University of Ljubljana
Faculty of Computer and Information Science

Modeliranje računalniških omrežij

OMNeT++ Programiranje preprostih modulov

Laboratorijske vaje

Torek, 22. oktobra 2013

Preprosti (simple) moduli

- definiramo v jeziku C++
- izhajajo iz razreda `cSimpleModule`
- `*.cc` in `*.h`
- dogodkovno vodeno programiranje
- vedno vključimo knjižnico `<omnetpp.h>`
- `Define_Module(ime_modula)`: makro, ki registrira modul – vstavimo na začetku `cc` datoteke
- osnovne metode
  - konstruktor in destruktor
  - `initialize()`
  - `handleMessage(cMessage *msg)`
  - `finish()`

Konstruktor

- izvede se ob kreaciji objekta – modula
- `imeModula::imeModula()`
- kliče ga simulator (ne uporabnik)

initialize()

```
virtual void initialize();
```

- kliče se še preden se začne procesiranje prvega dogodka
- če je potrebno, sprožimo prvi dogodek
  - kreiramo sporočilo:

```
msg = new cMessage("imeSporcila")
```

  - pošljemo sporočilo samemu sebi:

```
scheduleAt(startTime,msg);
```

handleMessage(cMessage *msg)

```
virtual void handleMessage(cMessage *msg);
```

- proženje ob vsakem dogodku – prihod sporočila *(message arrival)*
- sporočilo lahko predstavlja
  - sporočilo, ki ga modul pošlje samemu sebi (metoda `ScheduleAt`)
  - sporočilo od drugih modulov

Sporočila (Messages)

- dva razreda:
  - `cMessage` : sporočila med moduli, sporočila samemu sebi
  - `cPacket` : izpeljava, ki se uporablja za omrežne pakete (vaja 4)
- `cMessage` parametri
  - `name (const char *)`
  - `kind (integer)`
  - `scheduling priority`
  - `send time, arrival time, source module, source gate, destination module, destination gate`
  - `time stamp`

```
cMessage *msg = new cMessage(name, kind);
```

oba parametra sta opcijska

handleMessage(cMessage *msg)

Metode povezane s pošiljanjem sporočil

- `send()`
  - pošiljanje sporočil drugim modulom preko specificiranih vrat
  - primer: `send(job,"out");`
  - `sendDelayed()`: zakasnjeno pošiljanje
- `scheduleAt()`
  - pošiljanje sporočila samemu sebi
  - npr. generator prometa pošlje sporočilo samemu sebi po času, ki je definiran z medprihodnim časom
  - primer:

```
scheduleAt(simTime()+par("interArrivalTime").doubleValue(), msg);
```

- `cancelEvent()`:
  - preklic dogodka, ki je bil prožen z metodo `scheduleAt()`
  - primer: `cancelEvent(msg);`
- `cancelAndDelete()`:
  - preklic in brisanje dogodka

finish()

```
virtual void finish();
```

- kliče se ob koncu simulacije – ko ni več dogodkov *(events)*
- samo, če se je simulacija izvedla brez napake `(runtime error)`
- finish() se ne kliče vedno - čiščenje moramo realizirati v destruktorju
- možne funkcije
  - pisanje statistike
  - obdelava rezultatov
  - ...

Destruktor

- izvede se ob uničenju objekta
- `imeModula::~imeModula()`
- v metodi izbrišemo vse, kar smo alocirali z `new`
- za sporočila, ki so v teku, uporabimo metodo `cancelAndDelete(msg)`
- za ostale objekte uporabimo metodo `delete(...)`
- OMNeT++ nas ob koncu simulacije opozori, če nismo počistili za sabo
- kliče ga simulator

OMNeT++ razredi

Dostopni preko knjižnjice `omnetpp.h`

- `cQueue`: implementacija čakalne vrste
- `cMessage`
  - objekt za izmenjavo sporočil med moduli
  - sporočila samemu sebi
- `cGate`: objekt za delo z vrati
- `cChannel`: objekt za delo s kanali
- `cPacket`: izpeljava razreda `cMessage` za delo s paketi
- ...

Parametri modula

- parametri definirani v NED datotekah ali v datoteki omnetpp.ini

```
double delay = par("delay");
long numJobs = par("numJobs").longValue();
```

- `volatile` parametri: parameter se evaluira vsakič, ko ga simulator potrebuje

Delo z vrati

- Objekt `cGate`

```
cGate *outGate = gate("out");
```

- Pri `inout` vratih je potrebno uporabiti dva objekta

```
cGate *gIn = gate("g$i");
cGate *gOut = gate("g$o");
```

- `hasGate()` metoda

```
if (hasGate("optOut")) send(new cMessage(),"optOut");
```

- `gateSize()` pri vektorskih vratih

```
for (int i=0; i<gateSize("out"); i++)
{ cGate *gate = gate("out", i); ...}
```

Klicanje metod drugih modulov

- na začetek metode, ki jo kličemo od zunaj damo makro `Enter_Method()` oz. `Enter_Method_Silent()`
- `Enter_Method()` – izpis niza, silent = brez izpisa
- `Enter_Method("release(%ld)", amountToRelease);`

Naloga

V orodju OMNeT++ razširite model *M/M/1* sistema na *M/M/c* sistem (*c* predstavlja število strežnikov). Pri tem naj bo *c* podan kot parameter.

Bodite pozorni na to, da v destruktorju počistite za sabo - OMNeT++ nas ob koncu simulacije na to opozori!