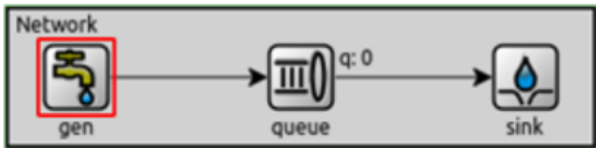
Transporte

Modelo de Colas

Se entrega como kickstarter un modelo de colas que consta de un generador (gen), una cola
(queue) y un destino (sink) conectados en una red Network (definida en el archivo .ned).



En el archivo omnetpp.ini se configura una simulación de 200s donde gen crea y transmite paquetes con intervalos dados por una distribución exponencial de media configurable, y queue
es capaz de atenderlos bajo una misma distribución.
El comportamiento de cada módulo se especifica en su respectiva clase en C++, declarada y
definida en un único archivo de código (Generator.cc, Queue.cc y Sink.cc). En particular, el módulo Sink toma métricas (.vec y .sca) de la demora de entrega de los paquetes.

Tarea Análisis

El modelo de colas ignora nociones de capacidad (tasa de transferencia de datos y memoria de
buffers). La primera tarea es modificar y extender el proyecto para hacer un análisis del impacto de estos parámetros en el tráfico de red.
Debido a que es la primera interacción con Omnet++, se provee una guía paso a paso, con snippets de código, que deberá ser complementada con el material de clases y consultas al manual de la herramienta.

Modificaciones en network.ned

El nodo Generator y Sink pasarán a ser parte de módulos compuestos denominados nodeTx
y nodeRx. Deberán contener un buffer de transmisión y recepción que se podrá instanciar con
el módulo de cola existente.

Las conexiones deberán configurarse con tasas y demoras de transmisión para dos casos de
estudio específicos:

- Caso de estudio 1:
    - NodeTx a Queue: datarate = 1 Mbps y delay = 100 us
    - Queue a NodeRx: datarate = 1 Mbps y delay = 100 us
    - Queue a Sink: datarate = 0.5 Mbps
- Caso de estudio 2:
    - NodeTx a Queue: datarate = 1 Mbps y delay = 100 us
    - Queue a NodeRx: datarate = 0.5 Mbps y delay = 100 us
    - Queue a Sink: datarate = 1 Mbps

El módulo gen deberá tomar como parámetro el tamaño del paquete en Bytes (packetByteSize) que tendrá un valor de 12500 Bytes. Los mismos serán generados en

intervalos de tiempos exponenciales con media de 100 ms.
Las queue serán limitados en tamaño de buffer (bufferSize), expresado en cantidad de paquetes. Se configurará con un valor máximo de 200, salvo la cola del nodo transmisor que se
dejará arbitrariamente alta.
Dentro del omnetpp.ini los parámetros agregados los tiene que poner aquí, en sección parámetros del network.ned correspondiente a cada nodo y en los archivos cc de cada tipo de
nodo para ser reconocidos. ej: buffer size debe agregarse aqui, en el modulo queue.cc y en queue de network.ned

Modificaciones en clases de C++

Los objetos cMessage no tienen parámetros de tamaño en Bytes. Se deberá cambiar por objetos cPacket y configurar su tamaño en base al parámetro de configuración correspondiente.
El tiempo de servicio de Queue deberá basarse en la duración de la transmisión del paquete una vez ya encolado (tiene en cuenta la tasa de datos configurada en la conexión).
Se deberá modificar el código de Queue para que controle que el tamaño del buffer no sobrepase el límite configurado. De estar lleno, el nuevo paquete se ignorará y se borrará.
Se deberán agregar nuevas métricas para el análisis en Queue, una que mida la cantidad de
paquetes en el buffer, y otra que mida la cantidad de paquetes descartados por buffer saturado.
class Queue: public cSimpleModule // agregar en la definicion de la clase

Tarea Diseño

La segunda tarea es diseñar un sistema de control de flujo y congestión (entre el destino y el
generador) de manera que se evite la pérdida de datos por saturación de buffers.

Modificaciones en network.ned

Se deberá agregar un canal de retorno desde el nodeRx al nodeTx para que el receptor pueda
acusar información que regule la tasa de transmisión (feedback). Así, las queues evolucionarán
a un nuevo módulo denominado transportTx y transportRx

Modificaciones en clases de C++

La modificación más importante será generar las clases TransportTx y TransportRx en base a la clase existente Queue. TransportRx deberá ser capaz de enviar información sobre el estado de su buffer a TransportTx para que la ésta regule su flujo de transferencia.
Considere generar un nuevo tipo de paquete (usando la definición de paquetes packet.msg) donde pueda agregar campos necesarios para su protocolo como “tamaño de buffer actual”, “bajar velocidad de transmisión”, “transmitir siguiente paquete”, etc.
Puede usar la función msg->setKind() y msg->getKind para diferenciar entre paquetes que son de datos y paquetes de control (feedback).