Egy feladatvégző szerver fürt modellezése és verifikációja

Készítette: **Izsó Benedek**

2011.

Tartalom

1 A protokoll leírása

Együttműködő feladatvégző egységek (pl. szerverek) esetén különösen fontos, hogy egymás működőképességéről konzisztens képpel rendelkezzenek. Mivel az egységek elromolhatnak, újraindulhatnak, ezért egy elosztott rendszerben ez nem mindig valósul meg.

Az alábbi protokoll egy feladatvégző egységekből (csomópontokból) álló, egy irányban láncolt gyűrű struktúrát épít fel, és tart karban. Minden csomópontot egy szám azonosít [0..N-1], ami a gyűrűben elfoglalt helyét is meghatározza. A gyűrűben az i. sorszámú csomópont után az i+1. következik, az N-1. sorszámú csomópont után pedig a 0. következik.

Egy csomópont a gyűrűhöz legfeljebb 5 perc után megpróbál csatlakozni. Teszi ezt úgy, hogy

1. 0-tól sorban elkezdi lekérdezni (ping-elni) a csomópontokat a saját azonosítójáig, és amelyik csomópontot így utoljára élőnek találja, arról feltételezi, hogy az van előtte, tehát oda majd befűzi magát.
2. Ha nem talált élő csomópontot, akkor 0-tól visszafele is elkezdi a ping-elést, amíg nem talál élő csomópontot maga elé.
3. Ha így is elér a saját azonosítójáig, akkor ő maga az első a körben.
4. Ha a művelet során talált maga előtt lévő csomópontot, akkor be is fűzi azután magát a gyűrűbe, azaz lekérdezi az előző csomópont *következő*jét, és azt beállítja magának; illetve megkéri az előzőt, hogy a következőre mutató pointerét állítsa őrá, azaz az újonnan belépett egyedre.

Belépés után a csomópont alapból dolgozik, ám 6 perc folyamatos munka után bármikor elromolhat. Éppen ezért legalább 5 perc munka után bármikor lepihenhet a gép, majd pihenése után újult erővel folytatja munkáját. (Azaz ismét el kell telnie legalább 6 percnek, hogy elromoljon, illetve legalább 5 percet dolgoznia kell.)

Minden csomóponthoz egy külön modulba épített watchdog is tartozik, mely a gép elromlását meg tudja állapítani. Ekkor a csomópont helyett ő válaszol minden a csomóponthoz érkező üzenetre, mégpedig hibajelzéssel. Amint a csomópont megjavul (maximum 5 perc), és csatlakozni akar a rendszerhez, akkor utasítja a watchdog modulját, hogy lépjen vissza figyelő állásba, és hagyja magát a csomópontot kommunikálni.

Ha elromlott egy csomópont, akkor a gyűrűben lévő másik (hibátlan) csomópont *következő* pointere mutathat hibás egyedre. Éppen ezért munka közben kétpercenként ping utasítással ellenőrzi, hogy él-e az a gép, akiről azt hiszi, hogy utána következik. Ha kiderül, hogy nem, akkor a következő sorszámú csomópontot pingeli, és így tovább, amíg egy élőt nem talál. Ekkor azt be is jegyzi magának *következő*ként.

Egy csomópont akár dolgozik, akár pihen, a struktúrát fenntartó üzenetekre mindig egyből válaszol, utána pedig előző pozíciójába tér vissza (ha dolgozott tovább dolgozik, ha pihent, akkor pedig tovább pihen). Szerencsére az üzenetváltások pikk-pakk megtörténnek, csak a javítás, a gép munkavégzése illetve pihenése mérhető ki. (A gyűrű struktúrát a rendszer működése közben használja fel, ám ezt a működést már nem kell modellezni.)

2 A feladat modellezése

A feladat modellje a csatolt *nodes.xml* fájlban található.

3 A követelmények ellenőrzése

Az alábbi követelmények ellenőrzése olyan konfigurációban történt, amikor a rendszerben három csomópont lehet.

1. Bizonyítsuk be, hogy nem alakulhat ki holtpont!

   A[] not deadlock

2. Vizsgáljuk meg, hogy mindig lehet-e olyan helyzet, amikor az összes csomópont pihen? Miért?

   A<> (forall (i : int[0,N-1]) Node(i).Idle)

3. Bizonyítsuk be, hogy lehet olyan helyzet, hogy mindegyik csomópont dolgozik!

   E<> (forall (i : int[0,N-1]) Node(i).Up)

4. Bizonyítsuk be, hogy ha egyszer mindegyik csomópont dolgozik, akkor utána mindig előállhat újra ugyanez a helyzet!

   (forall (i : int[0,N-1]) Node(i).Up) --> (forall (j : int[0,N-1]) Node(j).Up)