TAREA 1

Entrega: 8 de Mayo de 2024

Resultados de Aprendizaje

- Crea modelos de simulación para procesos físicos de redes u otro que efectivamente capturen el funcionamiento de los procesos a simular
- Crea programas computacionales que evalúan numéricamente los modelos de simulación obteniendo resultados que se ajustan a resultados teóricos conocidos
- Utiliza técnicas de visualización de datos en lenguajes de propósito general que permitan apreciar cuantitativamente las variaciones de las variables observadas

Problema

Servidor de redes sociales

Considere el contexto de un servidor (o conjunto de servidores) que contienen el servicio de una red social exitosa (Instagram, Telegram, Whatsapp o cualquier otra). Como es exitosa, atiende una gran cantidad de personas (miles o más). Sin embargo, este servidor tiene una capacidad limitada definida (**C**). ¿cómo puedo medir la calidad de servicio ofrecida en estas condiciones de usuarios considerados como infinitos? ¿Puedo entonces, dimensionar el valor que debe tener la capacidad C para ofrecer una calidad de servicio mínima aceptable?

- Suponga que existe un número tan grande de usuarios que utilizan el servicio ofrecido al servidor que se consideran como **infinitos**.
- Estos usuarios desean utilizar el servidor cuya **capacidad** es igual a **C=45** unidades de cómputo del servidor (una combinación de CPU, GPU, RAM y Disco Duro).
- Los procesos de llegadas y salidas son procesos aleatorios dados por un proceso Poisson (donde los tiempos entre llegadas y de servicio se pueden representar por una distribución exponencial), con sus respectivas tasas media de llegadas de usuario $\lambda = 300$ y tasa media de servicio en el enlace $\mu = 10$,
- No existe cola de espera (sin buffers).

¿Cuál es la probabilidad de que un usuario no pueda conectarse por congestión en el enlace?

Para evaluar la probabilidad de bloqueo del enlace en cuestión, programe un simulador en Python, el cual represente el modelo definido del sistema. Defina, inicialmente los módulos necesarios para su programación, aprovechando las capacidades de Python para modularizar la programación, y así poder reutilizar código en otras simulaciones.

El simulador debe incluir lo siguiente:

1. Definición de variables, estados y sistema (El código debe ser autocontenido y comentado con las explicaciones pertinentes).
2. Estado inicial del sistema. (No puede haber salidas si no hay llegadas)
3. Gestión de los eventos en la línea de tiempo, para así reconocer si los eventos son salidas o llegadas.
4. Como criterio de parada utilice la definición de un número finito de llegadas de usuarios (denominado LLEGADAS).

5. Cálculo de las métricas al finalizar la simulación. (probabilidad de bloqueo por congestión)

Declare un método para definir el avance de tiempo (FEL). En este sentido puede aprovechar las funciones de Python para crear y utilizar una línea de tiempo, o cree un medio propio para representar el avance de tiempo de la simulación. El simulador debe poder ejecutarse desde Jupyter Notebook, y visualizarse en un sitio web creado con Voila, para desplegar los datos y su posterior análisis, basándose en las librerías de visualización de datos de Python (Idee como hacerlo).

Como salida de cada simulación, el simulador debe obtener:

- La probabilidad de bloqueo obtenida por el simulador, en base al número de llegadas totales de usuarios ejecutadas (**LLEGADAS**).
- Una lista con las probabilidades de bloqueo obtenidas cada **LLEGADAS/100** llegadas de usuarios ejecutadas.

El simulador se debe ejecutar 8 veces, donde cada una de ellas debe ejecutarse con 10^i llegadas de usuarios (criterio de parada), donde i tiene el valor de 2,3,4,5,6,7,8 y 9. En base a lo anterior se deben incluir los siguientes gráficos utilizando las bondades de Python y Jupyter Notebook.

1. Un gráfico que indique las probabilidades de bloqueo (eje Y) por el número de llegadas totales de usuarios ejecutadas en cada simulación (eje X).
2. Ocho gráficos que muestren el avance de la probabilidad de bloqueo según el número de llegadas ejecutadas (Lista obtenida por cada simulación), para cada simulación hecha.

La asignación es individual, donde se debe entregar por correo electrónico a su queridísimo profesor, un archivo comprimido denominado de la siguiente forma "TEL341 Nombre Apellido T1". Este archivo comprimido debe contener todo lo necesario para su evaluación, es decir, el Notebook de Jupyter/Voila con los archivos asociados para su ejecución, el código en Python y los gráficos obtenidos.

Patricia Morales Calvo
patricia.morales@usm.cl