribos que han llegado a todos los clientes. Para poder modelar esto en NS3, se tuvo que contemplar lo siguiente:

- Supóngase que se cuenta con un cliente de máximo beneficio, `queueInfo`.
- Se sabe que cada cliente tiene dos colas de paquetes; una cola para los nuevos paquetes y otra cola para los paquetes a retransmitir.
- Se quiere calcular la cantidad de arribos desde un tiempo t a un tiempo $t+1$.
- Supóngase que en el instante de tiempo t el cliente `queueInfo` tiene 9 paquetes y en el instante de tiempo $t+1$ también tiene 9 paquetes, pero entre t y $t+1$ transmitió 4 paquetes y le llegaron 4 paquetes.

Este escenario está representando en la figura 2.1.2.1 y la solución está en la figura 2.1.2.2.

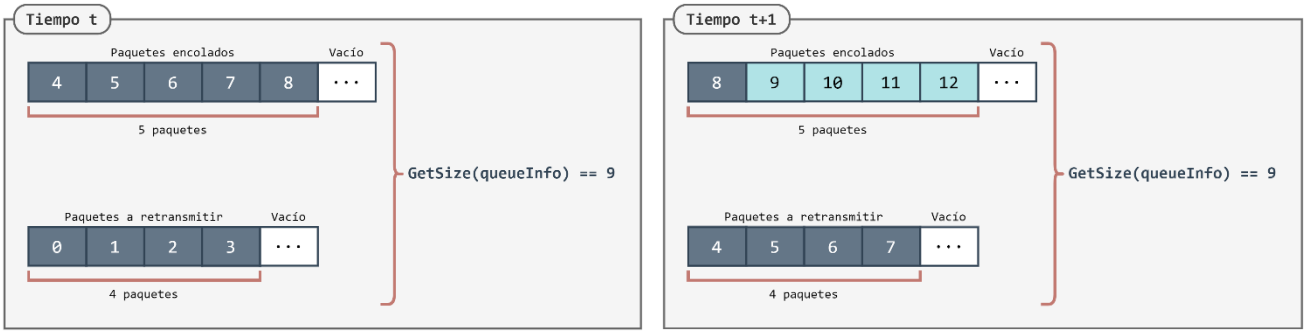


Figura 2.1.2.1. Representación del escenario para modelar el cálculo de los arribos.

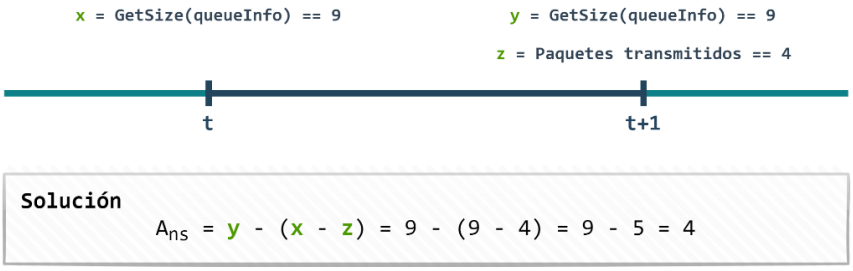


Figura 2.1.2.2. Solución al modelado del cálculo de los arribos.

En esta solución, la variable:

- `y` modela la cantidad de paquetes que tiene el cliente `queueInfo` en el instante $t+1$, contando los paquetes que arribaron.
- `x - z` modela la cantidad de paquetes que debería tener el cliente `queueInfo` en el instante $t+1$, si no hubieran arribado nuevos paquetes.

Por lo tanto, la diferencia de las variables `y` e `x - z` representa efectivamente los arribos del tiempo t al tiempo $t+1$.

Hay que remarcar un último detalle, la medición de los paquetes transmitidos, $\mathbf{z}$. Para poder obtener esta medida, se tuvo que implementar en un cierto módulo del simulador NS3, el manejado de la cantidad de paquetes que llegan al otro lado en cada *slot* de transmisión. Este cálculo es posible obtenerlo, por la forma de funcionar de WiFi, dado que se tiene confirmación, por medio de los ACKs, de los paquetes que son recibidos.

2.1.3. Implementación del cálculo de las capacidades

En la instrucción 9 del algoritmo presentado en la sección 1.3, se puede ver como siempre se mide del medio la capacidad de transmisión de todos los clientes. Para poder modelar esto en NS3, se realizó lo siguiente:

- Cada cliente empieza con la máxima capacidad de transmisión del medio, $\mathbb{C}_{n_s}^{max}$.
- Una vez que un cliente transmite, se actualiza la capacidad de transmisión, $\mathbb{C}_{n_s}$, con la cantidad de paquetes que transmitió.
- Esa capacidad se mantiene hasta que el cliente vuelva a transmitir y se vuelva a actualizar con la última cantidad de paquetes que transmitió.

3. Evaluación de la solución de QoS Slicing

En esta sección, se evalúa el comportamiento y rendimiento de la solución propuesta de QoS *slicing*.

El objetivo de esta evaluación, es mostrar como la solución provee las garantías de *throughput* y *delay* requeridas en los *slices* configurados, para la red WiFi propuesta. Hay que remarcar que, la solución provee dichas garantías, solo cuando se tienen los recursos necesarios.

Por último, se examinarán dos escenarios posibles de configuración. Ambos escenarios tienen los mismos requisitos en los *slices* desplegados, pero difieren en el comportamiento de los clientes. En un escenario, los clientes están fijos y en el otro, los clientes se pueden mover. Este último escenario, es el más cercano a la realidad y, por ende, es de mayor interés para evaluar el comportamiento y rendimiento de la solución.

3.1. Configuración común entre los escenarios

Los escenarios simulados, cuentan con un único Punto de Acceso (*Access Point*, AP) WiFi, tres *slices* y un cliente conectado a cada *slice*. El *slice* 1 está pensado para videoconferencia en vivo de alta calidad (por ejemplo, conferencias empresariales), y los *slices* 2 y 3 para videoconferencias en vivo de calidad media (por ejemplo, reuniones cortas de equipos de desarrollo). Los clientes se disponen en un rectángulo de 20×20 alrededor del AP, tal como se muestra en la figura 3.1.1.

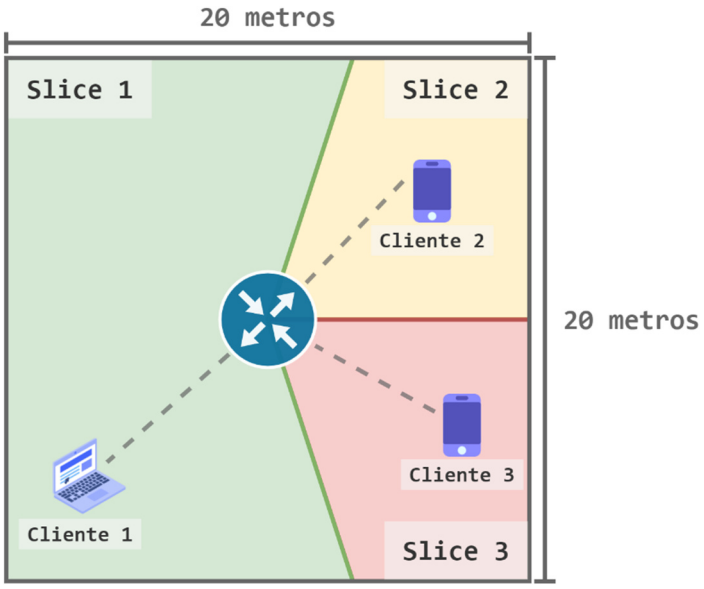


Figura 3.1.1. Disposición de los clientes alrededor del AP.

La configuración de cada *slice* se puede encontrar en la tabla 3.1.1. Cabe destacar que, las garantías de *bit rate* en los tres *slices* son más altas de lo que deberían ser, dado que se busca probar los límites de los recursos. Por otro lado, las garantías de *delay* son más ajustadas a la realidad, pero, aun así, siguen siendo bastantes exigentes.

	***Slice* 1**	***Slice* 2**	***Slice* 3**
Aplicación	Videoconferencia en vivo de alta calidad	Videoconferencia en vivo de media calidad	Videoconferencia en vivo de media calidad
Patrón de tráfico	*Bit rate* constante a 60 Mbps	*Bit rate* constante a 20 Mbps	*Bit rate* constante a 20 Mbps
***Bit rate* garantizado**	*Bit rate* constante de 30 Mbps	*Bit rate* constante de 10 Mbps	*Bit rate* constante de 10 Mbps
***Delay* garantizado**	50 *ms*	50 *ms*	50 *ms*
Límite de capacidad	60%	20%	20%
Número de flujos	1	1	1
Capacidad máxima de transmisión	80 Mbps		
QUANTUM	2.5 *ms*		

Tabla 3.1.1. Configuración del tráfico y de los *slices*.

Con la tabla 3.1.1 y junto al uso de la expresión (5.114) de la sección 5.9.4 [1], se pueden deducir todos los parámetros de entrada y de configuración del algoritmo 1. Los valores de dichos parámetros se pueden encontrar en la tabla 3.1.2.

	***Slice* 1**	***Slice* 2**	***Slice* 3**
H_s	0,6	0,2	0,2
K_s	30 Mbps	10 Mbps	10 Mbps
$A_{n_s}^{max}$	60 Mbps	20 Mbps	20 Mbps
$D_{n_s}^{max}$	60 Mbps	20 Mbps	20 Mbps
$\gamma_{n_s}^{max}$	60 Mbps	20 Mbps	20 Mbps
ϵ_{n_s}	2,11	1,11	1,11
V	3		
ω	1		
$\mathbb{C}_{n_s}^{max}$	80 Mbps		
$R_{n_s}^{max}$	80 Mbps		
QUANTUM	2.5 *ms*		

Tabla 3.1.2. Parámetros de entrada y de configuración del algoritmo 1.

Cabe remarcar que, los valores de Q_{n_s} no están en la tabla 3.1.2, dado que es un parámetro que cambia constantemente su valor.

A continuación, se expondrán y analizarán los resultados para los dos escenarios simulados. Ambos escenarios, tuvieron un tiempo de simulación de 500 segundos.

Todos los resultados son generados por la implementación del algoritmo en NS3, junto con la aplicación de un algoritmo de *parsing* de la información provista por la salida NS3.

3.2. Escenario estático

Este escenario cuenta con toda la configuración planteada en la sección 3.1, en adicción a que, los clientes no se mueven dentro del rectángulo de 20×20 una vez se posicionan dentro del mismo.

3.2.1. Análisis del *throughput*

En esta sección, primero se mostrará la evolución del *throughput* de los clientes del escenario, para posteriormente, exponer esos resultados en *boxplots* que expongan más detalles de la variación del *throughput*.

Los resultados de la evolución *throughput*, se pueden encontrar en la gráfica 3.2.1.1. En la gráfica 3.2.1.2. se pueden encontrar los *boxplots* de los resultados del *throughput*.

Gráfica 3.2.1.1. Resultados del *throughput* de los clientes en el escenario estático.

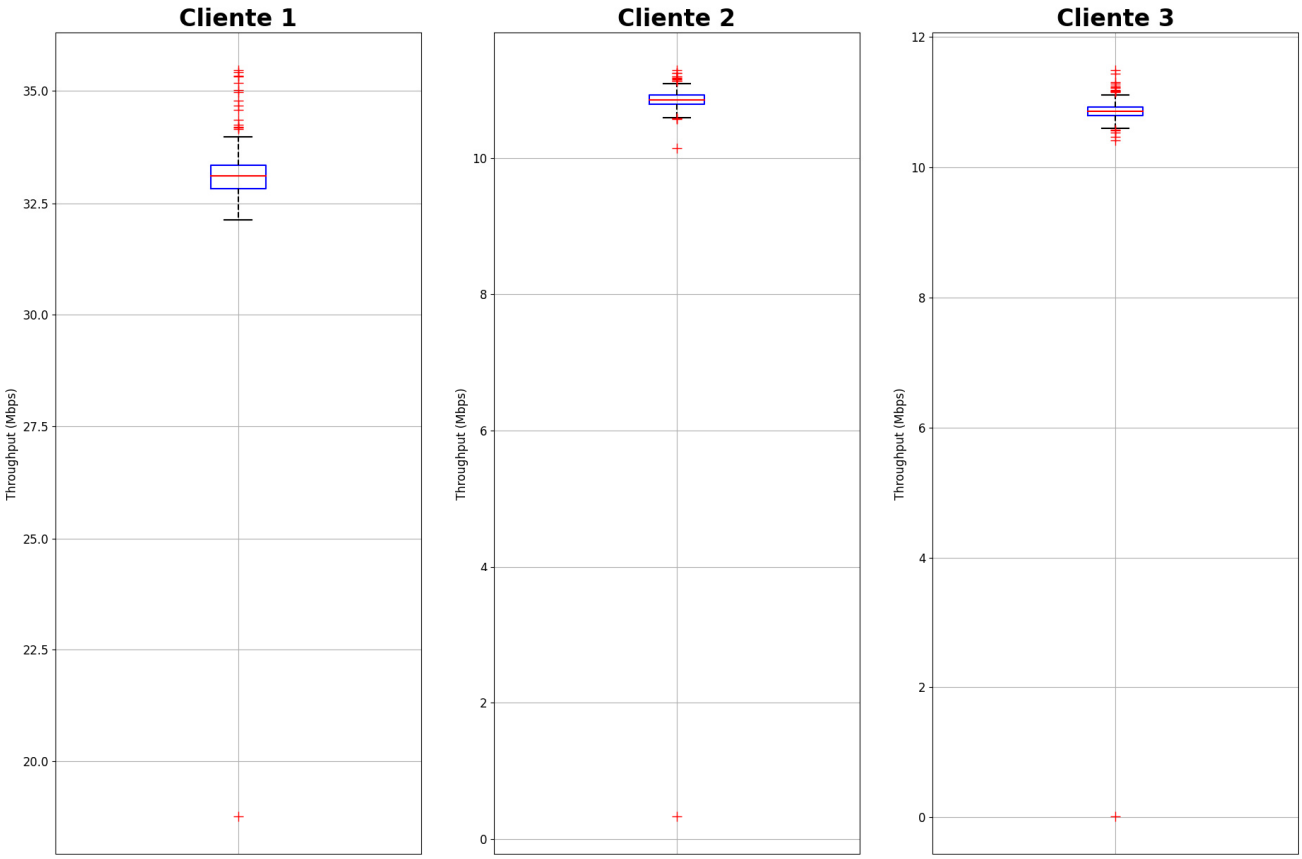
En la gráfica 3.2.1.1, se observa que, una vez pasada la etapa de convergencia del algoritmo (la cual dura unos pocos segundos al inicio), los resultados del *throughput* se mantienen por encima de lo que provee cada *slice*. Esto se debe a que, el mecanismo usado para contar la cantidad de bits transmitidos por segundo, contempla el tamaño del paquete como la suma del tamaño de los cabezales más la carga útil. Cuando se cuenta solo la carga útil de los paquetes, se obtienen los siguientes resultados de *throughput* promedio:

- Cliente 1: 30,6685 Mbps.
- Cliente 2: 10,0378 Mbps.
- Cliente 3: 10,0379 Mbps.

Como se puede observar, estos resultados de *throughput* promedio están mínimamente por encima de las garantías de *throughput* que provee cada *slice*; algo esperado, dado que no se observan muchas "anomalías" en cuanto a la variación del *throughput* a lo largo de la simulación. A su vez, esta ausencia de anomalías, permite que las garantías *throughput* de en los *slices* 2 y 3 sean prácticamente idénticas; algo destacable, dado que ambos *slices* proveen las mismas garantías.

Con este análisis, es de esperar que, surjan más "anomalías" que provoquen una mayor variación del *throughput* en el escenario dinámico, dado que los clientes van a tener libertad para moverse dentro de los parámetros que se configuren en el rectángulo del AP.

Finalmente, en la gráfica 3.2.1.2 se pueden observar los *boxplots* de los valores que se muestran en la gráfica 3.2.1.1, corroborando que, los resultados *throughput* de son muy estables y con poca variación. Principalmente, la mayor cantidad de datos atípicos, es muy seguro que estén proviniendo por la fase inicial en la que el algoritmo se está ajustando a las necesidades de cada *slice*.

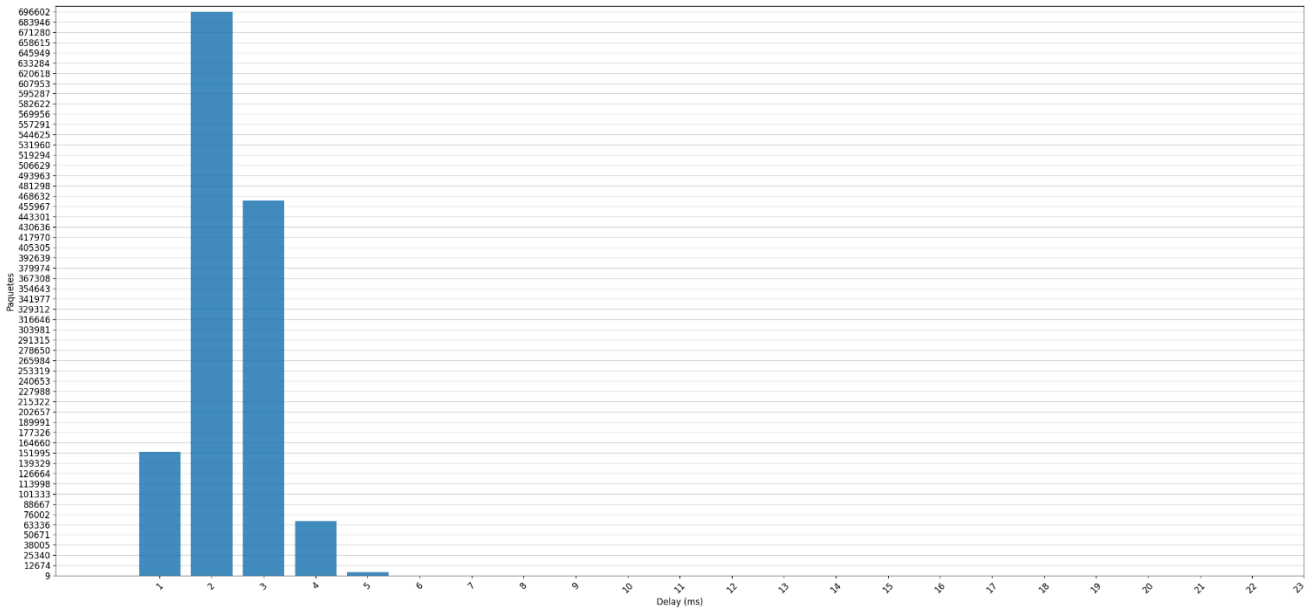


Gráfica 3.2.1.2. *Boxplots* del *throughput* de los clientes en el escenario estático.

3.2.2. Análisis del *delay*

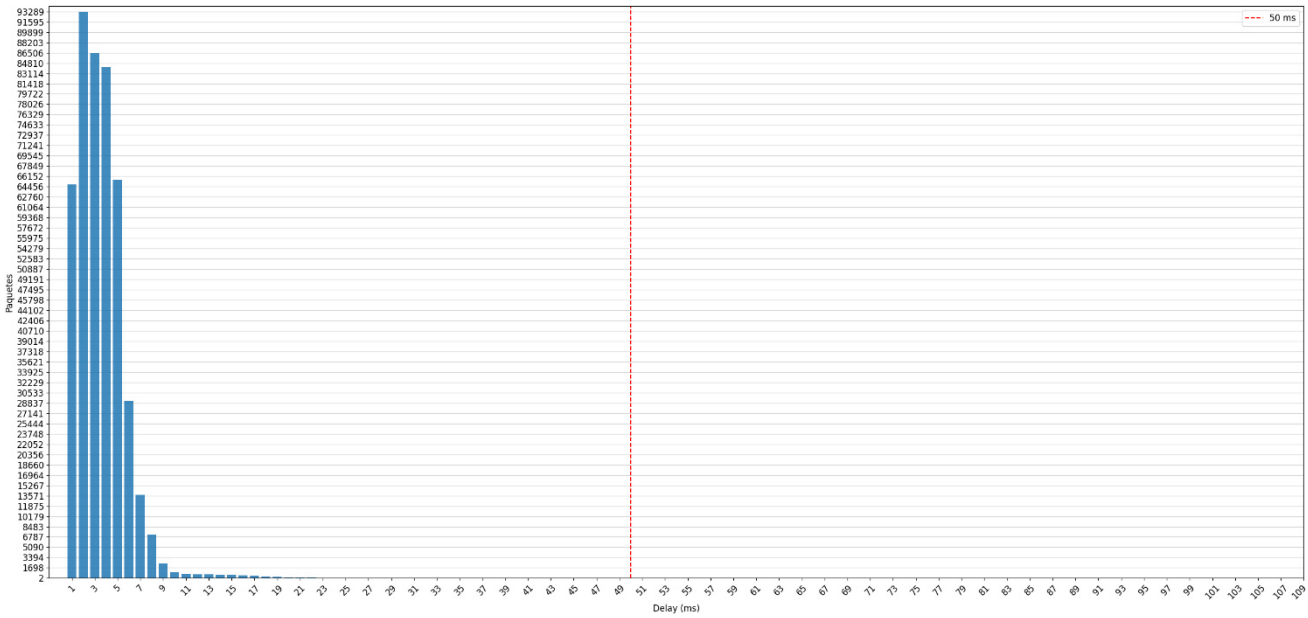
En esta sección, primero se mostrarán los histogramas de la cantidad de paquetes con *delay* de cada cliente, para posteriormente, graficar esos histogramas como *boxplots* y así, obtener otras métricas de lo que está sucediendo en la simulación. En adición a eso, también se mencionará la proporción de paquetes cuyo *delay* de cola no cumpla con la cota máxima establecida para el algoritmo, en los *slices* en lo que esto ocurra.

Los histogramas de la cantidad de paquetes con *delay* de cada cliente, se pueden encontrar en las gráficas 3.2.2.1, 3.2.2.2 y 3.2.2.3. Los *boxplots* de los histogramas se pueden encontrar en la gráfica 3.2.2.4.



Gráfica 3.2.2.1. Histograma de la cantidad de paquetes con *delay* del cliente 1 en el escenario estático.

Como se observa en la gráfica 3.2.2.1, para el cliente 1 se cumplen las garantías del *delay* acotado. Esto tiene varias explicaciones. La primera y la que se desprende por la naturaleza de este escenario, es que los clientes están fijos, lo cual provoca que el algoritmo funcione de manera muy estable al no presentarse anomalías en la variación de las variables que emplea, no teniendo que, por ende, aumentar o decrementar la cantidad de descartes de paquetes de un instante a otro. Por otro lado, en la simulación, el *slice* 1 es el primero en crearse, esto puede estar provocando que, al ser el único *slice* en el AP (por unos cuantos milisegundos), la convergencia de las garantías que se requieren para el mismo, se dé más rápido que en el resto de los *slices*. Esto podrá indagarse más a fondo, cuando se analice la evolución de las variables del algoritmo en cada cliente, en la sección 3.2.3.



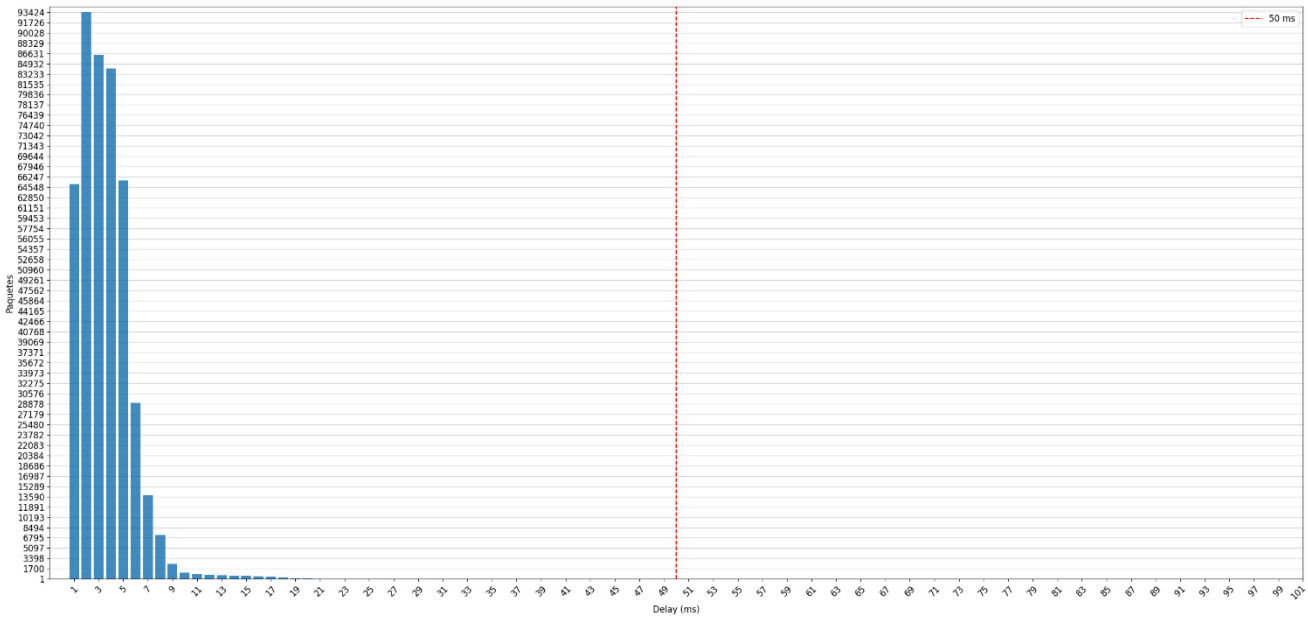
Gráfica 3.2.2.2. Histograma de la cantidad de paquetes con *delay* del cliente 2 en el escenario estático.

Como se observa en la gráfica 3.2.2.2, para el cliente 2 no se cumplen las garantías del *delay* acotado. En este caso, de los 453.850 paquetes representados en el histograma, 421 paquetes tienen *delay* de cola mayor a 50 *ms*. Es decir, un 0,0927% de los paquetes no cumple las garantías de *delay*. Esto tiene varias posibles explicaciones:

1. Los paquetes que no cumplen estas garantías de *delay* son los paquetes que pertenecen a la fase inicial, donde el algoritmo aún se está adaptando a las necesidades de los *slices*.
2. La duración del *slot* en trasmisión en la fase de inicio del algoritmo es varias veces mayor a lo establecido (2.5 *ms*), provocando que los intervalos en los que demoran en descartarse paquetes sea bastantes cercanos al *delay* máximo de cola establecido (50 *ms*).

Sobre el punto 1), para indagar más a fondo de si esto es lo que realmente ocurre, en la simulación, habría que no contar el *delay* de los paquetes en los primeros segundos en que el algoritmo se está ajustando. De este modo, se podría saber con total exactitud si la fase de convergencia es la que está dando paquetes con un *delay* de cola mayor a la cota establecida.

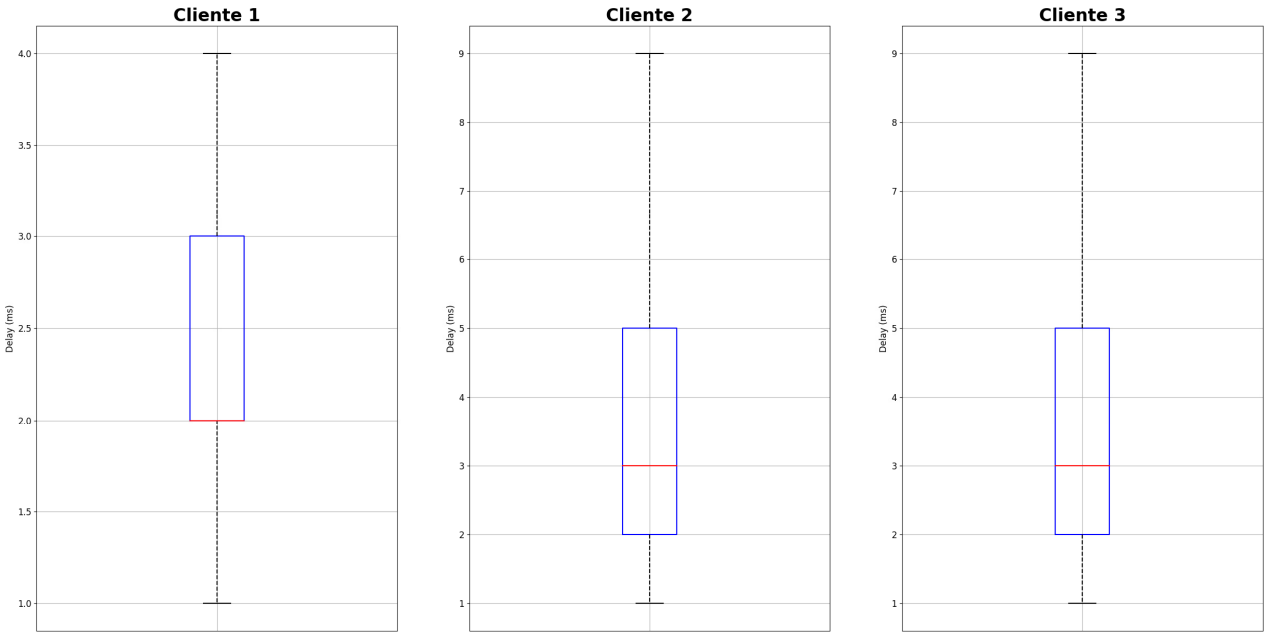
Sobre el punto 2), ciertamente la longitud del intervalo de transmisión puede estar afectando, dado que, por lo que se observa en la variable X_s de las gráficas 3.2.3.1, 3.2.3.2 y 3.2.3.3, la duración del *slot* de transmisión en la fase inicial del algoritmo, llega hasta ser 4 veces mayor al QUANTUM establecido. Es decir, algunos intervalos de transmisión hacen que se demore hasta 10 *ms* en actualizar las variables y tomar decisiones en los descartes de los paquetes; y 10 *ms* ya es un número más cercano al *delay* de cola máximo que se quiere garantizar (50 *ms*).



Gráfica 3.2.2.3. Histograma de la cantidad de paquetes con *delay* del cliente 3 en el escenario estático.

Como se observa en la gráfica 3.2.2.3, para el cliente 3, nuevamente no se cumplen las garantías del *delay* acotado. En este caso, de los 453.855 paquetes representados en el histograma, 246 paquetes tienen *delay* de cola mayor a 50 *ms*. Es decir, un 0,0542% de los paquetes no cumple las garantías de *delay*. Los posibles motivos por los cuales ocurre esto, son los mismos que los ya visto para el *slice* 2, dado que ambos *slices* tienen la misma configuración y, como se verá en la sección 3.2.3, ambos tienen un comportamiento muy similar en la evolución de las variables de sus clientes.

Por último, en la gráfica 3.2.2.4 se presentan los *boxplots* de los *delays* de los clientes. En la misma, se puede observar que la gran mayoría de los paquetes mantiene un *delay* de cola muy por debajo de la cota establecida.

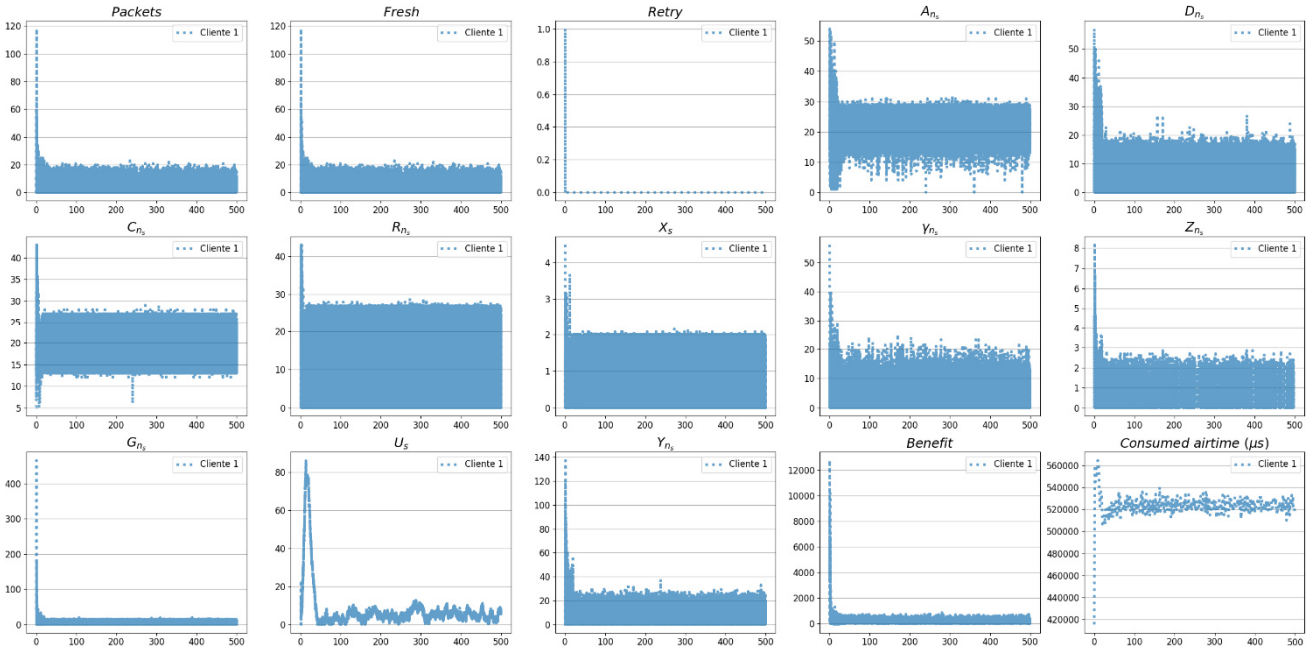


Gráfica 3.2.2.4. *Boxplots* de los *delays* de los clientes en el escenario estático.

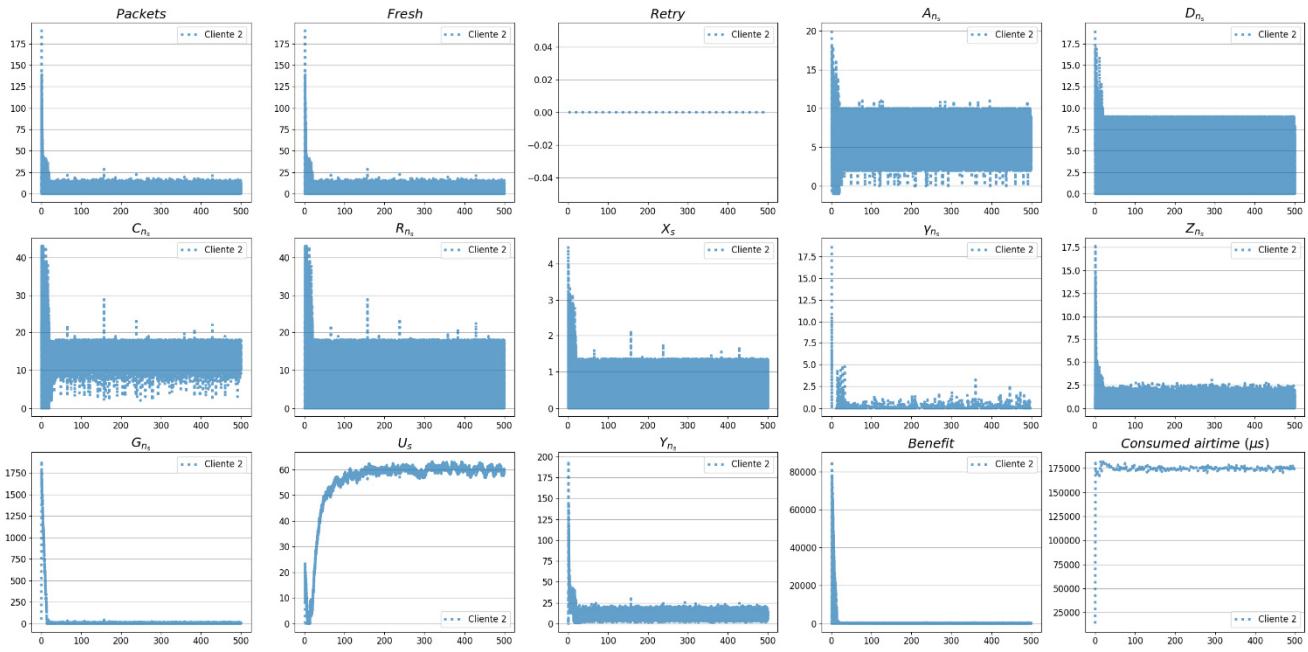
3.2.3. Análisis de las variables

En esta sección, se mostrará la evolución de las variables más importantes del algoritmo en cada cliente y *slice*, así como se expondrá la evolución de otras métricas, como lo son, el beneficio de cada cliente en cada intervalo de transmisión y el *airtime* consumido en cada segundo de la simulación.

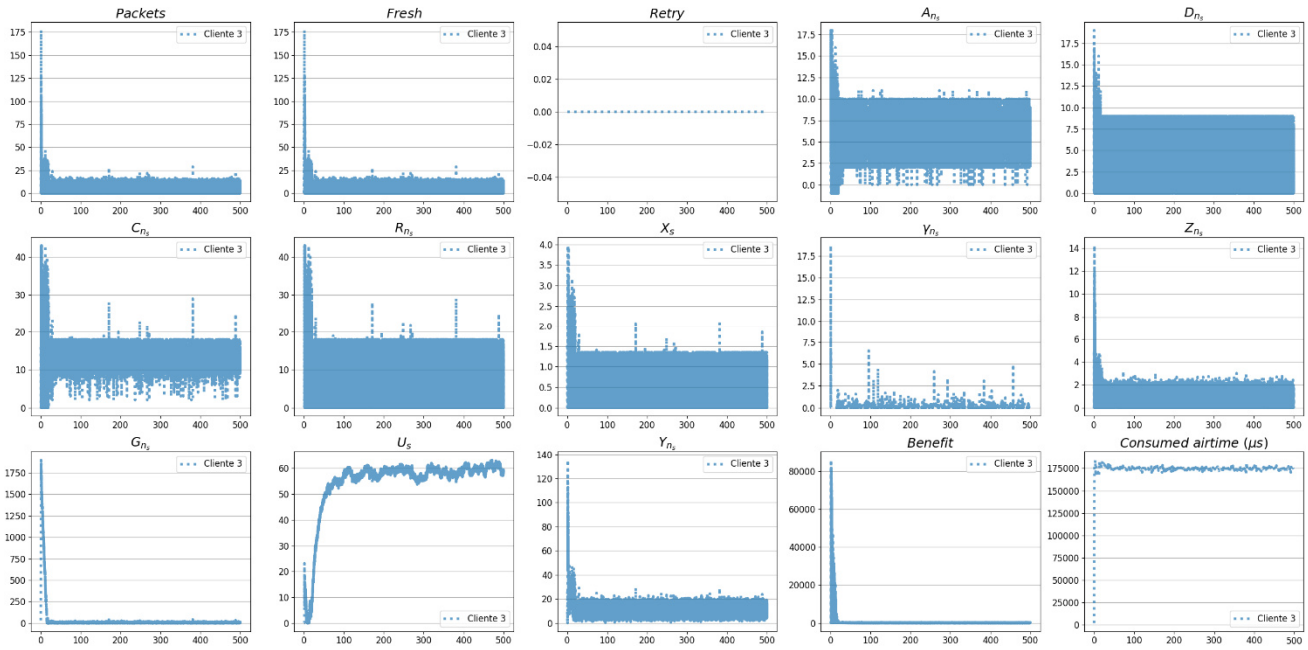
La evolución de las variables de cada cliente y *slice*, se pueden encontrar en las gráficas 3.2.3.1, 3.2.3.2 y 3.2.3.3. Recordar que, sin ser por las últimas dos gráficas de cada cliente y *slice*, todas las variables están expresadas en paquetes por *slot* de transmisión.



Gráfica 3.2.3.1. Evolución de las variables del cliente 1 y *slice* 1 en el escenario estático.



Gráfica 3.2.3.2. Evolución de las variables del cliente 2 y *slice* 2 en el escenario estático.



Gráfica 3.2.3.3. Evolución de las variables del cliente 3 y *slice* 3 en el escenario estático.

Como se observa en las gráficas 3.2.3.1, 3.2.3.2 y 3.2.3.3, el comportamiento que se destaca a primera vista, es que se cuenta con una fase inicial del algoritmo, en la cual, el mismo se está ajustando a las garantías solicitadas en cada *slice*, para posteriormente, contar con una fase de convergencia y estabilidad.

Para hacer más directo el análisis de las variables de los clientes y *slices*, se propone comparar las variables de los clientes y *slices* entre sí, mediante un punteo de los aspectos más destacados a continuación:

- Como ya se venía hablando en la sección 3.2.2, el *slice* 1 cuenta con una convergencia más rápida a la estabilidad que los *slices* 2 y 3. Esto se puede observar, dado que la fase inicial dura menos en todas las gráficas del cliente 1 y *slice* 1, principalmente, en la grafica G_{n_s}, la cual, indica cuantos paquetes aún necesita transmitir el cliente para garantizar los requisitos de *throughput* que solicita el *slice* 1. Esto también argumenta que, la fase inicial en los *slices* 2 y 3, tiene una mayor duración, refutando así, los resultados de *delay* analizados en la sección 3.2.2.

- Como es de esperarse, la capacidad de transmisión, $\mathbb{C}_{n_s}$, el *bit rate*, R_{n_s}, y la tasa de arribos, A_{n_s}, es mayor en el *slice* 1, dado que provee unas garantías de *throughput* mayor que las de los *slices* 2 y 3. Además, esto también justifica el hecho de que tiene una mayor tasa de descartes de paquetes, D_{n_s}, para mantener los requisitos de *delay* de cola.

- La evolución de las variables de los clientes y *slices* 2 y 3, respectivamente, es muy similar, dado que ambos *slices* garantizan los mismos requisitos y los clientes están fijos en el rectángulo del AP, no dando lugar a variación alguna.

- Si bien el cliente 1 no trasmite en tantos *slots* de tiempo como los clientes 2 y 3 (dado que el U_s es mucho menor en el cliente 1), este cliente 1 transmite durante mucho más tiempo, ya que, el *Consumed Airtime* promedia en los 0,52 segundos, mientras que en los clientes 2 y 3, promedia en los 0,175 segundos. Esto también es justificado, con el hecho de que la capacidad del cliente 1 varía entre 13 y 27 paquetes por *slot* de tiempo, mientras que la capacidad de los clientes 2 y 3, varía entre los 6 y 18 paquetes por *slot*, teniendo incluso *slots* de tiempo en donde se llegan a transmitir 2 paquetes.

- El hecho de que los clientes 2 y 3 lleguen a transmitir pocos paquetes en varios *slots* de tiempos, se justifica con el hecho de que la tasa de arribos, A_{n_s}, es muy parecida a la tasa de descartes, D_{n_s}.

- Ligado al tema de la capacidad de transmisión de cada cliente, $\mathbb{C}_{n_s}$, está la duración de cada intervalo de transmisión. Se puede observar que la duración de los intervalos de transmisión (variable X_s) del cliente 1, es mucho mayor a la duración de los intervalos de transmisión del cliente 2 y 3. Esto tiene sentido con el hecho de que, la capacidad de transmisión, $\mathbb{C}_{n_s}$ (la cual se mide en paquetes por *slot* de transmisión), es mucho mayor en el cliente 1.

- Pese a que la tasa de arribos, A_{n_s}, y la tasa de descartes, D_{n_s}, entre el cliente 1 y los clientes 2 y 3 tengan bastantes diferencias, la cantidad de paquetes en cola (variable *Packets*) en cada *slot* de tiempo, toma valores muy similares entre los clientes. Esto se debe, principalmente, a que las garantías de *delay* entre los clientes de las *slices* son las mismas.

- El hecho de que la suma del *Consumed Airtime* de los tres clientes en cada segundo llegue, a lo sumo, a 0,89 segundos, se argumenta por el hecho de que:
  - Existen momentos en los cuales los clientes no tienen paquetes para transmitir.
  - Las confirmaciones de ACK no se cuentan en la *Consumed Airtime*.
  - Puede existir un cierto *overhead* en las cuentas que genere tiempo de ocio, en el cual, no se transmitan paquetes.

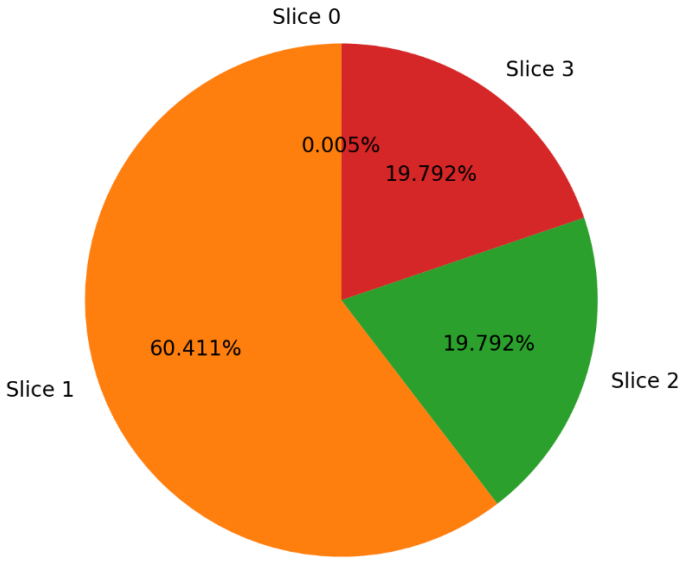
- Los valores del beneficio de cada cliente se mantienen muy estables, dado que las garantías de *throughput* se cumplen siempre. Si las garantías de *throughput*, no se cumplieran, la variable G_{n_s} tomaría valores muy grandes, generando un gran impacto en el cálculo del beneficio.

- La variable Z_{n_s}, la cual, modela una prioridad dinámica en cada cliente y adicionalmente ayuda tomar decisiones de descartes de paquetes, toma valores prácticamente idénticos entre los clientes, dado que las garantías de *delay* entre los clientes de las *slices* son las mismas y, adicionalmente, no se llegan a dar grandes variaciones en los valores de las variables, dado que los clientes se mantienen siempre en su misma posición relativa al AP.

3.2.4. Análisis de la ocupación del AP.

En la gráfica 3.2.4.1 se muestra la ocupación del AP, la cual, muestra qué *slice* estaba ocupando el canal cuando se estaba transmitiendo. Como se observa, se cumplen los límites de capacidad que se habían propuesto en la tabla 3.1.2. Esto está directamente relacionado con el hecho de que se cumplan los requisitos de *throughput* de los tres *slices*.

Una última aclaración sobre esta gráfica, es que el *slice* 0 se utiliza en los primeros segundos para destinar el flujo del control (por ejemplo, el flujo que marca la creación de un nuevo *slice*); algo que se ha pasado por alto en todo este análisis, ya que, no es de relevancia alguna para el estudio de la evaluación del escenario.



Gráfica 3.2.4.1. Ocupación del AP en el escenario estático.

3.3. Escenario dinámico.

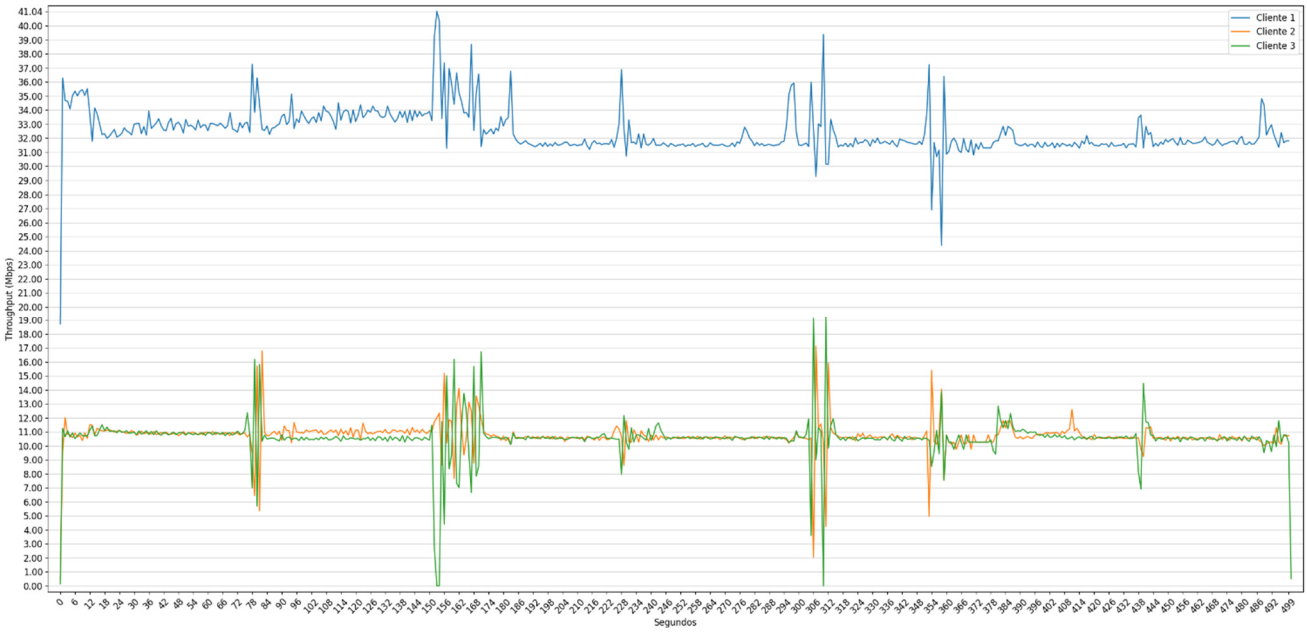
Este escenario cuenta con toda la configuración planteada en la sección 3.1, en adicción a que, los clientes se mueven dentro del rectángulo de 20×20 según los siguientes parámetros configurados:

- Cuando un cliente se mueve, lo hace durante un segundo, en una cierta dirección y a una velocidad elegida de manera uniforme entre uno y dos metros por segundo.
- Después de moverse, se queda quieto el cliente durante dos segundos.
- Los clientes no se pueden mover fuera del rectángulo del AP.

3.3.1. Análisis del *throughput*.

En esta sección, primero se mostrará la evolución del *throughput* de los clientes del escenario, para posteriormente, exponer esos resultados en *boxplots* que expongan más detalles de la variación del *throughput*.

Los resultados de la evolución *throughput*, se pueden encontrar en la gráfica 3.3.1.1. En la gráfica 3.3.1.2. se pueden encontrar los *boxplots* de los resultados del *throughput*.



Gráfica 3.3.1.1. Resultados del *throughput* de los clientes en el escenario dinámico.

En la gráfica 3.3.1.1, se observa que, al igual que en el escenario estático, existe una etapa de convergencia del algoritmo (la cual dura unos pocos segundos al inicio), pero, una vez pasada la etapa de convergencia, la esperada etapa de estabilidad que estaba presente en el escenario estático, no se aprecia del todo en el escenario dinámico; se ven bajadas y subidas en lo que viene a ser el nivel "estable" del *throughput*, las cuales ocurren simultáneamente en los tres clientes de los *slices*. Estas variaciones en el *throughput*, se deben a que los clientes se mueven en ese instante, lo que provoca que la capacidad de

transmisión varié, haciendo que, el algoritmo tenga que "recomponerse" y volver a estabilizar los valores de las variables. A pesar de todo, esto es un buen indicativo de que el algoritmo sabe actuar frente a una variación o cambio en el medio. Además, se observa que las garantías de *throughput* siempre se intentan al margen de los requisitos estipulados en cada *slice*, mostrando que, cuando un cliente apenas puede transmitir durante un segundo, en los siguientes segundos el algoritmo hace que ese cliente transmita durante más tiempo, para así, garantizar el requisito de *throughput* promedio que necesita ese cliente del *slice*, en cuestión.

Por otro lado, cuando solo se cuenta la carga útil de los paquetes transmitidos para el cálculo del *throughput* promedio, se obtienen los siguientes resultados:

- Cliente 1: 29,9218 Mbps.
- Cliente 2: 9,90359 Mbps.
- Cliente 3: 9,73187 Mbps.

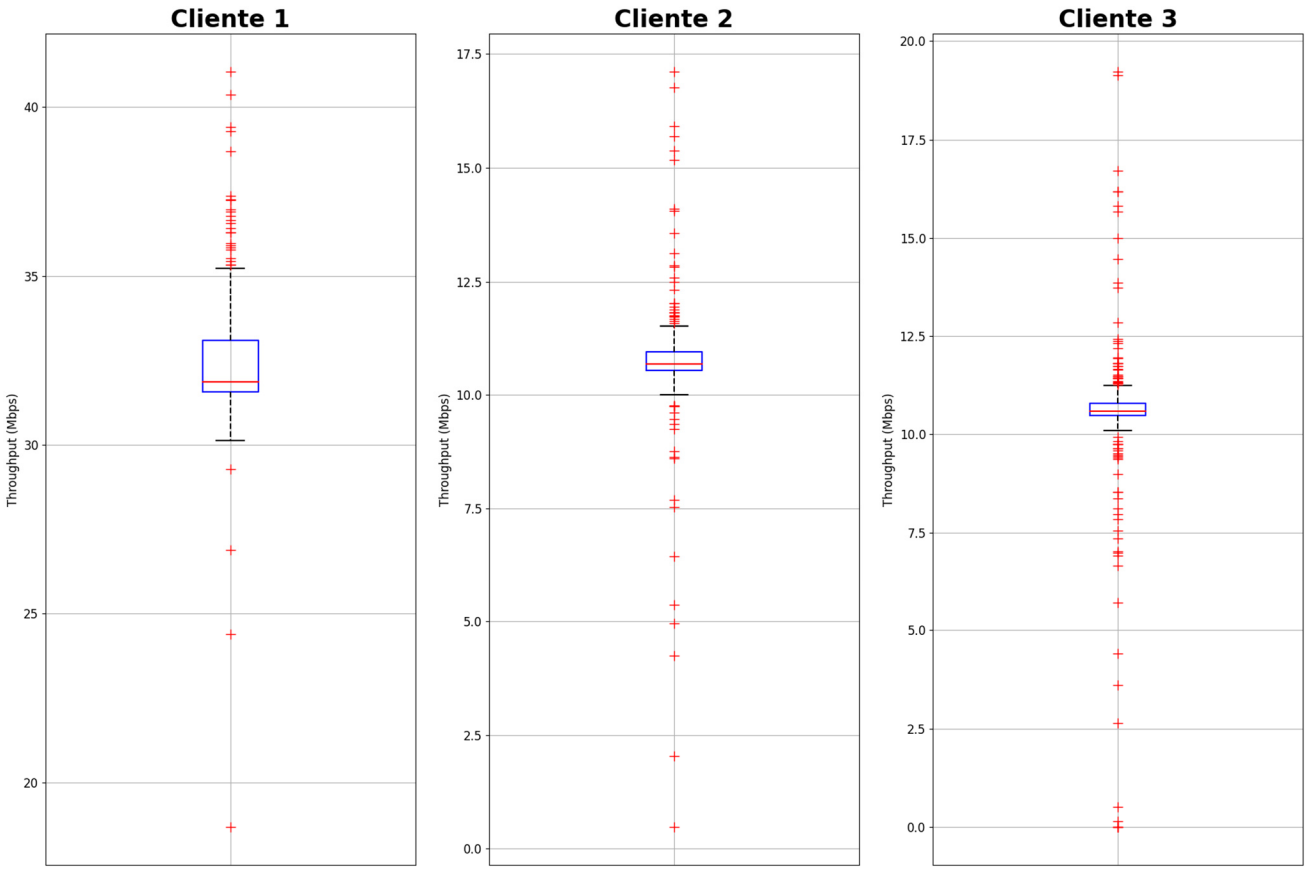
Como se puede observar, estos resultados de *throughput* promedio están mínimamente por debajo de las garantías de *throughput* que provee cada *slice*. La explicación de esto tiene dos motivos:

1. Todos los clientes se mueven a la vez.
2. El promedio se deja afectar mucho por los datos atípicos.

Sobre el punto 1), hay que destacar que, si todos los clientes no se movieran a la vez, se podría tomar la decisión de mejorar el *throughput* de los clientes que se mantienen fijos, hasta que los clientes que se movieron se vuelvan a quedar quietos. Esto aumentaría el *throughput* promedio de los clientes que permanecieron inmóviles desde un principio, pero este *throughput* promedio se volvería a balancear, dado que se les dejaría transmitir durante más tiempo a los clientes que se terminaron de mover. Esta estrategia no se puede aplicar en este escenario, dado que todos los clientes se mueven a la vez, provocando que, ese tiempo de transmisión, no se aprovechado de la manera más eficiente (ya que el algoritmo se tiene que reajustar para saber cuál es la forma de cumplir con las garantías de los *slices*).

Con este análisis, se pudo comprobar lo que se venía concluyendo del escenario estático, el hecho de que los clientes puedan moverse en el escenario, provoca "anomalías" en la evolución del *throughput*.

Finalmente, en la gráfica 3.3.1.2 se pueden observar los *boxplots* de los valores que se muestran en la gráfica 3.3.1.1, corroborando la existencia de una mayor variabilidad en el *throughput* (las cajas de los *boxplots* son más largas que las del escenario estático), al igual que, una mayor presencia de datos atípicos, provocados por el movimiento de los clientes, en adición, a la fase inicial en la que el algoritmo se está ajustando a las necesidades de cada *slice*.



Gráfica 3.3.1.2. *Boxplots* del *throughput* de los clientes en el escenario dinámico.

3.3.2. Análisis del *delay*.

En esta sección, primero se mostrarán los histogramas de la cantidad de paquetes con *delay* de cada cliente, para posteriormente, graficar esos histogramas como *boxplots* y así, obtener otras métricas de lo que está sucediendo en la simulación. En adición a eso, también se mencionará la proporción de paquetes cuyo *delay* de cola no cumpla con la cota máxima establecida para el algoritmo, en los *slices* en lo que esto ocurra.

Los histogramas de la cantidad de paquetes con *delay* de cada cliente, se pueden encontrar en las gráficas 3.3.2.1, 3.3.2.2 y 3.3.2.3. Los *boxplots* de los histogramas se pueden encontrar en la gráfica 3.3.2.4.

Gráfica 3.3.2.1. Histograma de la cantidad de paquetes con *delay* del cliente 1 en el escenario dinámico.

Como se observa en la gráfica 3.3.2.1, para el cliente 1 no se cumplen las garantías de *delay* de cola acotado. A pesar de que el cliente 1 en el escenario estático era el más beneficiado en términos del *delay* de cola acotado, el movimiento que conlleva la formulación del escenario dinámico, provoca grandes variaciones en los cálculos del algoritmo, que obligan a mantener en la cola por más tiempo a los paquetes. Las consecuencias que provoca el movimiento de los clientes en el algoritmo, son los siguientes:

1. La duración del *slot* de transmisión es mayor.
2. Hay más paquetes para transmitir (provoca que los paquetes tarden más en salir de la cola, ver variable *Packets*).
3. El *bit rate* de transmisión mientras el cliente se mueve puede disminuir (provocando que los paquetes tarden aun más de salir de la cola).

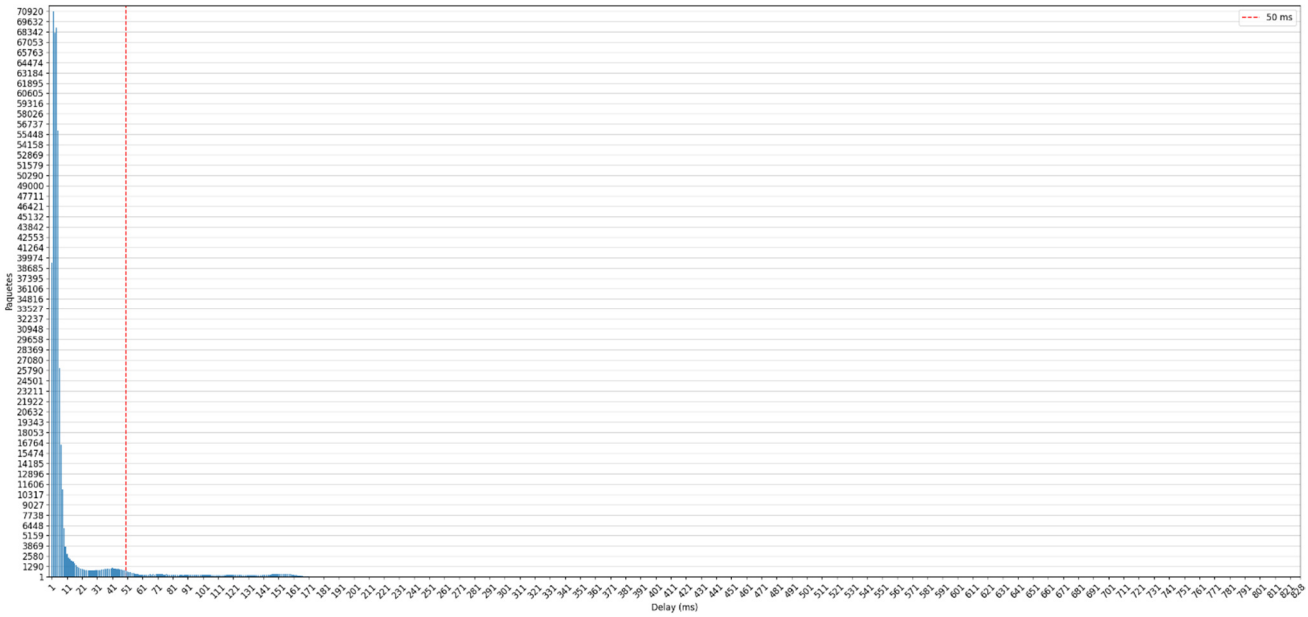
El punto 1) ya se venía explicando en el escenario estático. En el escenario dinámico tiene una influencia aun mayor, ya que, como se observa en la sección 3.3.3, las variables X_s de los distintos *slices*

toman valores de hasta 5, lo que provoca que la duración del *slot* de transmisión llegue a ser de hasta 12,5 *ms*.

Ligado con el punto 1), se encuentra el punto 2). Al durar más el *slot* de transmisión, se acumulan más paquetes en la cola del AP, de los clientes que no transmiten. Esto directamente impacta en el *delay* de cola de los paquetes que tendrán los paquetes a lo último de la cola. Además, por más que el algoritmo se adapte para descartar más paquetes (debido a que el *slot* de transmisión duró más), esto no significa que, ante una ráfaga de arribos de nuevos paquetes, el algoritmo vaya a provocar más descartes; el algoritmo solo incrementa la máxima cantidad de descartes posibles, si la duración del *slot* de transmisión fue mayor, no si la cantidad de arribos incrementó (provocado porque los clientes se dejaron de mover) de un momento a otro sin que el tamaño del *slot* de transmisión variara.

Por otro lado, el punto 3) no necesita mucha explicación. El *delay* de cola se ve directamente afectado por el *delay* de transmisión, cuanto más grande sea mayor el *delay* de transmisión, más tarda un paquete de salir de la cola del AP y mayor es el *delay* de cola, por ende.

Por último, de los 1.352.895 paquetes representados en el histograma, 43.031 paquetes tienen *delay* de cola mayor a 50 *ms*. Es decir, un 3,1806% de los paquetes no cumple las garantías de *delay*.

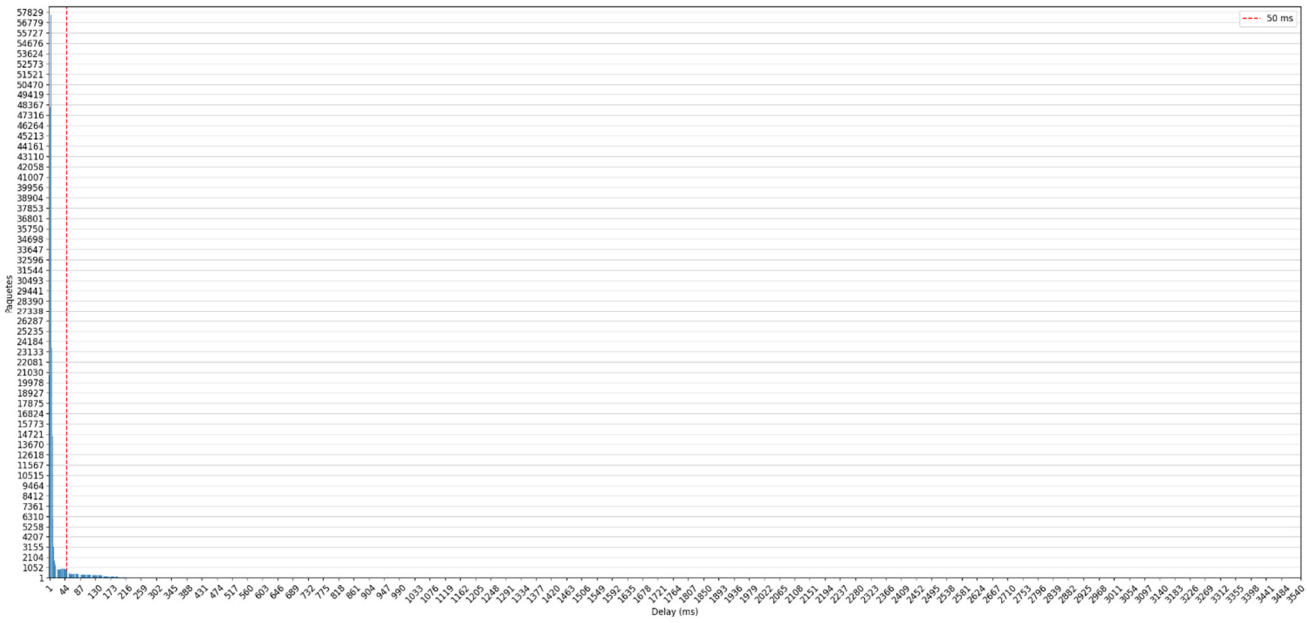


Gráfica 3.3.2.2. Histograma de la cantidad de paquetes con *delay* del cliente 2 en el escenario dinámico.

Las conclusiones obtenidas de la gráfica 3.3.2.2 para el cliente 2 son las mismas que se vieron para el cliente 1. El movimiento de los clientes está provocando que el *delay* de cola no sea acotado. La explicación del porque el *delay* de cola es mejor en el *slice* 1, se debe a que en el mismo se garantiza un mayor requisito de *throughput*, haciendo que el *delay* de transmisión (y, por ende, el *delay* de cola) sea menor. Adicionalmente, se puede observar en la gráfica 3.3.3.2 de la sección 3.3.3 que el tamaño de la variable Z_{n_s} del cliente 2 toma valores mayor a la variable Z_{n_s} del cliente 1 (gráfica 3.3.3.2), esto está

justificando que no se están descartando paquetes lo suficientemente rápido y no se están garantizando los requisitos de *delay* de cola acotado.

Por último, de los 447.775 paquetes representados en el histograma, 35.500 paquetes tienen *delay* de cola mayor a 50 *ms*. Es decir, un 7,9280% de los paquetes no cumple las garantías de *delay*; una proporción mayor comparada con la del *slice* 1, la cual, se justifica por la evolución de la variable Z_{n_s} ya mencionada.



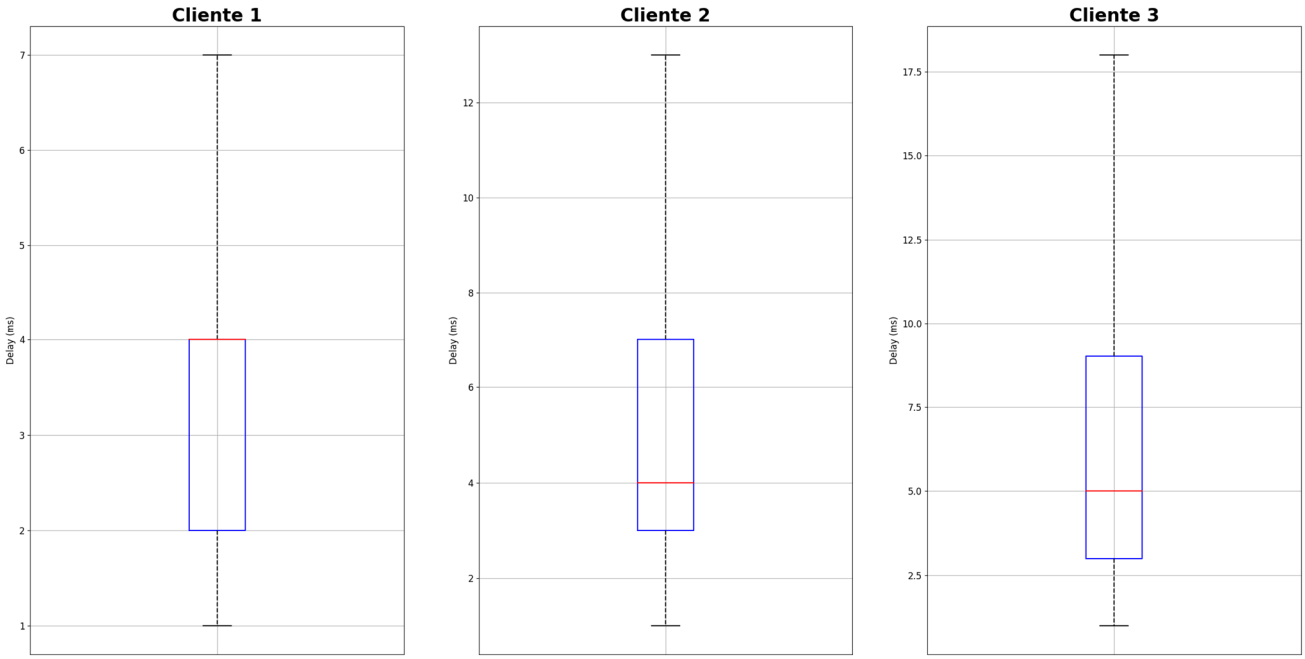
Gráfica 3.3.2.3. Histograma de la cantidad de paquetes con *delay* del cliente 3 en el escenario dinámico.

Las conclusiones obtenidas de la gráfica 3.3.2.3 para el cliente 3 son las mismas que se vieron para el cliente 2. Es decir, el movimiento de los clientes está provocando que el *delay* de cola no sea acotado y no se están descartando suficientes paquetes, dado que la variable Z_{n_s} del cliente 3 (gráfica 3.3.3.3 de la sección 3.3.3) está tomando valores pico muchos más grandes que la variable Z_{n_s} de los clientes 2 y 3. Esto justifica el inusual pico en *delay* de cola que se puede llegar a ver en la gráfica 3.3.2.3.

Por último, de los 440.020 paquetes representados en el histograma, 45.758 paquetes tienen *delay* de cola mayor a 50 *ms*. Es decir, un 10,399% de los paquetes no cumple las garantías de *delay*; una proporción mayor comparada con la del *slice* 1 y 2, la cual, se justifica por la evolución de la variable Z_{n_s} ya mencionada.

Como última observación sobre el movimiento de los clientes, se puede ver que ahora la cantidad de paquetes transmitidos de los clientes 2 y 3 varia mucho más (antes, la cantidad de paquetes transmitidos era prácticamente idéntica). En particular, en el cliente 3 obtuvo menos paquetes transmitidos; esto explica por qué también el *throughput* del cliente 3 es menor al del cliente 2.

Finalmente, en la gráfica 3.3.2.4 se presentan los *boxplots* de los *delays* de los clientes. En la misma, se puede observar que aun gran parte de los paquetes mantiene un *delay* de cola muy por debajo de la cota establecida, pese a eso, se aprecia una mayor variación en cuanto al *delay* de cola.

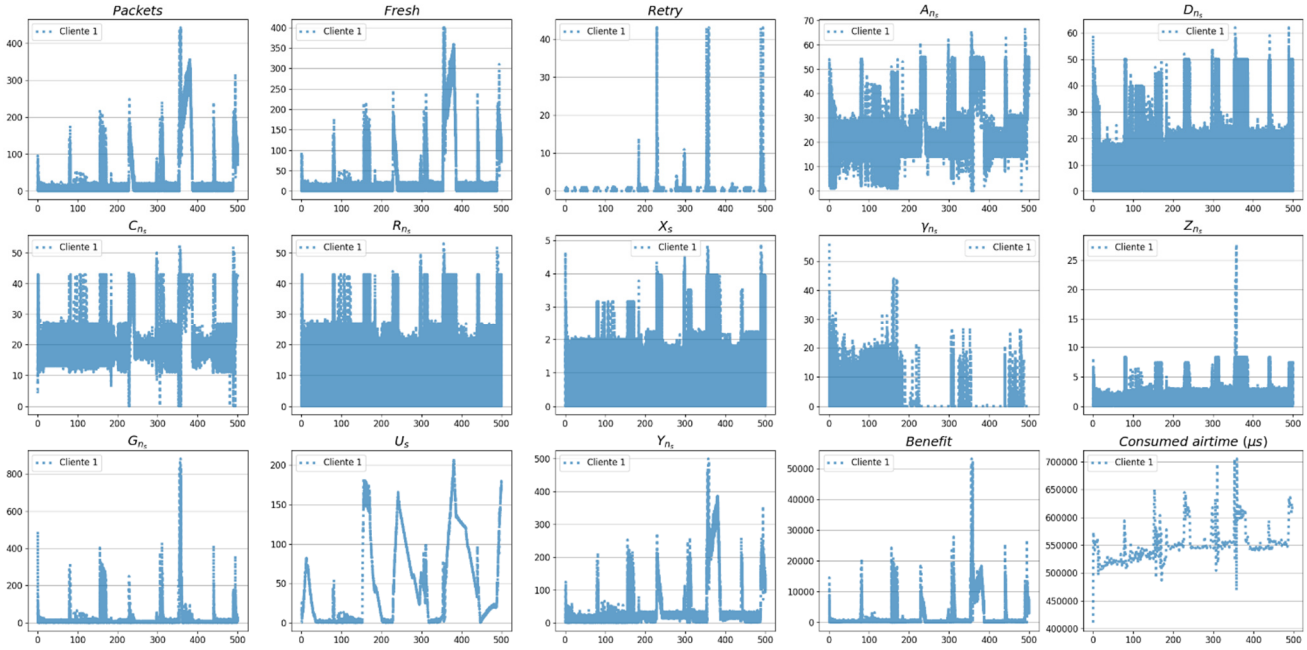


Gráfica 3.3.2.4. *Boxplots* de los *delays* de los clientes en el escenario dinámico.

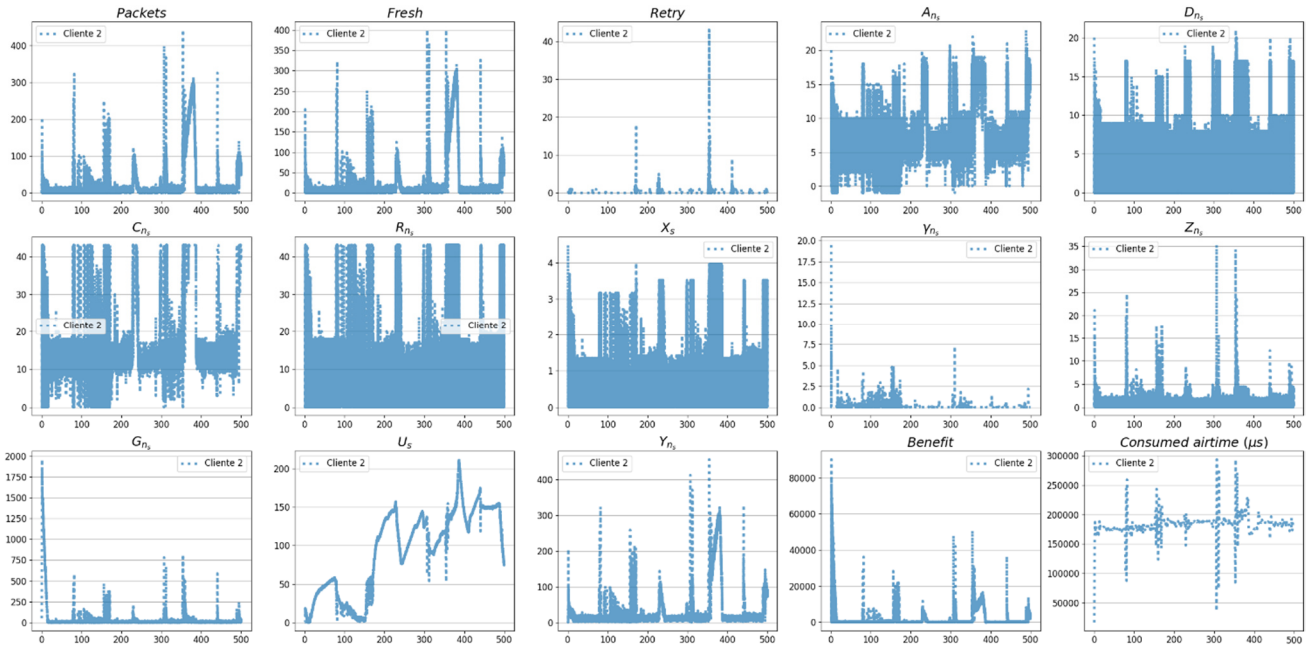
3.3.3. Análisis de las variables.

En esta sección, se mostrará la evolución de las variables más importantes del algoritmo en cada cliente y *slice*, así como se expondrá la evolución de otras métricas, como lo son, el beneficio de cada cliente en cada intervalo de transmisión y el *airtime* consumido en cada segundo de la simulación.

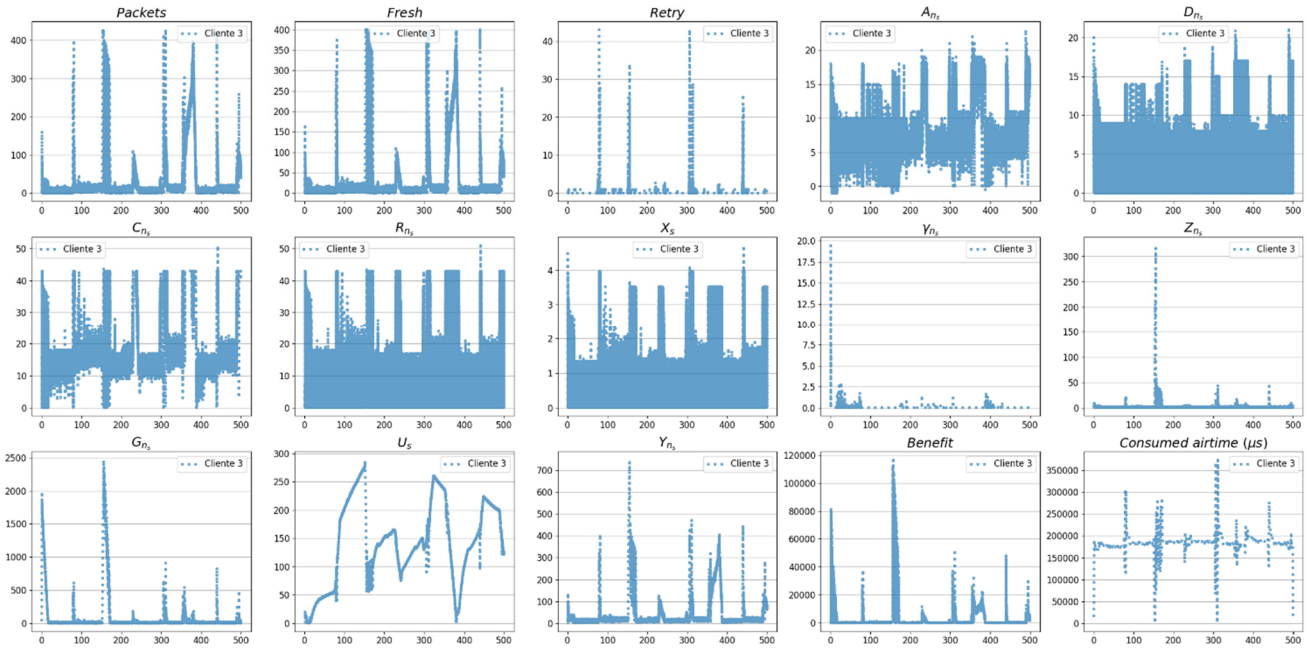
La evolución de las variables de cada cliente y *slice*, se pueden encontrar en las gráficas 3.3.3.1, 3.3.3.2 y 3.3.3.3. Recordar que, sin ser por las últimas dos gráficas de cada cliente y *slice*, todas las variables están expresadas en paquetes por *slot* de transmisión.



Gráfica 3.3.3.1. Evolución de las variables del cliente 1 y *slice* 1 en el escenario dinámico.



Gráfica 3.3.3.2. Evolución de las variables del cliente 2 y *slice* 2 en el escenario dinámico.



Gráfica 3.3.3.3. Evolución de las variables del cliente 3 y *slice* 3 en el escenario dinámico.

Como se observa en las gráficas 3.2.3.1, 3.2.3.2 y 3.2.3.3, a diferencia de lo visto con el escenario estático, no se puede observar una fase de convergencia y estabilidad de las variables. De nuevo, esto se debe a la movilidad presente en los clientes del AP.

Muchos de los aspectos estudiados para el análisis de estas variables ya fueron expuestos en el análisis hecho en la sección 3.2.3. En este análisis, solo se expondrán las nuevas características presentadas:

- Se observan mayores picos en los valores de todas las variables de los clientes. Esto se debe, a que una vez los clientes dejan de moverse, el algoritmo tiene que reajustar las variables haciendo que los clientes transmitan durante más tiempo o haciendo que los clientes descarten más paquetes.

- Los momentos pico de la variable G_{n_s} de los clientes, tiene sentido con el momento en el cual las garantías *throughput* decaen. A pesar de todo esto, estos picos son muy momentáneos dado que el algoritmo vuelve a reajustarse para garantías los requisitos de *throughput* promedio. Esto también explica los picos momentáneos en la gráfica del cálculo del beneficio (*Benefit*), al igual que en las gráficas $\mathbb{C}_{n_s}$, R_{n_s}, X_s y U_s.

- Los picos momentáneos de la variable G_{n_s}, además, se relacionan directamente con el tamaño de la cola de paquetes (*Packets*).

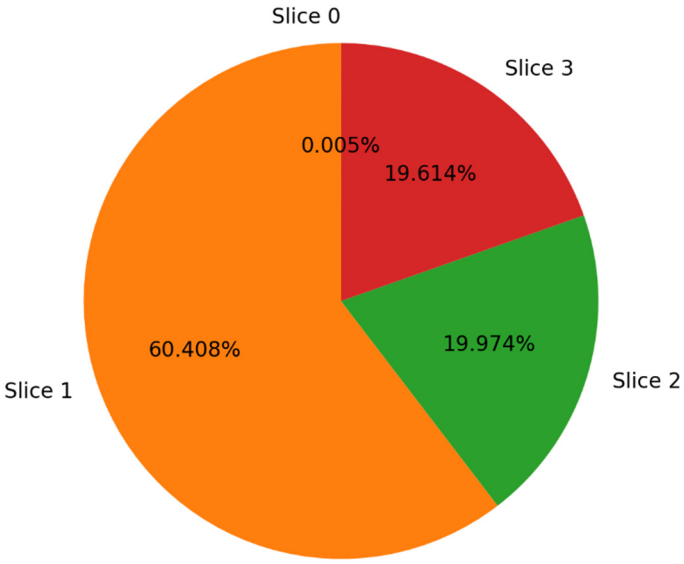
- Los picos de decadencia de la variable U_s tienen sentido con los picos de decadencia de la gráfica *Consumed Airtime* (μs), ya que esta última indica cuánto tiempo estuvo transmitiendo un cliente durante cada segundo.

- La evolución general de la gráfica de la tasa de arribos, A_{n_s} está directamente relacionada con la evolución general de la gráfica de la cantidad de descartes, D_{n_s}. Ya que, cuando arriban más paquetes, a la vez, se tienen que descartar más paquetes para asegurar el *delay* de cola acotado.

3.3.4. Análisis de la ocupación del AP.

En la gráfica 3.3.4.1 se muestra la ocupación del AP, la cual, muestra qué *slice* estaba ocupando el canal cuando se estaba transmitiendo. Como se observa, se cumplen los límites de capacidad que se habían propuesto en la tabla 3.1.2. Esto, en un principio, puede ser algo contraintuitivo, dado que en el escenario hay mucha variación en los cálculos, pero no hay que olvidar, que esta grafica está directamente relacionada con el hecho de que se cumplan los requisitos de *throughput* de los tres *slices*.

Se mantienen las mismas aclaraciones sobre la gráfica, hechas en la sección 3.2.4.



Gráfica 3.3.4.1. Ocupación del AP en el escenario dinámico.

4. Conclusiones

Como resumen general de este proyecto, se logró:

- Lograr comprender los conceptos de *Slicing*, QoS *Slicing*, QoS *Slicing* el sentido de QoS *Slicing* en WiFi y la solución planteada en la tesis.
- Entender las características del sistema cíber – físico planteado.
- Entender la implementación de las Capas 2 y 3 de WiFi en el simulador NS3.
- Realizar un análisis y diseño de los cambios necesarios para implementar la solución propuesta en NS3.
- Implementar esos cambios y realizar una evaluación de la solución en un escenario simulado.

5. Trabajo futuro

Algunas de los temas que resta por ver son:

- Probar el algoritmo con un escenario más grande y complejo. Esto es algo totalmente necesario para comprobar a fondo la correctitud del algoritmo.
- Realizar los ajustes necesarios para que funcione con ese escenario.
- Realizar un análisis de las diferentes optimizaciones en todos los cálculos que realiza el algoritmo y optimizar esos cálculos.
- Evaluar el uso de *Machine Learning* para encontrar los mejores parámetros de ajustes para un escenario y momento de la simulación dados. Esto es porque el funcionamiento del algoritmo, es un momento determinado, depende mucho de los parámetros de ajuste que se configuran "a mano". Entonces, estaría bien que esos parámetros se pudieran configurar de manera automática

6. Bibliografía

[1] Richart, M. (2019, Noviembre). Resource Allocation and Management Techniques for Network Slicing in WiFi Networks. Montevideo, Montevideo, Uruguay.

[2] NS3. (2021). *NS3*. Retrieved from ns-3 Network Simulator: https://www.nsnam.org