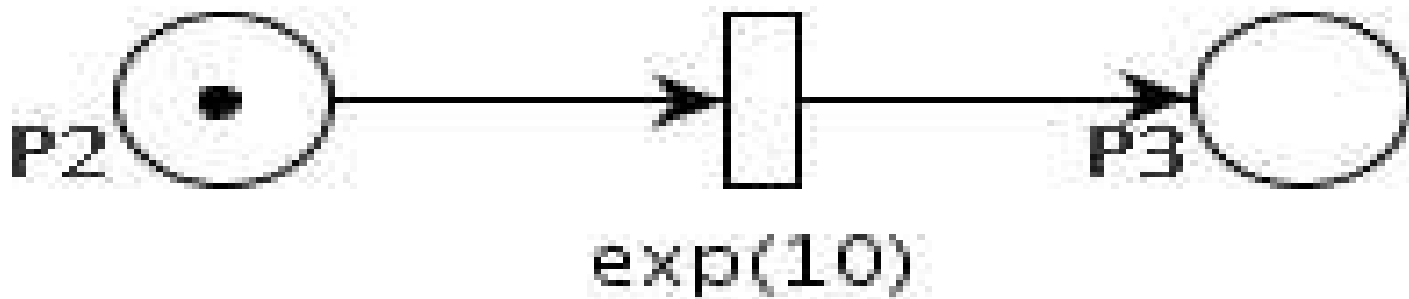
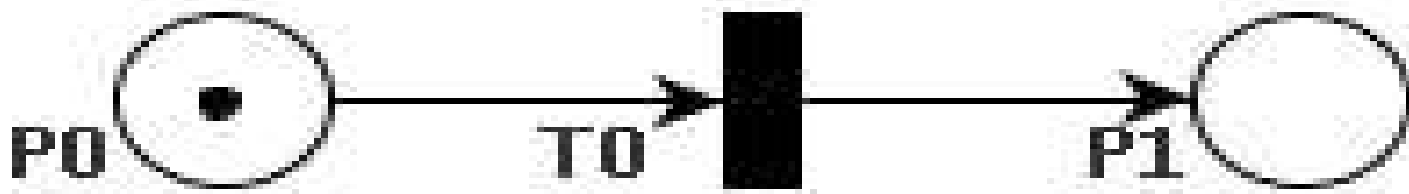


Demonstrační cvičení IMS #1


Diskrétní modelování – Petriho sítě a
SIMLIB/C++


Petriho síť v modelování



Parametry míst

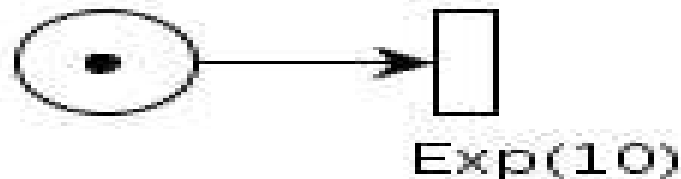
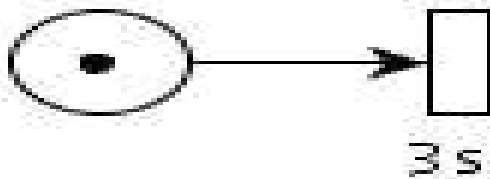
- ◆ Místa mohou specifikovat:
 - ◆ kapacitu (maximum uložených značek)
 - ◆ počáteční stav (počet značek)
 - ◆ (daný stav)

Kap = 10 



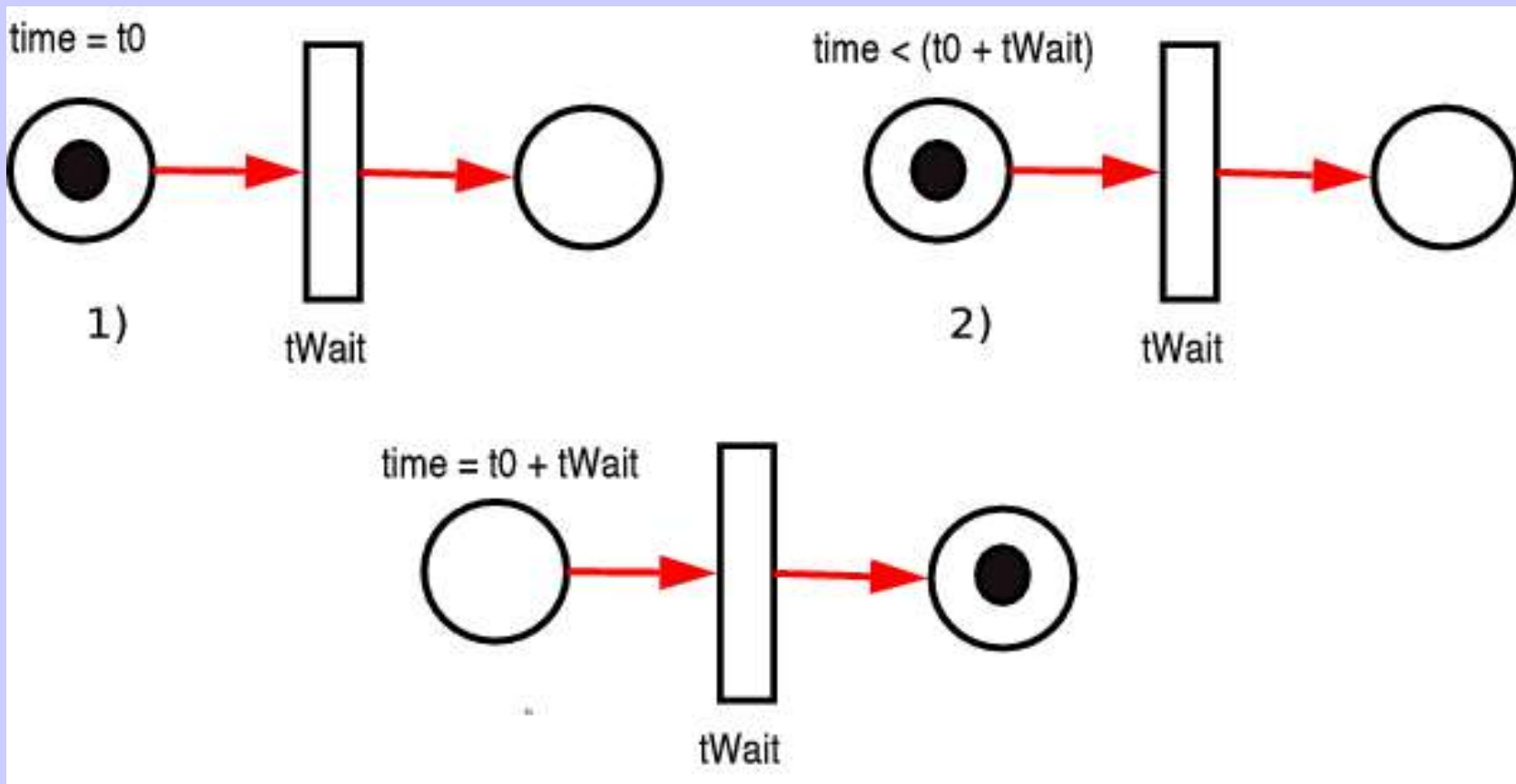
Parametry přechodů

- ◆ Časování – konstantní, stochastické (pouze u časovaných)
- ◆ Priorita (pouze okamžité)
- ◆ Pravděpodobnost (pouze okamžité)
- ◆ Parametry **NELZE** kombinovat



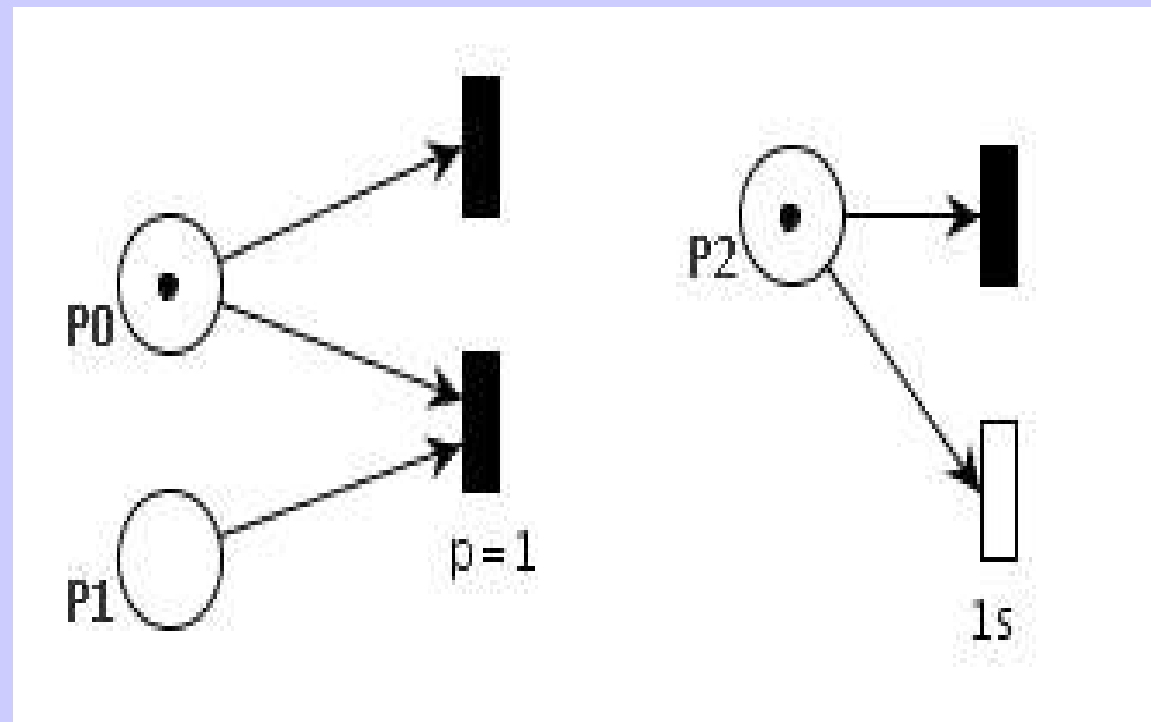
Sémantika časovaného přechodu

- ◆ kapacita, generátor náhodných čísel

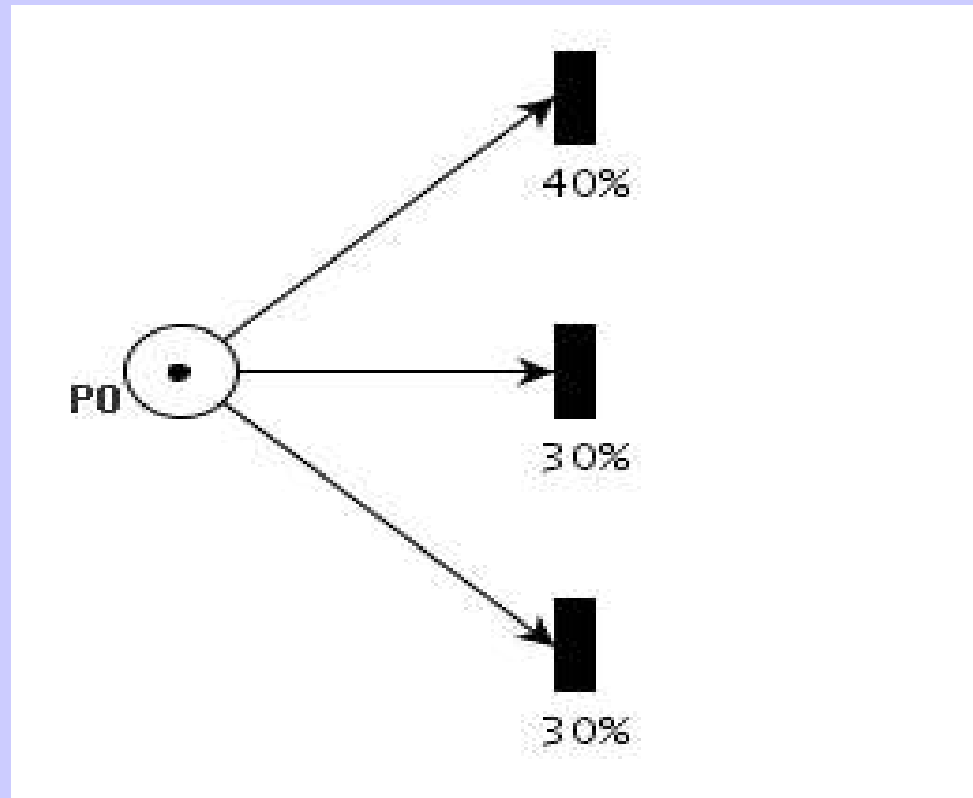


Parametry přechodu

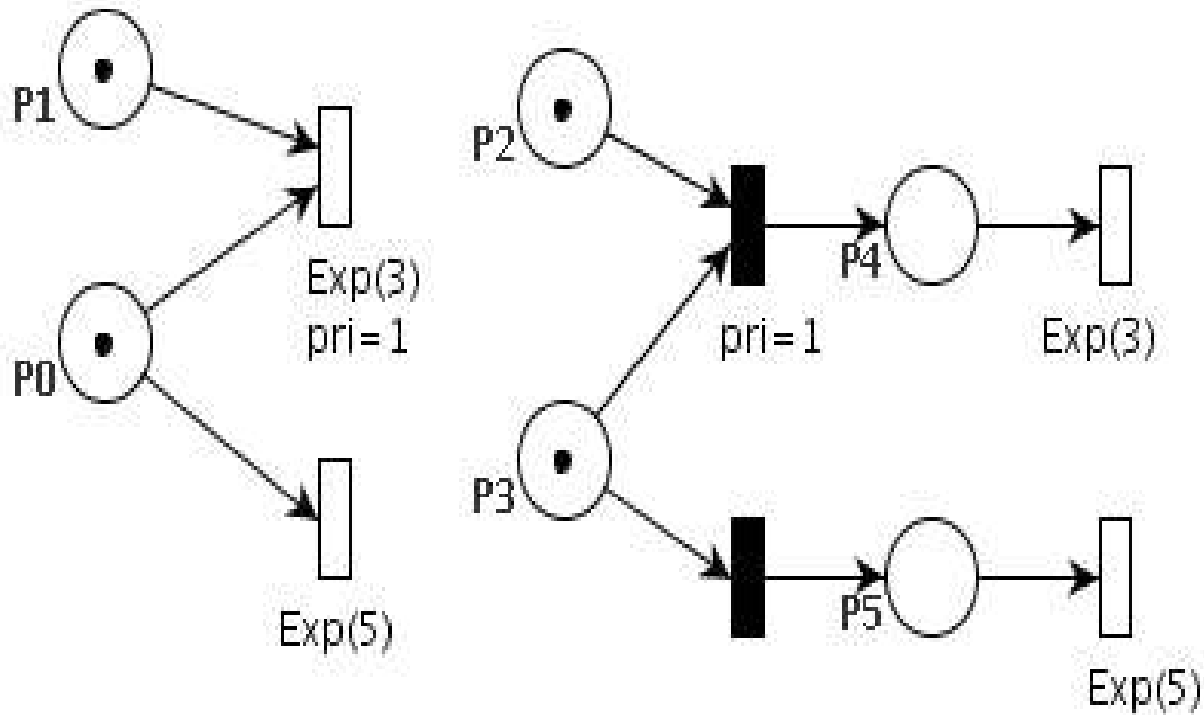
- ◆ Okamžitý (nečasový) přechod má automaticky prioritu.
- ◆ Značíme $\text{pri}=X$, kde X je $\{0,1,2,\dots\}$.
Implicitně $\text{pri}=0$



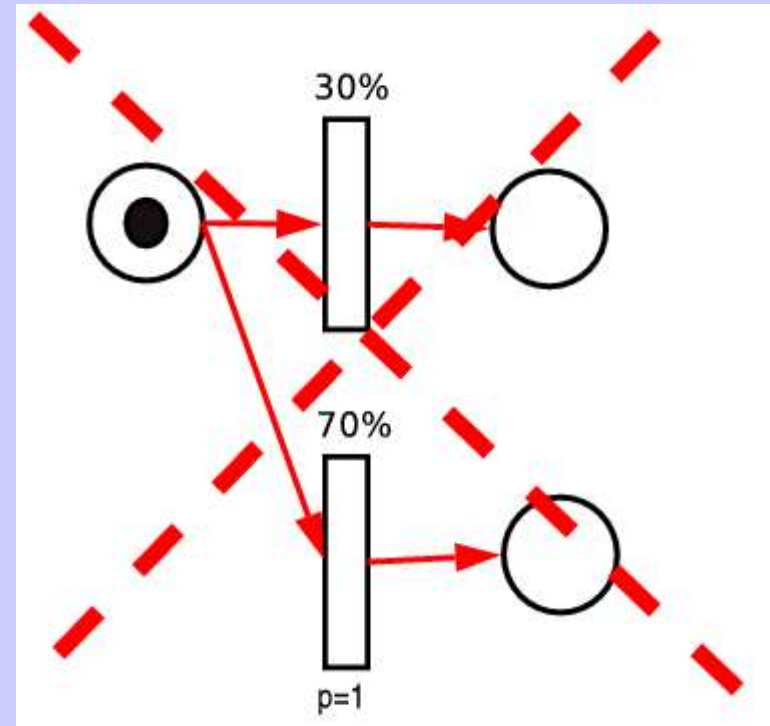
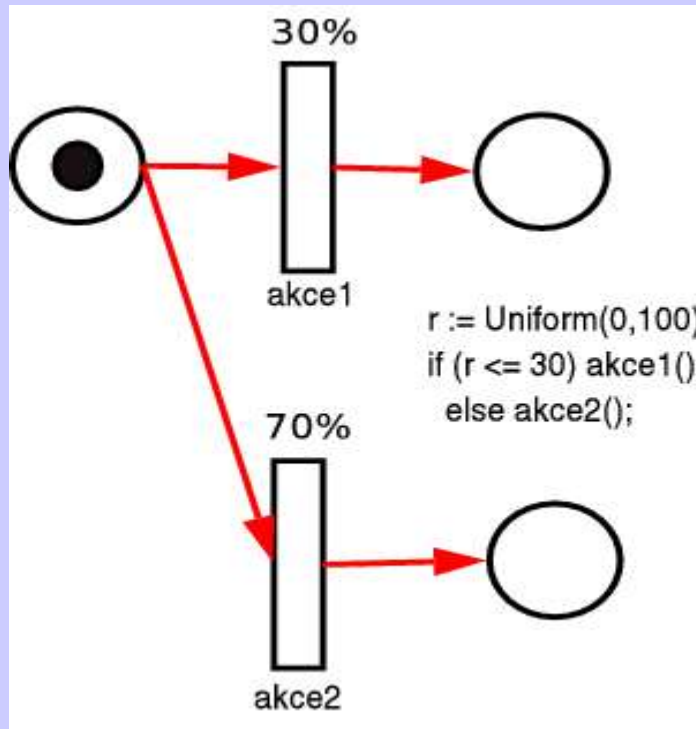
Parametry přechodu - pravděpodobnost



!!!!!! Chybně !!!!!

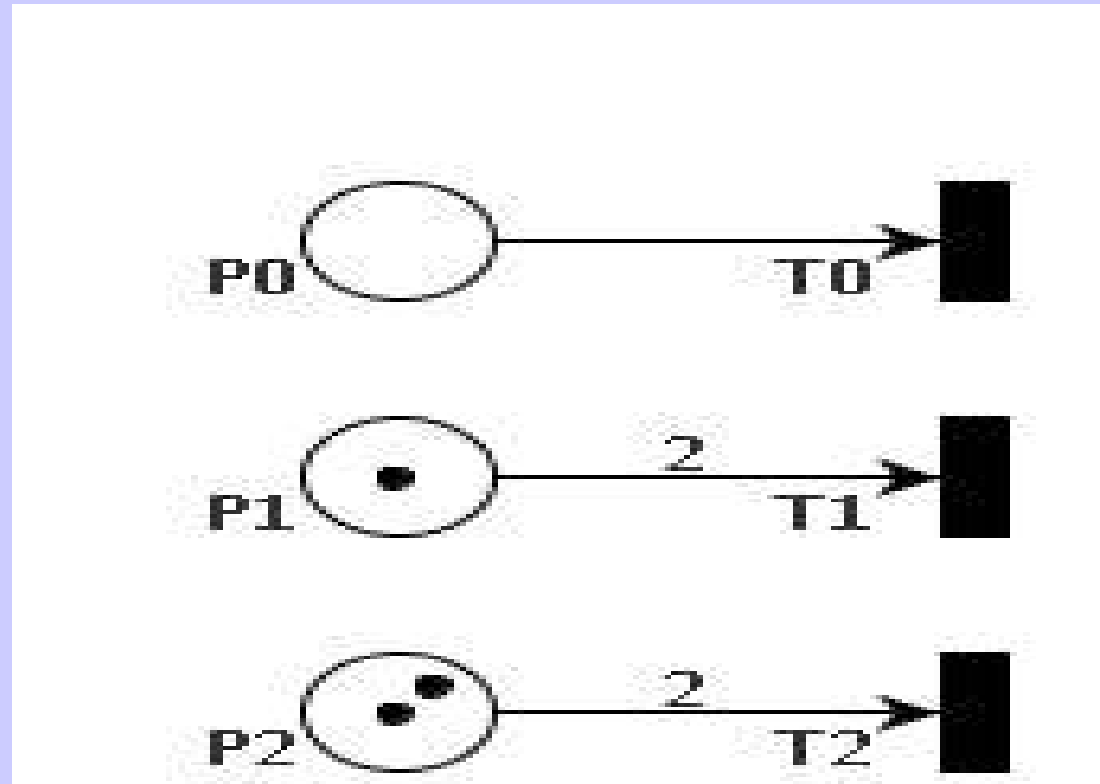


Chyby



Parametry hrany

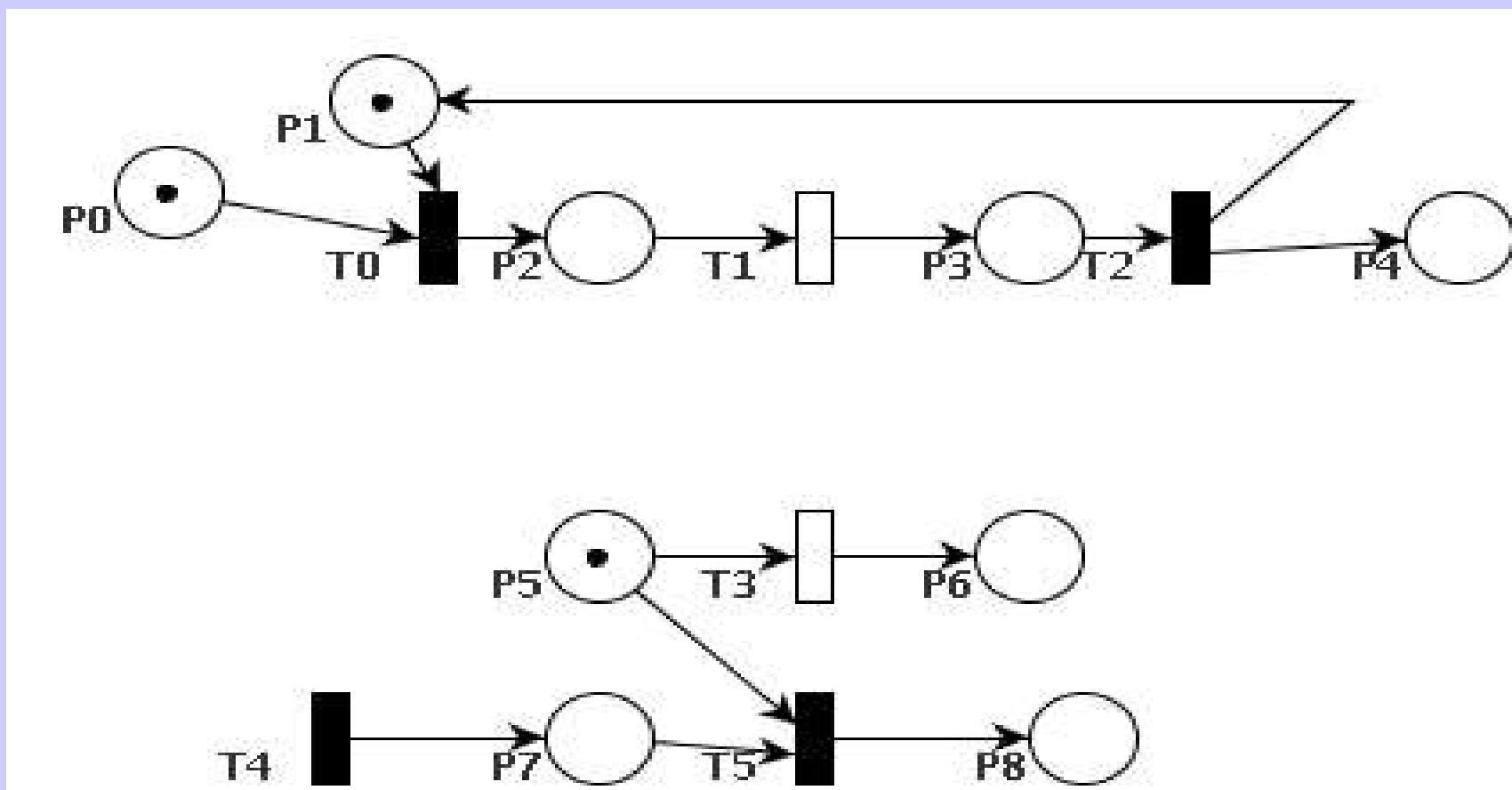
- ◆ Váha hrany
- ◆ Implicitně 1



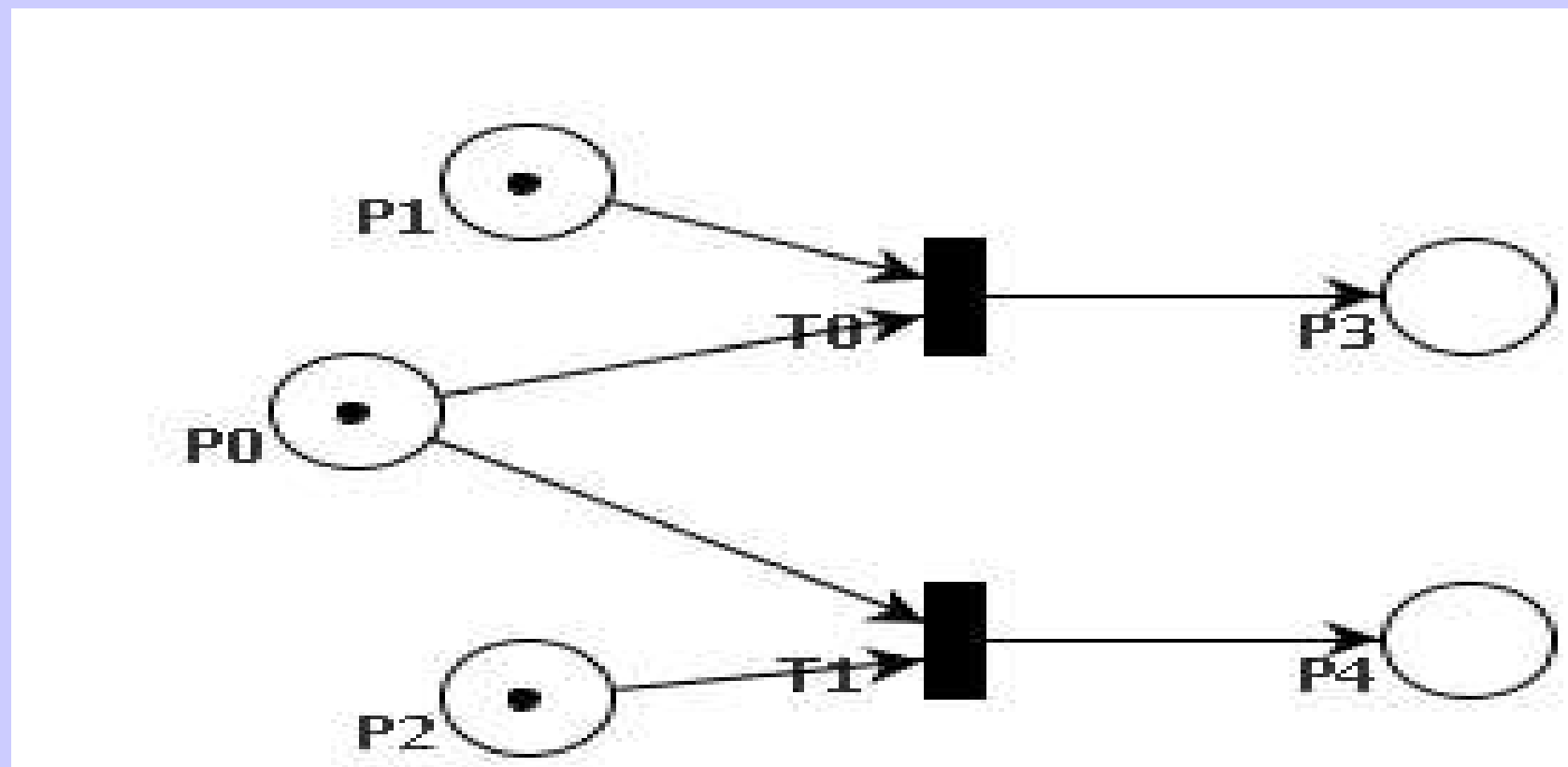
Příklady

- ◆ Modelujeme diskrétní systémy – událostně nebo procesně řízená simulace.
- ◆ Identifikovat procesy.
- ◆ Identifikovat zdroje (obvykle sdílené).
- ◆ SHO:
 - ◆ obslužné linky, charakteristiky
 - ◆ způsob vstupování procesů (intervaly mezi příchody)
 - ◆ způsob obsluhy procesů (intervaly, obslužná síť)
 - ◆ statistiky

Obsazení zařízení

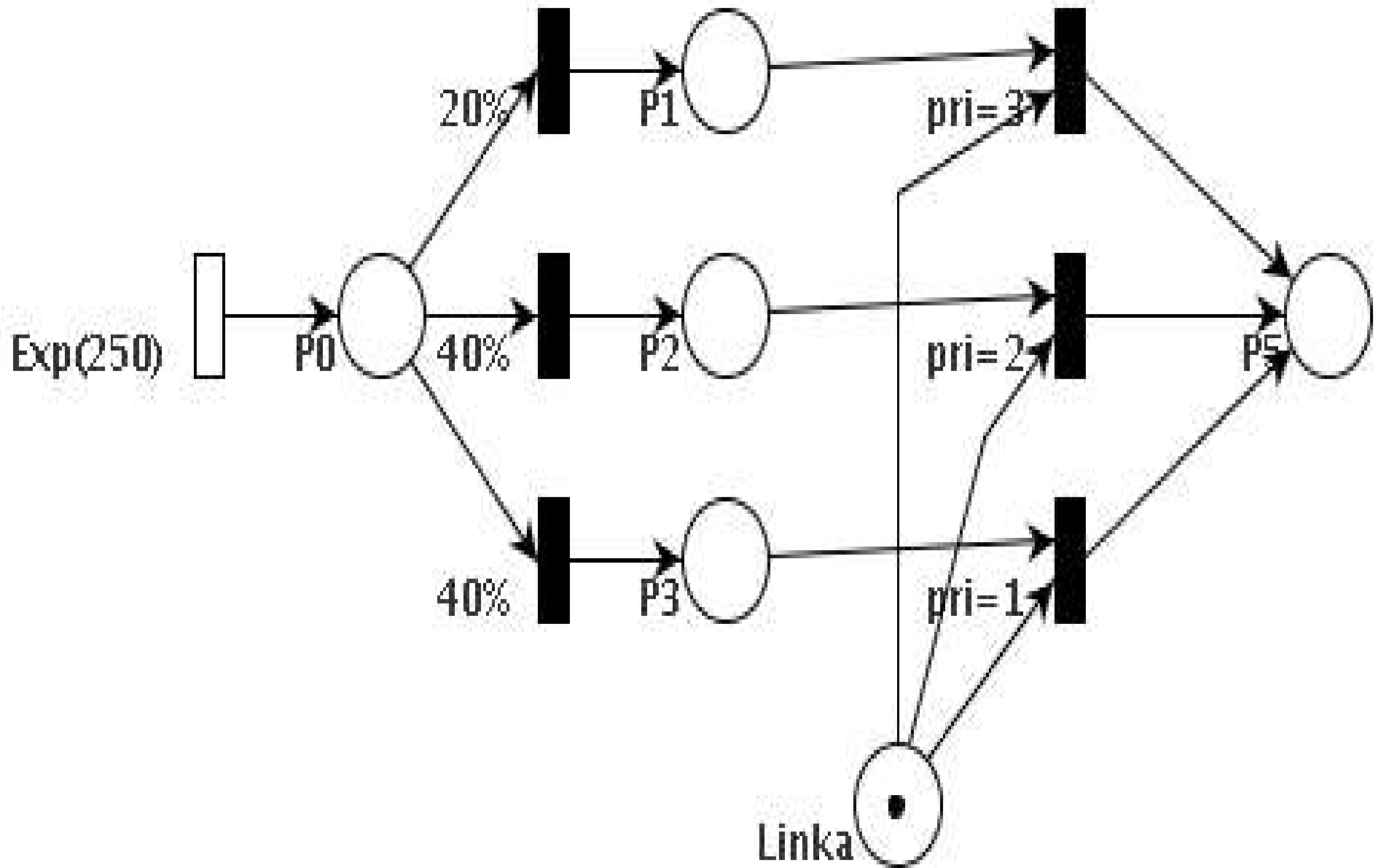


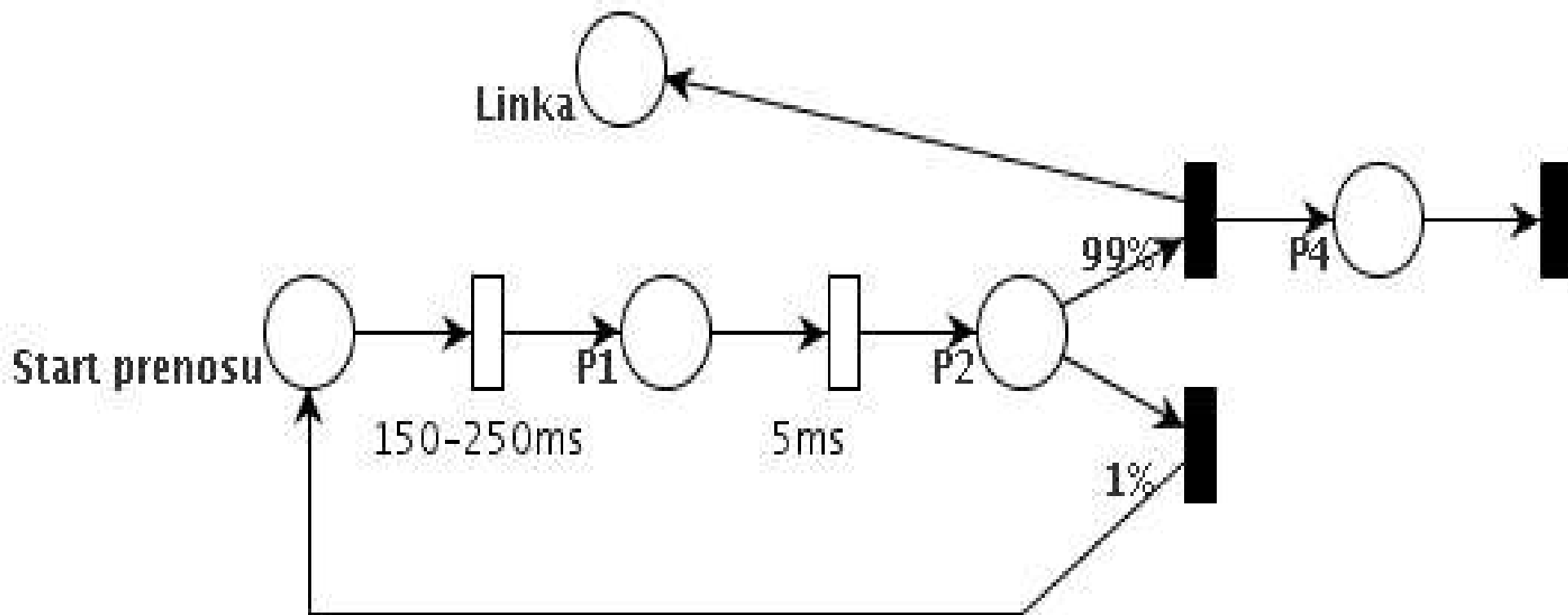
Volba z více zařízení



Příklad “přenosová linka”

- ◆ V intervalech daných exponenciálním rozložením se středem 250 ms vzniká potřeba na odeslání zprávy
- ◆ Odeslání zprávy linkou trvá 150-250 ms
 - ◆ pak dotaz na správnost přenosu – 5 ms
 - ◆ 1% chyba, pak opakování přenosu
- ◆ Priority zpráv:
 - ◆ 20% - vysoká priorita
 - ◆ 40% - střední
 - ◆ 40% - malá (normální)






```
Facility Linka("Prenosova linka");  
Histogram hist("Doba v systemu", 100, 100, 15);
```

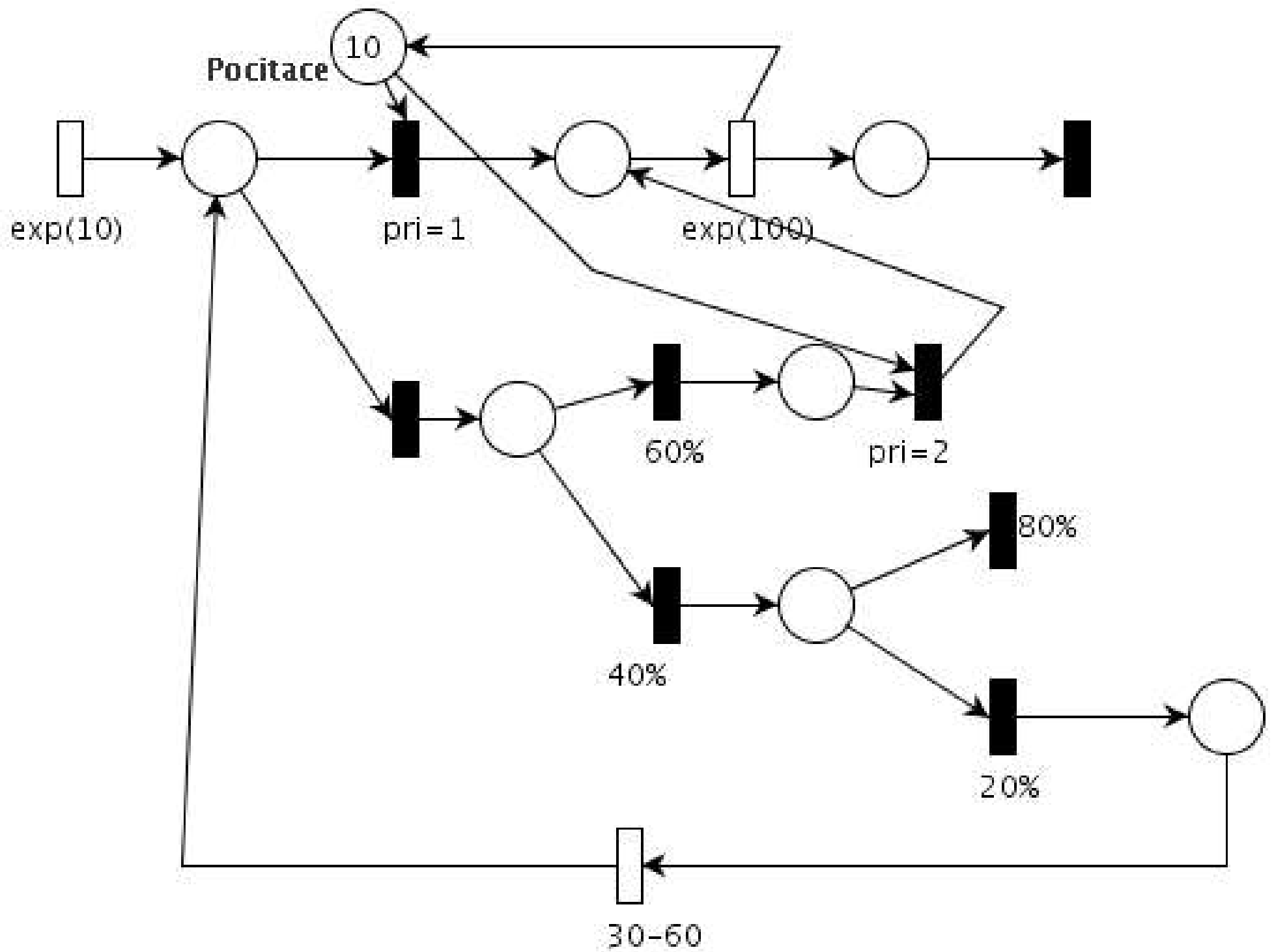
```
class Paket : public Process {  
public:  
    void Behavior() {  
        double time=Time;  
        Seize(Linka);  
        opak:  
        Wait(Uniform(150,250)); // prenos dat  
        Wait(5); // dotaz na uspesnost  
        if (Random()<=0.01) goto opak; // 1% chyba, opakovani  
        Release(Linka);  
        hist(Time-time);  
    }  
};
```

```
class Gener : public Event {
public:
    void Behavior() {
        /*
            20% - vysoka priorita
            40% - stredni
            40% - mala
        */
        double r = Random();
        Paket *p = new Paket;
        if (r<=0.2) p->Priority=3;
        else
            if (r>0.2 && r<=0.6) p->Priority=2;
            else
                p->Priority=1;

        p->Activate();
        Activate(Time+Exponential(250));
    }
};
```

Učebna

- ◆ V poč. učebně je 10 počítačů. Studenti přichází v intervalech daných exp. rozložením se středem 10 min.
- ◆ Pokud je počítač volný, obsadí ho a pracují (exp(100min)).
- ◆ Jinak se 60% okamžitě postaví do fronty. Zbytek odchází. 20% se však po 30-60 min vrací.



```

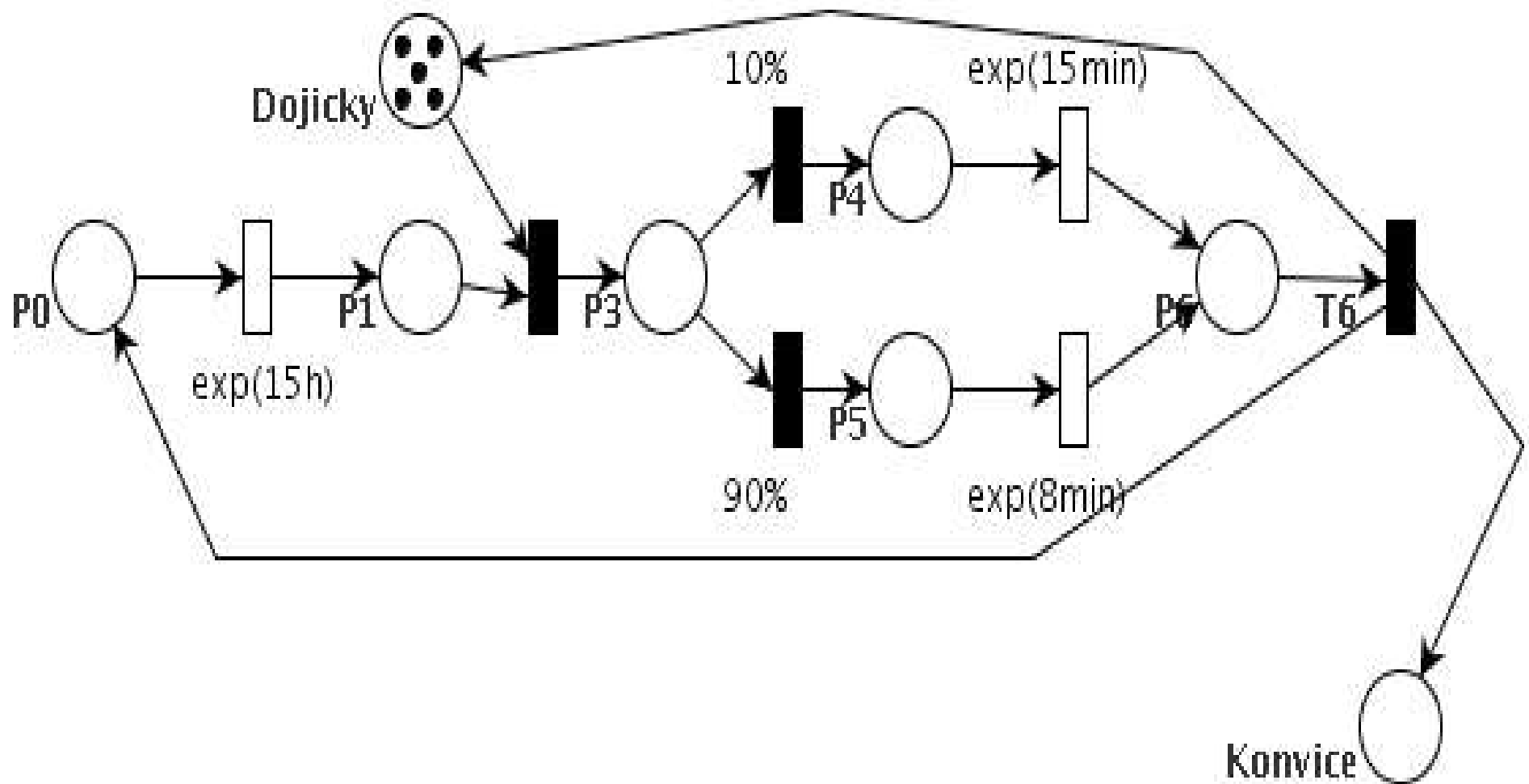
class Student : public Process {
public:
    void akce() {
        Enter(pocitace, 1);
        Wait(Exponential(100));
        Leave(pocitace, 1);
    }
    void Behavior() {
        opak:
        if (pocitace.Full()) {
            if (Random()<=0.6) akce();
            else {
                if (Random()<=0.2) {
                    Wait(Uniform(30,60));
                    goto opak;
                }
            }
        } else
            akce();
    }
};

```

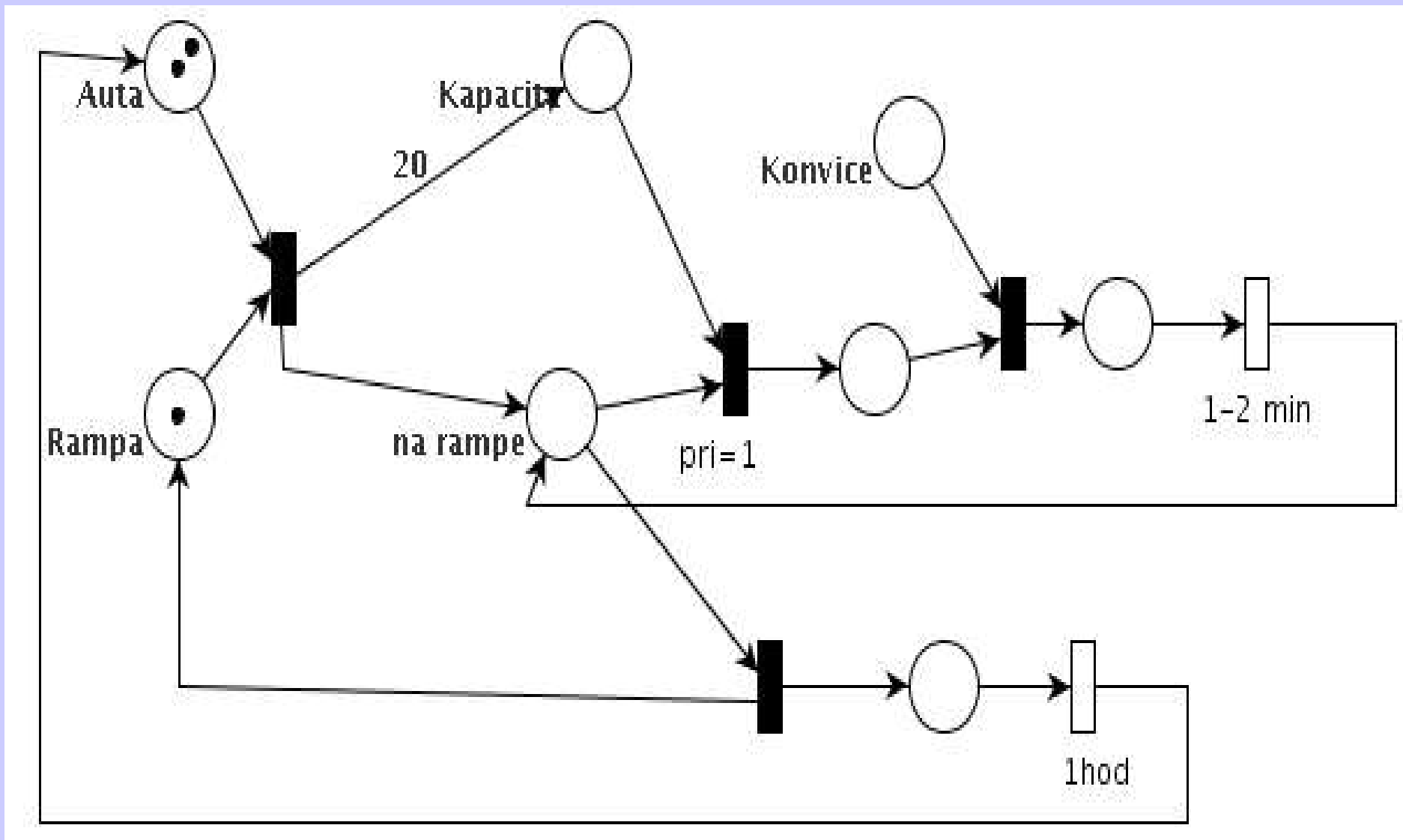
Příklad: kravín

- ◆ Kravín má 100 krav, 5 dojiček, 1 nakládací rampu, 2 auta
- ◆ Krávy v intervalech $\exp(15\text{h})$ potřebují podojit (dojička, 10% případů trvá $\exp(15\text{min})$, jinak $\exp(8\text{min})$).
- ◆ Vznikne konvice s mlékem. Nakládají se na rampě do auta (kapacita 20). Auto náklad odváží (1hod).
- ◆ Identifikovat procesy.

Proces "kráva"



Proces "auto"




```
class Krava : public Process {
    void Behavior() {

        // zivotni cyklus
        while (1) {
            Wait(Exponential(15*60)); // 15 hod

            Enter(dojicky, 1); // bere dojicku

            // doba dojeni
            if (Random()<=0.1) Wait(Exponential(15));
            else Wait(Exponential(8));

            konvic++; // dalsi hotova konvice

            Leave(dojicky, 1); // uvolneni dojicky
        }
    }
};
```

```

class Auto : public Process {
    void Behavior() {
        while (1) {
            Seize(rampa); // postavi se na rampu
            double time=Time;

            // bere 20 konvic
            for (int a=0; a<20; a++) {
                WaitUntil(konvic>0); // ceka na hotovou konvici
                konvic--;
                Wait(Uniform(1,2)); // nalozi ji
            }

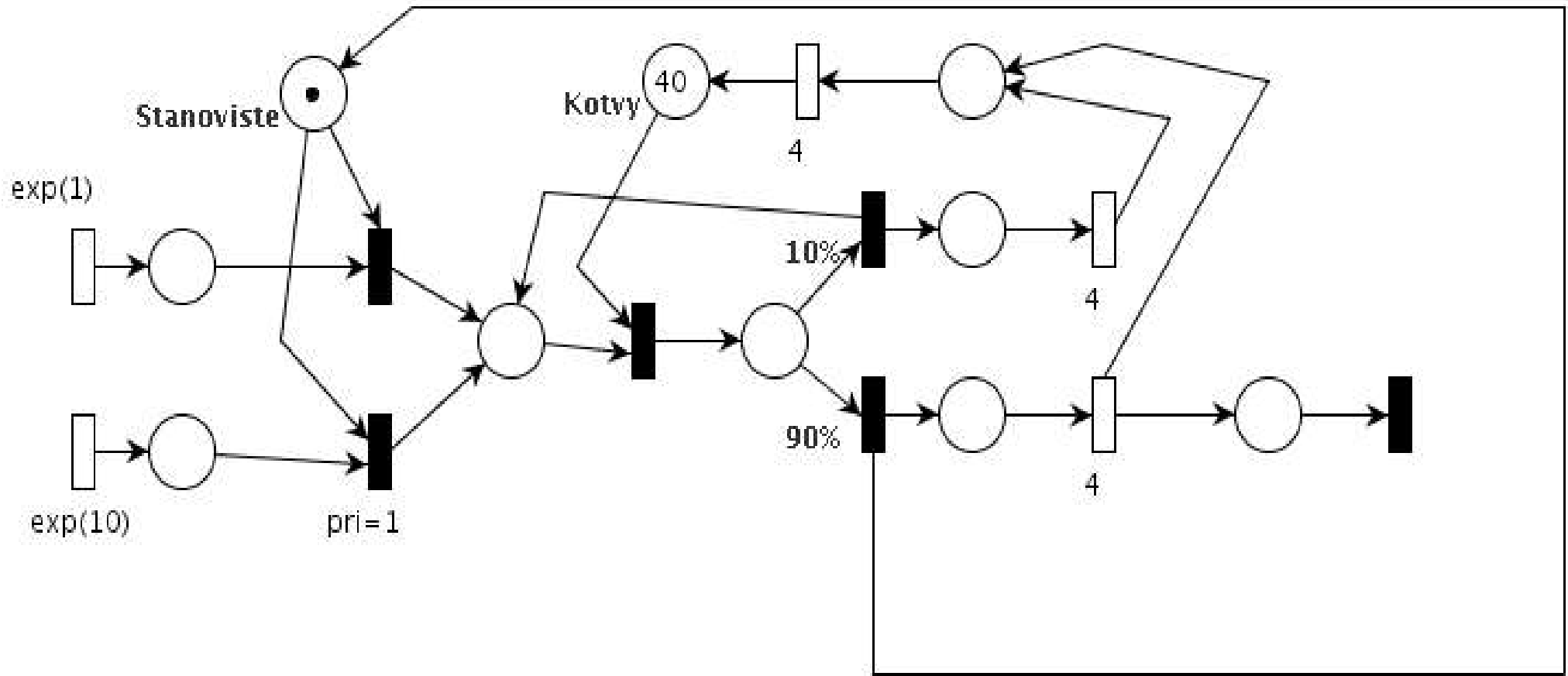
            Release(rampa);
            nalozeni(Time-time); // doba nakladani
            Wait(60);
        }
    }
};

```

```
int main() {  
    Init(0,200*60); // 200 hodin casovy ramec  
  
    // vygenerovat 100 krav do systemu (zustavaji tam)  
    for (int a=0; a<p_krav; a++)  
        (new Krava)->Activate();  
  
    // dve auta do systemu  
    (new Auto)->Activate();  
    (new Auto)->Activate();  
  
    Run();  
  
    rampa.Output();  
    dojicky.Output();  
    nalozeni.Output();  
}
```

Příklad “vlek”

- ◆ Lyžaři: obyčejní - $\exp(1)$, závodníci – $\exp(10)$, závodníci mají přednost
- ◆ 40 kotev, jedno startovní stanoviště
- ◆ při nastupování:
 - ◆ 10% - chyba, nástup se opakuje, kotva jede dál
 - ◆ 90% - 4 min nahoru, kotva jede další 4min zpět



```
const double jedna_cesta = 4.0;
```

```
Store Kotvy("Sklad kotev", 40);  
Facility Stanoviste("Stavoviste");
```

```
Histogram dobaCesty("Doba stravena lyzarem u vleku", 0, 1, 15);  
Histogram pocetPokusu("Pocet pokusu nastoupit", 1, 1, 10);
```

```
Stat cekaniZavodniku("Doba cekani zavodniku");
```

```
// generuje dva typy lyzaru. Obecny predpis
class Generator : public Event {
public:
    Generator(double interv, int pri) : Event() {
        Interval = interv;
        Pri = pri;
    };

    void Behavior() {
        (new Lyzar(Pri))->Activate();
        Activate(Time+Exponential(Interval));
    }

    double Interval;
    int Pri;
};
```

```
int main()
{
    SetOutput("lyzar.dat");
    Init(0, 1000);
    (new Generator(1,0))->Activate();
    (new Generator(10,1))->Activate();
    Run();

    Kotvy.Output();
    Stanoviste.Output();
    dobaCesty.Output();
    pocetPokusu.Output();
    cekaniZavodniku.Output();
}
```



```

class Lyzar : public Process {
public: Lyzar(int pri) : Process(pri) {} ;
void Behavior() {
    double time = Time;
    int pok=0;
    Seize(Stanoviste);

    opak:
    Enter(Kotvy, 1);
    Wait(Exponential(0.5)); pok++;
    if (Random()<=0.1) {
        // nezdareny start, kotva jede sama dve cesty
        (new KotvaBezi(2))->Activate();
        goto opak;
    }
    Release(Stanoviste);
    pocetPokusu(pok);
    Wait(jedna_cesta);
    dobaCesty(Time-time);
    (new KotvaBezi(1))->Activate(); // nahore opousti kotvu a ta jede
sama dolu do skladu
}

```

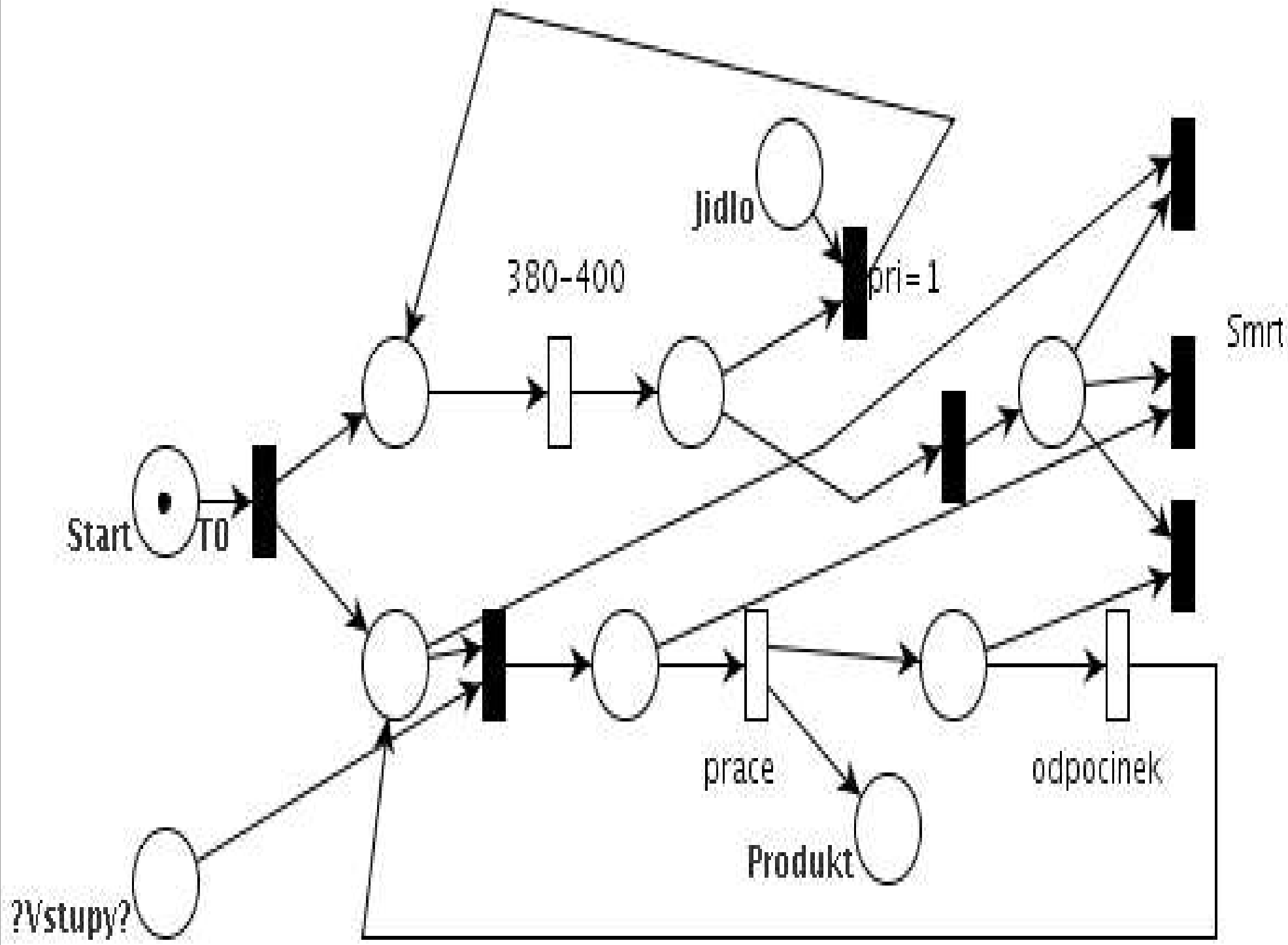
```
// proces volne ujizdejici kotvy
class KotvaBezi : public Process {
public:
    KotvaBezi(int t) : Process() { T=t; } ; // T=1 - jedna cesta,
T=2 - cesta tam a zpet

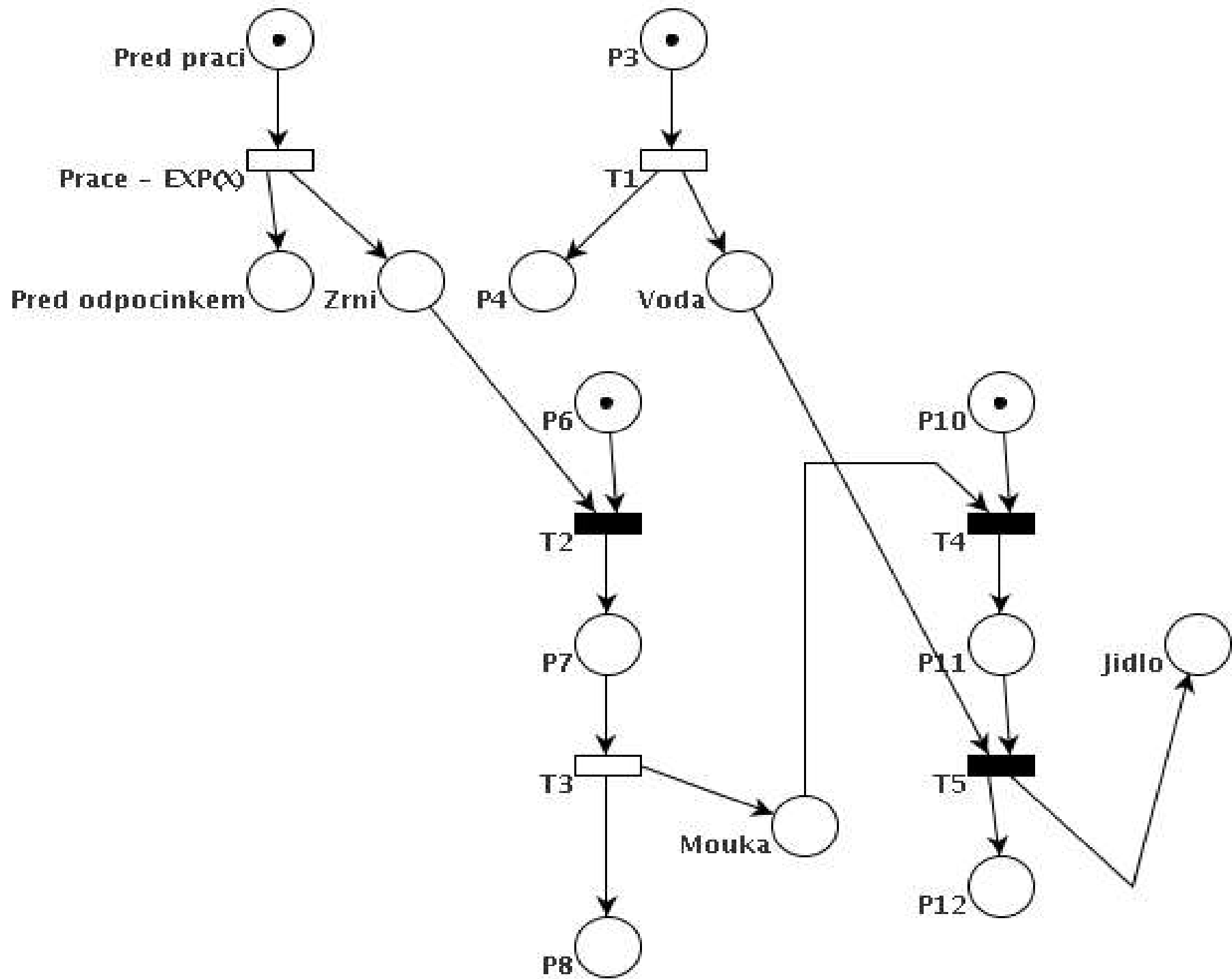
    void Behavior() {
        Wait(jedna_cesta*T);
        Leave(Kotvy, 1); // dojede na seradiste a uvolni kotvu
    }

    int T;
};
```

Výrobní systém

- ◆ Máme profese ve výrobním systému:
zemědělec, vodohospodář, mlynář, pekař
- ◆ úředníci
- ◆ každá entita v systému musí v intervalech
380-400 min sníst jedno jídlo, jinak “opouští
systém”
- ◆ sledujeme chod systému





```
class Profese:public Process {
public:
    Profese(char *name):Process() {
        (new JidloProc(this))->Activate();
    }
    void Behavior() {
        while (1) {
            prace();
            odpocinek();
        }
    }

    int najezSe() {
        if (skladJidlo.obsahuje() > 0) {
            skladJidlo.vzit(this);
            return 1;
        }
        return 0;
    }
}
```

```

class JidloProc:public Process {
public:
    JidloProc(Profese * kdo):Process() {
        Kdo = kdo;
    };
    void Behavior();
    Profese *Kdo;
};

void JidloProc::Behavior()
{
    while (1) {
        Wait(Uniform(380, 400));
        if (!Kdo->najezSe()) {
            Print("Cas %f , Process %s zdechl\n", Time, Kdo->Name);
            if (Kdo->where())
                Kdo->Out();
            Kdo->Cancel();
            Deads++;
            return;
        }
    }
}

```

```
class Farmar:public Profese {
public:
    Farmar(char *n):Profese(n) {
    } virtual void prace() {
        Wait(Exponential(40));
        skladObili.vlozit(1);
    }
};
```

```
class Pekar:public Profese {
public:
    Pekar(char *n):Profese(n) {
    }
    virtual void prace() {
        skladMouka.vzit(this);
        skladVoda.vzit(this);
        Wait(Exponential(120));
        skladJidlo.vlozit(5);
    }
};
```



```
for (int pocUredniku = 0; pocUredniku < 100; pocUredniku++) {  
    Print("Pokus %d uredniku", pocUredniku);
```

```
for (int pokus = 0; pokus < 10; pokus++) {  
    Init(0, cCas); Deads=Counter=0;  
    skladJidlo.Clear(); skladVoda.Clear(); .....  
    (new Farmar("Farmar 1"))->Activate();  
    (new Farmar("Farmar 2"))->Activate();  
    (new Vodnik("Vodnik 1"))->Activate();  
    (new Mlynar("Mlynar 1"))->Activate();  
    (new Pekar("Pekar 1"))->Activate();  
    (new Pekar("Pekar 2"))->Activate();
```

```
for (int a = 0; a < pocUredniku; a++)  
    (new Zevloun("urednik"))->Activate();
```

```
Run();
```

```
if (Deads>0) { .....
```

Námět – ekonomický systém

- ◆ Vyjdeme z předchozího modelu
- ◆ Každý výrobce má svoje proměnné náklady (náklady na výrobu). Má peníze.
- ◆ Trh – udává ceny zboží. Pokud je na trhu zboží, jeho cena pomalu klesá. Pokud je poptáváno, jeho cena pomalu stoupá (dva stavy)
- ◆ Výrobce začne vyrábět, když má vstupy a cena na trhu mu pokryje náklady
- ◆ Musí nakupovat jídlo.
- ◆ Jaký bude vývoj cen?

Příště

- ◆ SIMLIB
- ◆ Modelování poruch
- ◆ Simulační experimenty
- ◆ Statistiky