SimNet: diseño

A discrete event, system level wireless network simulator and software framework

Pablo Belzarena, Víctor González Barbone

Instituto de Ingeniería Eléctrica
Facultad de Ingeniería
Universidad de la República Uruguay

Diciembre 2023

Contenido

SimNet, presentación del diseño y estado de avance.

- Antecedentes y objetivos.
- Arquitectura.
- Escenario de simulación.
- Simulación por eventos.
- Cola de paquetes y manejo de transmisión.
- Integración con PyWiCh.
- Próximos objetivos.

Antecedentes y objetivos

Antecedentes:

- varios proyectos anteriores (tesis, fin de carrera).
- realizaciones de prototipo.
- difíciles de extender a diferentes modelos.

Objetivos:

- buen diseño de arquitectura, modularización, bajo acoplamiento, alta cohesión; un "software framework".
- eficiencia: estructuras de datos y su manejo.
- integrar desarrollos de proyectos anteriores.
- integrar e interactuar (datos de archivo) con simulador de canal PyWich: ejemplo y prueba de concepto de extensión.
- orientado a la extensión: funcionalidad, protocolos, algoritmos.
- mantener la evolución del software con estos objetivos.

Diagrama de paquetes

Biblioteca libsimnet:

- núcleo del sistema.
- se pueden sobreescribir.
- solo desarrolladores SimNet.

Extensiones:

- para distintos modelos de simulación.
- agregan clases, sobreescriben las del núcleo.
- en SimNet, dentro del paquete `extensions`, o
- fuera de SimNet, "out of tree".

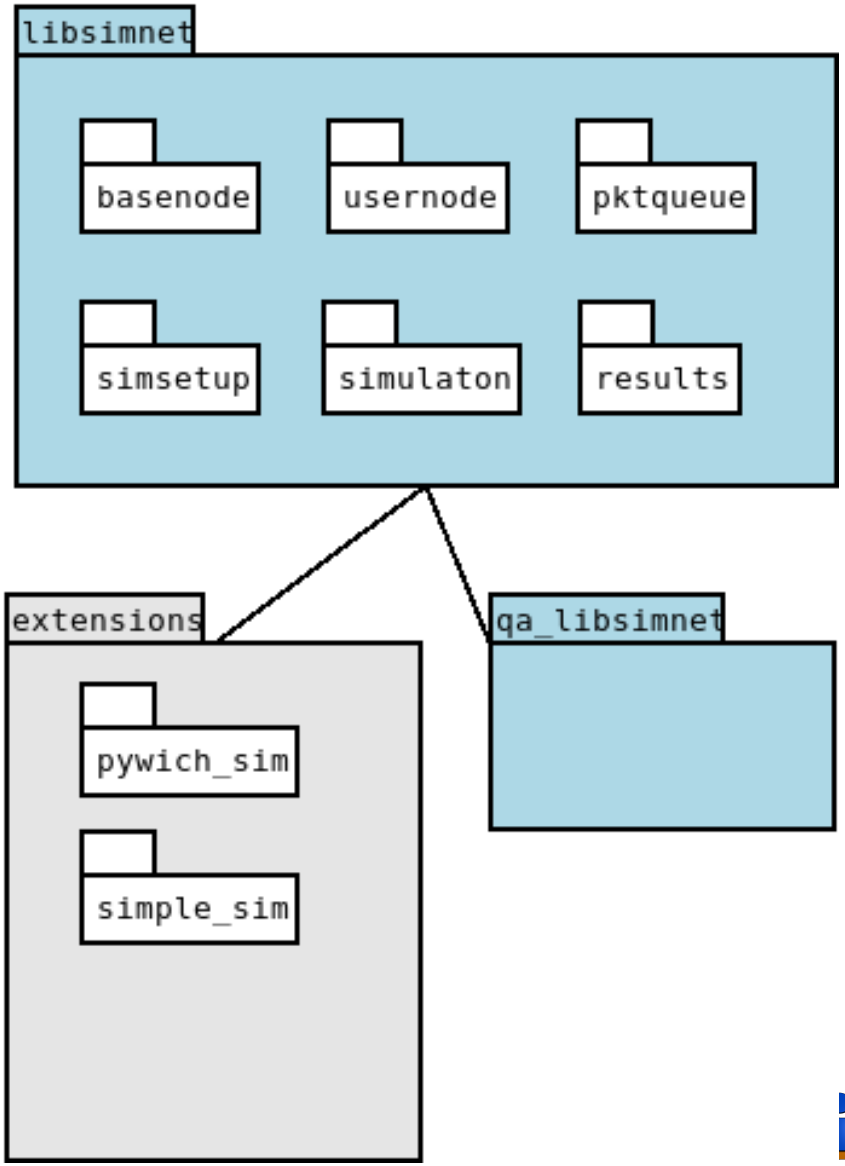


Diagrama de clases

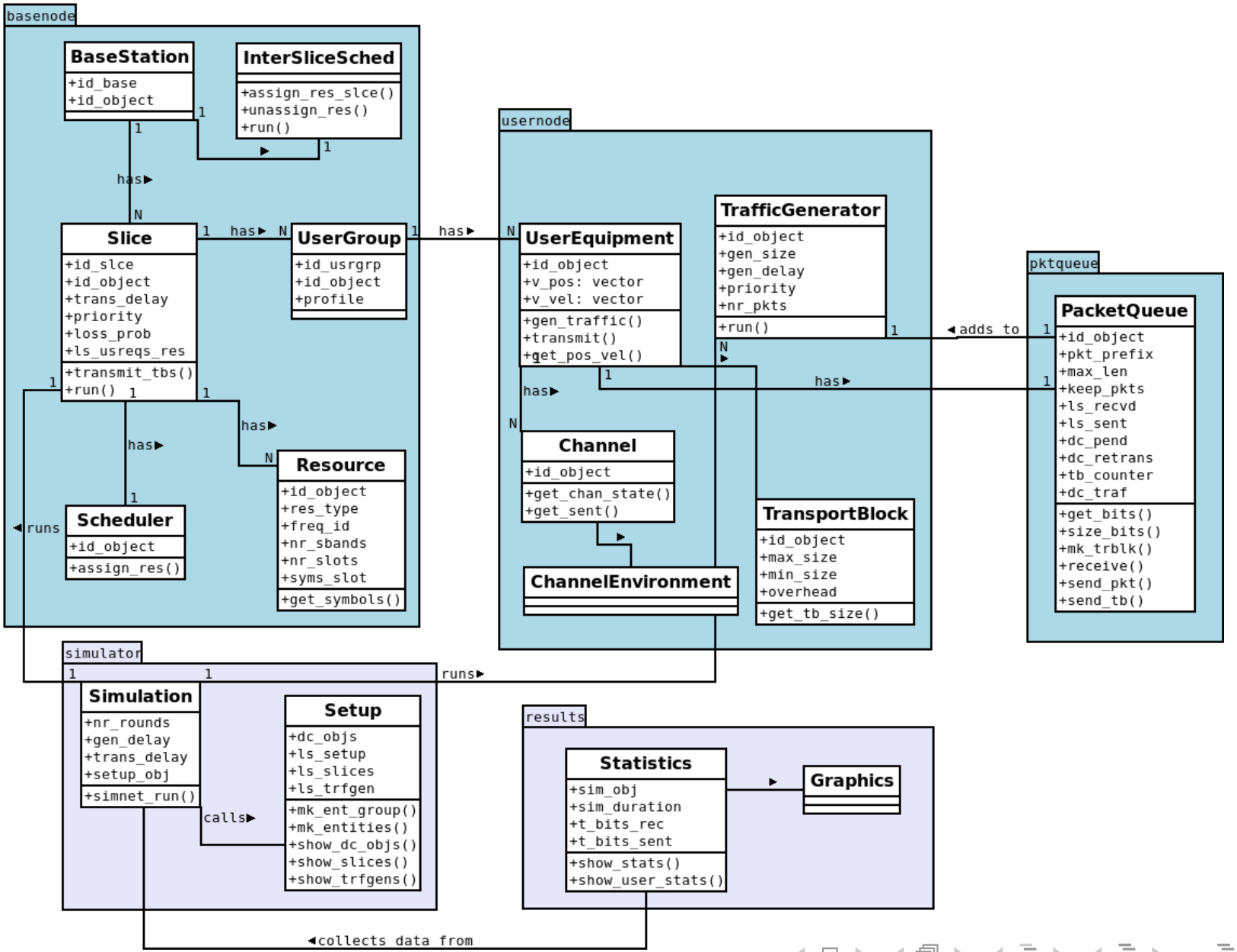
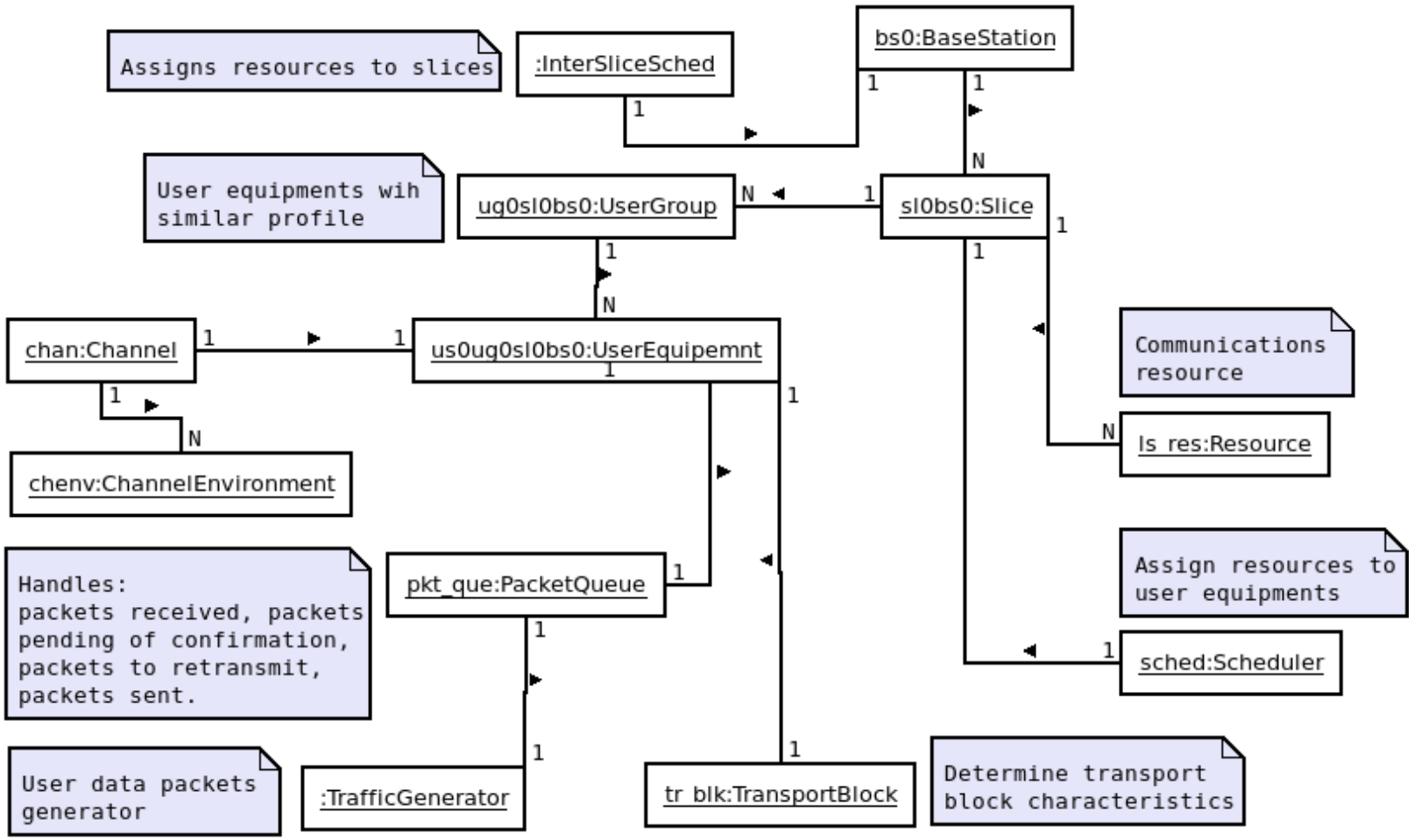


Diagrama de objetos



Estructura de directorios

SimNet, directorio del proyecto. Contiene:

- `libsimnet` : el núcleo, modifican solo desarrolladores.
- `extensions/xxxx_sim` : clases agregadas y sobreescritas de `libsimnet` para simulador Xxxx. Incluye setup de escenario y código de prueba.
- sandbox : códigos de prueba de algoritmos y estructuras.
- docs : documentación formato MD (markdown language), usados en Gitlab, Github, etc.
- html : documentación del código, con pydoctor.

Un nuevo modelo de simulación puede implementarse:

- fuera de la estructura de directorios de SimNet, o
- dentro del directorio `extensions`.

Escenario de simulación

Concentra en un módulo la definición del escenario de simulación, para poder probar fácilmente distintos escenarios.
Clase `simsetup.Setup`, crea escenario en base a lista de items:

```
[ class_name, nr_items, attach_to, ls_pars, ... ]
```

- define objetos, cantidad y a qué entidad se enganchan.
- permite fijar parámetros para las distintas entidades.
- devuelve dos listas: `ls_slices` (slices), y `ls_trfgen` (generadores de tráfico).

En el módulo de usuario `qa_simulator`, las funciones:

- `setup_sim`: fija parámetros, crea objeto Setup.
- `run_sim`: recibe objeto Setup, crea objetos Simulation y Statistics, corre la simulación.

Simulación por eventos

Modos de simulación: alternativas, ventajas e inconvenientes:

- simuladores genéricos, e.g. simpy, muchas funcionalidades.
- threads: muchos, overhead, coordinación entre sí.
- procesamiento secuencial con unidades de tiempo.
- eventos, en una cola de prioridad.

Clase de Python `queue.PriorityQueue`:

- inserción: `put(evento)`, ubica en lista ordenada.
- extracción: `get()` devuelve el evento más próximo.
- inserción en O(log(n)), extracción en O(1).
- el evento debe ser ordenable; en SimNet, una lista `[ time_t, priority, id_action]`

Eventos en SimNet:

- cada proceso periódico agenda su próximo evento.
- procesos: generar tráfico, transmitir, repartir recursos.

Eventos en SimNet

Tipos de evento, lista: `[id_action, delay, priority]`

- `id_action`: nombre de una función, ordenable, se busca en un diccionario `{id_action : function}`
- `delay`: siguiente evento en `time_t + delay`.
- `priority`: para eventos que ocurren en un mismo momento.
- eventos: GenTraffic, GenerateTBs, TransmitTBs, AssignRes, ShowProgress, EndSimulation.
- delay en EndSimulation es la duración de la simulación.

Secuencia de simulación:

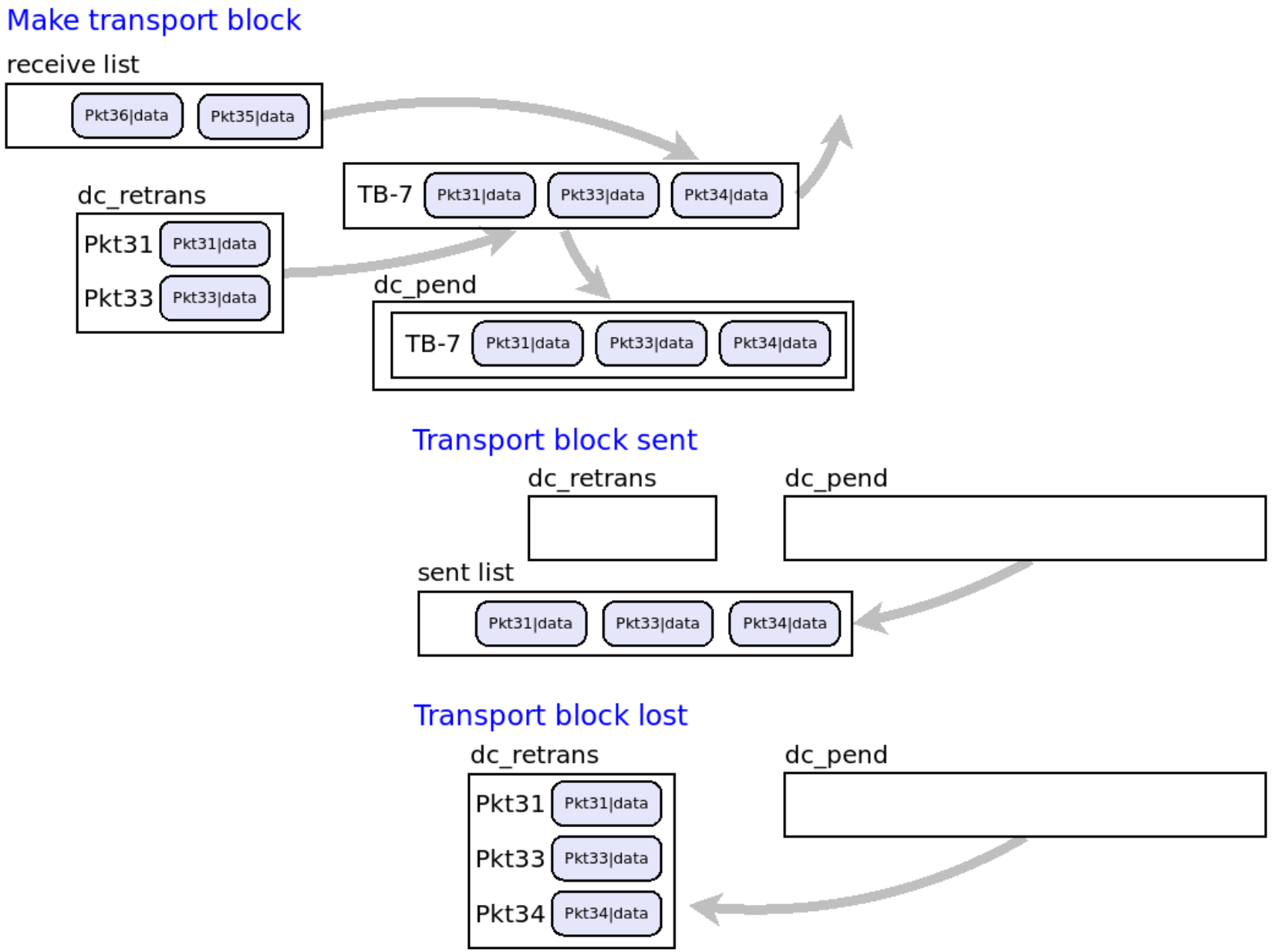
- al inicio, genera un evento de cada tipo.
- extrae un evento, sale siempre el más cercano.
- ejecuta la función correspondiente al evento extraído.
- la función devuelve un nuevo evento, cuyo delay puede variar.
- inserta el nuevo evento en la cola de prioridades.

PacketQueue, lista de paquetes

Formato de registro de un paquete:
`[ id_pkt, time_rec, time_sent, object | nr_bits ]`

- los datos pueden ser un objeto, un string o un número de bits.
- una vez transmitidos, pueden conservarse o descartarse.
- la lista puede tener longitud máxima, al superarla se descartan los paquetes.
- transmisión por paquetes o por transport blocks.
- un transport block incluye uno o más paquetes, según recursos, canal, etc., al momento de transmitir.
- un transport block puede transmitirse con éxito o perderse.
- los paquetes perdidos se retienen y retransmiten con la mayor prioridad.
- contadores registran recibidos, enviados, retransmitidos, descartados, en número de paquetes y en bits.

PacketQueue, proceso de transmisión (I)



PacketQueue, proceso de transmisión (II)

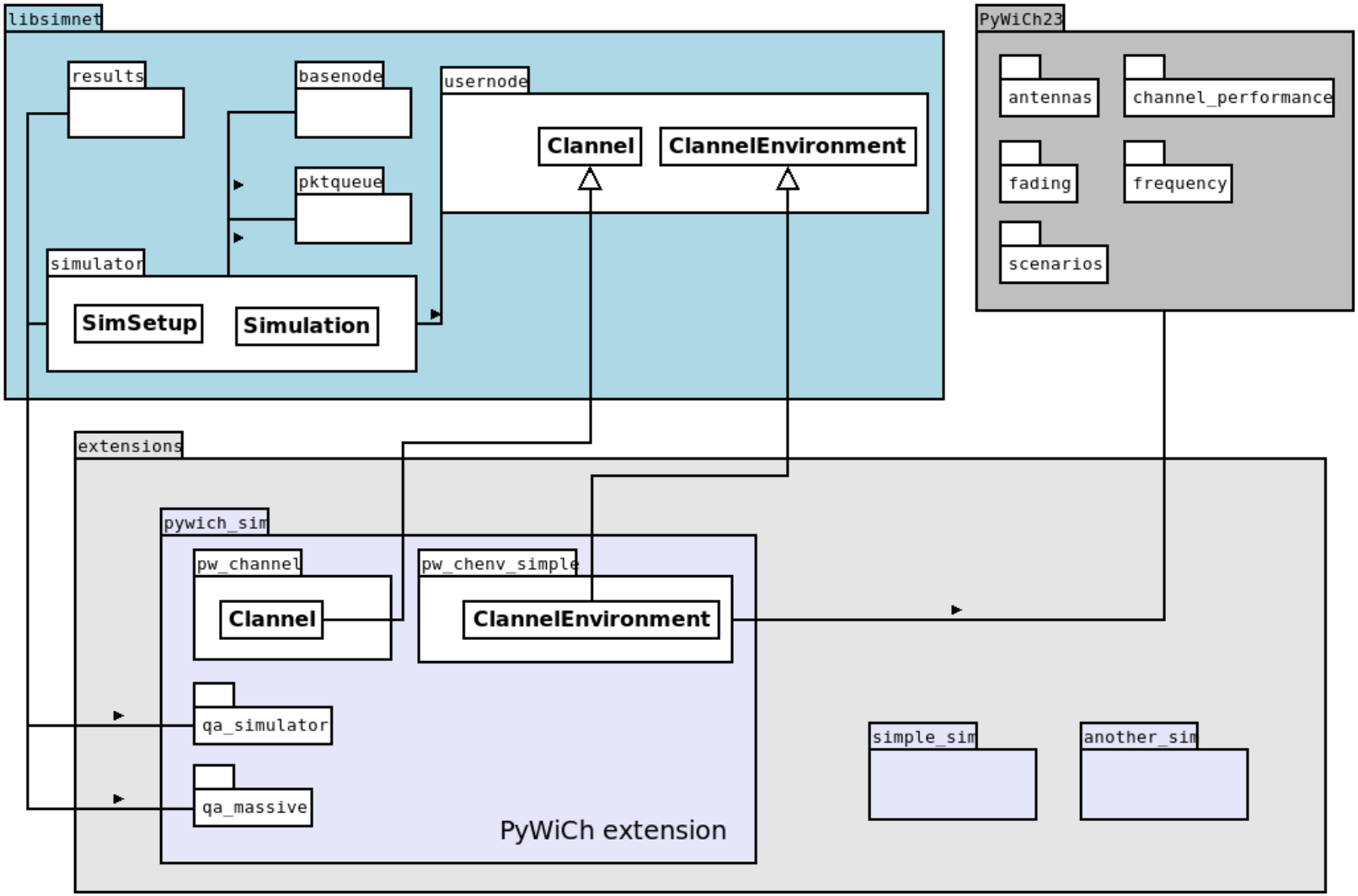
`mk_trblk(tb_size)` devuelve un transport block (TB):

- extrae del diccionario de retransmisión
  `dc_retrans = {id_packet : packet};`
- extrae de la cola de paquetes recibidos;
- crea TB = `[tr_blk_id, pkt_1, pkt_2, ...]`
- agrega a diccionario de TBs pendientes (en aire):
  `dc_pend[tr_blk_id] = [pkt_1, pkt_2, ...]`

`send_tb(tr_blk_id, "Sent"|"Lost", t_stamp)` procesa envío:

- para cada paquete en `dc_pend[tr_blk_id]` :
  - IF "Sent":
    - agrega timestamp al paquete; ajusta contadores.
    - opcionalmente lo mueve a la cola de paquetes enviados.
    - si el paquete está en `dc_retrans`, lo extrae.
  - IF "Lost":
    - IF el paquete no está en `dc_retrans`, lo agrega.
- extrae el TB de pendientes: `del dc_pend[tr_blk_id]`

PyWiCh, integración en SimNet



Próximos objetivos

Objetivos inmediatos:

- en libsimnet, identificar clases y métodos a sobreescribir.
- crear un ejemplo simple como guía para la extensión de libsimnet.
- documentación, páginas de descripción, instalación, extensión.
- pruebas de rendimiento en distintos escenarios.

Objetivos de próximo desarrollo:

- estudiar e integrar protocolos y algoritmos implementados en proyectos anteriores.
- adaptar interfaces de libsimnet para facilitar la extensión e integración de otras implementaciones de protocolos y algoritmos (mejora continua).

Final

Muchas gracias

IIE en el Este