Computer Science Journal of Moldova, vol.**3**, no.1(7), 1995

Modeling of the characteristics of the priority disciplines DD

O.Trofilov G.Mishkoy I.Grama V.Katruck

Abstract

The numerical procedures for evaluating of some characteristics of the priority disciplines DD are developed by using the object oriented programming approach.

1 Introduction

The most often used in the real time systems priority disciplines of the absolute and relative priorities are related to the out-of-system disciplines. For these systems hard conservationism is intrinsic. For example, under the absolute priority discipline a nearly served request of lower priority might be interrupted. Under the relative priority discipline a request, on the contrary, has to wait while the service of a lower priority request has been completed, despite of the fact that its service has just begun. The so-called, mixed priority discipline DD (in [1], [2] and some other papers use is made of the term Discretionary priority Discipline) is more flexible. This discipline was considered by N.Jaswal [1] in analyzing the problem of (without switching) two requests' flows served by one device. For two flows this discipline is described by him as follows:
in case the service time is less than the given value θ the absolute priority is used, otherwise the relative one is. The system $M_r/G_r/1/\infty$ with the mixed priority (with service past break) and with the time a device needs to switch in case the service of a non priority request has been interrupted and in case the device returns to its service was analyzed in [3] by means of [1]. Upon this in above mentioned work

the interrupted switching is assumed to be realized identically anew. The mixed priority queuing model in more general organization has been studied in [4] (some results have been published in [5] too). First, the number of priority classes is assumed to be arbitrary. Second, besides the pattern with past interrupted service of a request and with anew switching, other cases stipulated by further destiny of the interrupted service and the interrupted orientation have been explored. In some practical situations the interruption of a request under service may occur only after "quantum" time of service. Therefore, it is quite natural to consider the discipline symmetrical to the above mentioned one. Namely, to use the relative priority in case the service time is less than θ otherwise the absolute priority is used. For the mixed priority discipline we will use the notation as follows:
`Absolute DD` if the interruptions are allowed, and `Relative DD` if they are not allowed. A more detailed description of these disciplines is given in the next section. Note, nevertheless, that one of the major difficulties in using the mixed priority disciplines is that for their implementation the continuous supervision upon the service process is required (because of the necessity to take into account the service time). The search of the above cases of disciplines with additional conditions of implementation simplicity is justifiable and actual. For example [2], the real time system scheduling procedure reduces to the priority system of $M_r/G_r/1/\infty$ type. In [2] is proved, in particular, that the replacement of the priority system discipline for a simplified one (more convenient for implementation) gives rise to a greater number of interruptions. The analysis of switching losses is made [2] by simulation because, as is mentioned in [2], to account the expenditures related to processor switching analytically is difficult. Nevertheless, though considerable difficulties arising due to non-zero switching turned out to be overcome, the problem of using the obtained results to simulation remains non trivial. This is just the problem of this paper. It consists in software development for the priority system for the evaluation of 'busy period' characteristics.

2 Analytical means

2.1 The Discipline Absolute DD

r independent Poisson flows $L_1, \ldots, L_r$ with parameters $a_1, \ldots, a_r$, respectively, arrive into the queuing system with one channel. The service durations of the requests of L_i flow are independent random variables with distribution functions $B_i(x)$, $i = 1, \ldots, r$. Among the requests from different flows priorities are stated. The flows are numbered in order of priorities decreasing. The priority is a mixed one: in case the time of service of a request from L_k flow, $k = 1, \ldots, r$, is less than the given value θ_k, $k = 2, \ldots, r$ the absolute priority is used, otherwise the relative one is. The orientation (switching) of the device occurs only upon interruptions of service and upon the returns to the interrupted service. So if service of the L_j-request is interrupted by the arrival of the L_i-request, $i < j$, then, first, the orientation of the device to L_i ($\rightarrow i$) will be used. Since the system becomes free from the requests of the priorities higher than j, the orientation ($\rightarrow j$) will be realized, and only then the system will be ready to deal with the interrupted request. One may treat the orientation ($\rightarrow i$) [3] as the time needed to defend the interrupted request (task) from L_j flow, and orientation ($\rightarrow j$) might be treated as the time needed to restore this task. The orientation durations ($\rightarrow k$) are independent random variables with the d.f. $C_k(x)$, $k = 1, \ldots, r$. Random variables C_k and B_i are jointly independent. Any orientation ($\rightarrow k$), of course, might be interrupted by a request of any priority higher than k.

Let's introduce the pattern classification as follows. We denote each pattern by two indexes $I.J$. The former is to point out the destination of the interrupted orientation, the latter is to point out that of the interrupted service. We will assume that in case $J = 1$ the interrupted request is past-break served, in case $J = 2$ this is anew served and in case $J = 3$ this is lost. In case $I = 1$ the interrupted orientation is anew used, in case $I = 2$ it is past-break oriented, and in case $I = 3$ it is identically anew oriented. To take one example, pattern 1.3 means the above queuing system with past-break service of the interrupted request and with identically anew use of the interrupted orientation.

For patterns `I.J` we give the distribution of the busy period duration and that of auxiliary characteristics such as Π_k-period, $\bar{\Pi}_{kk}$-period, k-cycle of service, Π_{kk}-period and other durations. Here and below use is made of the definitions and terms from paper [4].

2.2 Formulae for Absolute DD

The Laplace-Stiltjes transform of the $d.f.$ of k-cycle of service is determined as follows:

for patterns I.1

$$h_k(s) = \int_0^{\theta_k} e^{-(s+\sigma_{k-1}[1-\pi_{k-1}(s)\nu_k(s)])x} dB_k(x) +$$

$$+e^{-(\sigma_{k-1}[\bar{\pi}_{k-1}-\pi_{k-1}(s)\nu_k(s)])\theta_k} \int_{\theta_k}^{\infty} e^{-(s+\sigma_{k-1}[1-\bar{\pi}_{k-1}(s)])x} dB_k(x),$$

for patterns I.2

$$h_k(s) = \left(\int_0^{\theta_k} e^{-(s+\sigma_{k-1})x} dB_k(x) + e^{-\sigma_{k-1}\bar{\pi}_{k-1}(s)\theta_k} \times \right.$$

$$\left. \int_{\theta_k}^{\infty} e^{-(s+\sigma_{k-1}[1-\pi_{k-1}(s)])x} dB_k(x) \right) \times$$

$$\left(1 - \sigma_{k-1}\pi_{k-1}(s)\nu_k(s) \int_0^{\theta_k} e^{-(s+\sigma_{k-1})x} [1 - B_k(x)] dx \right)^{-1},$$

for patterns I.3

$$h_k(s) = \int_0^{\theta_k} e^{-(s+\sigma_{k-1})x} dB_k(x) + e^{-\sigma_{k-1}\pi_{k-1}(s)\theta_k} \times$$

$$\int_{\theta_k}^{\infty} e^{-(s+\sigma_{k-1}[1-\pi_{k-1}(s)])x} dB_k(x)$$

where $\nu_k(s)$ is expressed as follows:

for patterns 1.J

$$\nu_k(s) = \frac{c_k(s+\sigma_{k-1})}{1-\frac{\sigma_{k-1}}{s+\sigma_{k-1}}\Big[1-c_k(s+\sigma_{k-1})\Big]\pi_{k-1}(s)};$$

for patterns 2.J

$$\nu_k(s) = c_k(s+\sigma_{k-1}[1-\pi_{k-1}(s));$$

for patterns 3.J

$$\nu_k(s) = (s+\sigma_{k-1})\int_0^\infty \frac{e^{-(s+\sigma_{k-1})x}\ dC_k(x)}{s+\sigma_{k-1}[1-\pi_{k-1}(s)[1-e^{-(s+\sigma_{k-1})x}]]},$$

but $\bar{\pi}_k(s)$ and $\pi_k(s)$ appeared above is uniquely defined from the system of recurrent functional equations in the following theorem.

Statement 1 *For all the patterns*

$$\begin{aligned}
\sigma_k\bar{\pi}_{k-1}(s) &= \sigma_{k-1}\bar{\pi}_{k-1}(s+a_k(1-\bar{\pi}_{kk}(s))) + a_k\pi_{kk},\\
\sigma_k\pi_k(s) &= \sigma_{k-1}\pi_{k-1}(s+a_k)\\
&\quad +\sigma_{k-1}\{\pi_{k-1}(s+a_k(1-\bar{\pi}_{kk}(s))) - \pi_{k-1}(s+a_k)\}\\
&\quad \times\nu_k(s+a_k(1-\pi_{kk}(s))) + a_k\pi_{kk}(s),\\
\pi_{kk}(s) &= \nu_k(s+a_k(1-\bar{\pi}_{kk}(s)))\bar{\pi}_{kk}(s),\\
\bar{\pi}_{kk}(s) &= h_k(s+a_k(1-\bar{\pi}_{kk}(s))).
\end{aligned}$$

One can find some numerical characteristics of the service from given formulae. We give the first moments of k-cycle of orientation, $\bar{\Pi}_k$-period, $\bar{\Pi}_{kk}$-period, Π_k-period and Π_{kk}-period for patterns I.1, for instance. Let

$$\rho_k = \sum_{i=1}^{k} a_i b_i,$$

where

$$b_1 = \frac{\beta_{11}+c_{11}}{1+a_1c_{11}},$$

$$
\begin{aligned}
b_i &= \beta_{i1} + [\theta_i - \int_0^{\theta_i} B_i(x)dx][\Phi_{i-1}\dots\Phi_2(1+a_1c_{11})q_i - 1], \\
\Phi_1 &= 1, \quad \Phi_i = 1 + \pi_{i-1}(a_i)[q_i - 1], \\
q_i &= \frac{1}{c_i(\sigma_{i-1})}, \quad i = 2, ..., k.
\end{aligned}
$$

Statement 2 *If* $\rho_k < 1$ *then*

$$
\begin{aligned}
\sigma_k\pi_{k1} &= \frac{\Phi_k\dots\Phi_2(1+a_1c_{11}) + \rho_k - 1}{1-\rho_k} \\
\bar{\pi}_{kk1} &= \frac{b_k}{1-\rho_k}, \\
h_{k1} &= \frac{b_k}{1-\rho_{k-1},} \\
\sigma_k\bar{\pi}_{k1} &= \frac{\rho_k}{1-\rho_k,} \\
\nu_{k1} &= \frac{1}{\sigma_{k-1}}[q_k - 1]\frac{\Phi_{k-1}\dots\Phi_2(1+a_1c_{11})}{1-\rho_{k-1}}
\end{aligned}
$$

2.3 The discipline Relative DD

The description in section 2.1 remains in force but with following changes: in case time of service of the k-th priority request is less than given value θ_k ($k = 2, \dots, r$) the arrived request of the priority which is higher than k is assigned the relative priority, otherwise the absolute one is.

All the notations and definitions introduced above are remained in force as well. Let's in addition

$$
\begin{aligned}
\bar{\gamma}_k(s) &= s + \sigma_k[1 - \bar{\pi}_k(s)], \\
\gamma_k(s) &= s + \sigma_k[1 - \pi_k(s)\nu_{k+1}(s)].
\end{aligned}
$$

The Laplace-Stiltjes transform of the d.f. of k-cycle of service is determined as follows:

for patterns I.1 $h_k(s) =$

$$
\frac{\int_0^{\theta_k} e^{-(\bar{\gamma}_{k-1}(s))x}dB_k(x) + e^{-\sigma_{k-1}\pi_{k-1}\theta_k}\int_{\theta_k}^{\infty} e^{-(s+\sigma_{k-1})x}dB_k(x)}{1 - \sigma_{k-1}\pi_{k-1}(s)\nu_k(s)e^{\sigma_{k-1}\bar{\pi}_{k-1}(s)\theta_k}\int_{\theta_k}^{\infty} e^{-(s+\sigma k-1)x}[1-B_k(x)]dx}
$$

for patterns I.2 $h_k(s) =$

$$\int_0^{\theta_k} e^{-\bar{\gamma}_{k-1}(s)x} dB_k(x) + e^{-(\bar{\gamma}_{k-1}(s)-\gamma_{k-1}(s))\theta_k} \int_{\theta_k}^{\infty} e^{-\gamma_{k-1}(s)x} dB_k(x),$$

where $\nu_k(s)$ for each of the patterns has the same form as for the Absolute DD discipline but $\bar{\pi}_k(s)$ is written in place of $\pi_k(s)$. The values $\bar{\pi}_{k-1}(s)$ and $\pi_{k-1}(s)$ appeared above are uniquely defined from the system of recurrent functional equations in the following theorem.

Statement 3 *The Laplace-Stiltjes transform $\sigma\pi(s)=\sigma_r\pi_r(s)$ of the busy period distribution function is determined from the following recurrent equations:*

$$\begin{aligned}
\sigma_k\bar{\pi}_k(s) &= \sigma_{k-1}\bar{\pi}_{k-1}(s + a_k(1 - \bar{\pi}_{kk}(s))) + a_k\bar{\pi}_{kk}(s), \\
\sigma_k\pi_k(s) &= \sigma_{k-1}\pi_{k-1}(s + a_k(1 - \bar{\pi}_{kk}(s))) + a_k\pi_{kk}(s), \\
\pi_{kk}(s) &= \nu_k(s + a_k(1 - \bar{\pi}_{kk}(s)))\bar{\pi}_{kk}(s), \\
\bar{\pi}_{kk}(s) &= h_k(s + a_k(1 - \bar{\pi}_{kk}(s))).
\end{aligned}$$

One can find the first moments for the above durations. ρ_k is determined in the same manner as for the Absolute DD discipline. We have, for instance, for pattern I.2

$b_1=\beta_{11}$,
and $b_i, \quad i = 2, \ldots, r$ are determined as follows:

$$b_i = \theta_i[1 - B_i(\theta_i)] + \int_0^{\theta_i} x dB_i(x) + \Big(a_1c_{11} + q_i + \sum_{j=2}^{i-1} \frac{a_j}{\sigma_{j-1}}(q_j - 1)\Big) \times$$

$$\Big(\int_{\theta_i}^{\infty} e^{-\sigma_{i-1}} dB_i(x) - \theta_i[1 - B_i(\theta_i)]\Big),$$

where $q_i = 1 + \sigma_{i-1}c_{i1}$, $i = 2, \ldots, r$.

Statement 4 *if* $\rho_k < 1$ *then*

$$\begin{aligned}
\nu_{k1} &= \frac{q_k - 1}{\sigma_{k-1}(1-\rho_{k-1})}, \\
h_{k1} &= \frac{b_k}{1-\rho_k}, \\
\bar{\pi}_{kk1} &= \frac{b_k}{1-\rho_k}, \\
\sigma_k \bar{\pi}_{k1} &= \frac{\rho_k}{1-\rho_k}, \\
\pi_{kk1} &= \frac{b_k \sigma_{k-1} + q_{k-1}}{\sigma_{k-1}\rho_{k-1}}, \\
\sigma_k \pi_{k1} &= \frac{a_1 c_{11} + \rho_k + \sum_{i=2}^{k} \frac{a_i}{\sigma_{i-1}}(q_i - 1)}{1-\rho_k}.
\end{aligned}$$

3 A modified approach to the programming organization of a priority queuing system

3.1 The description of the priority system service PSS

In this section we give the description of a queuing system with priority and orientation by means and methods of object- oriented programming approach. For this purpose the hierarchy of objects, describing our system, had been developed earlier in [6] but while making use of the objects at the stage of building numerical procedures and dialogue's support environment the possibility of the more transparent description of the system with the help of interconnected objects was revealed.

Above all, we expand the basic object `RV`, corresponding to the abstract random variable (r.v.):

Object type

$$\texttt{RV} = \begin{cases} \textit{Ancestor}: & \\ \textit{Fields}: & \texttt{RegisterNo,ParamNum;} \\ \textit{Methods}: & \texttt{Init;} \\ & \texttt{Done;virtual;} \\ & \texttt{DistrF(t);virtual;} \\ & \texttt{DensityF(t);virtual;} \\ & \texttt{LaplaceF(t);virtual;} \\ & \texttt{DLaplaceF(t);virtual;} \\ & \texttt{Mom1;virtual;} \\ & \texttt{GetParam(j);virtual;} \\ & \texttt{SetParam(j,x);virtual;} \end{cases}$$

Description:

- `Init, Done`: empty constructor and destructor,
- `DistrF(t)`: method calculating the inversion of the Laplace-Stieltjes transform of `LaplaceF(t)`,
- `DensityF(t)`: method calculating the inversion of the Laplace transform of `LaplaceF(t)`,
- `LaplaceF(t)`: method calculating the Laplace-Stieltjes transform of `DistrF(t)`,
- `DLaplaceF(t)`: method calculating the Laplace- Stieltjes transform of `DensityF(t)`,
- `Mom1`: empty method calculating math. expectation.

The field `RegisterNo` keeps the registration number of a r.v. corresponding to its distribution and the field `ParamNum` informs on the number of distribution parameters. The methods `GetParam(j)`, `SetParam(j,x)`, from which the former returns and the latter sets the value of the j parameter, are added. This abstract object type does not represent a really existing r.v. to construct the object types related with the r.v. of the desired kind to be used in modeling the queuing system. To take

an example [6], one can construct an object type`MyRV` corresponding to the r.v. with the known Laplace- Stieltjes transform. To access, for example, the value of d.f. `DistrF(t)` for a variable of this type we write simply `MyRV.DistrF(t)` to call the inherited method inverting the Laplace- Stieltjes transform of `MyRV.LaplaceF(t)`.

The object type `ExpRV` related to the exponential distribution is modified by adding two of the latter methods:

Object type

$$\texttt{ExpRV} = \begin{cases} \textit{Ancestor}: & \texttt{RV;} \\ \textit{Fields}: & \texttt{Intensity;} \\ \textit{Methods}: & \texttt{DistrF(t);virtual;} \\ & \texttt{DensityF(t);virtual;} \\ & \texttt{LaplaceF(t);virtual;} \\ & \texttt{DLaplaceF(t); virtual;} \\ & \texttt{Mom1; virtual;} \\ & \texttt{GetParam(j); virtual;} \\ & \texttt{SetParam(j,x); virtual;} \\ & \texttt{GetIntensity;} \\ & \texttt{SetIntensity(x);} \end{cases}$$

`GetIntensity` returns and `SetIntensity` sets intensity of the exponential distribution. They duplicate the implementation of `GetParam` and `SetParam` for this distribution but are added merely for convenience. The organization of the priority system structure in the form of priority level collection appears to be highly natural and convenient. Namely, as mentioned above, the input queue is formed by all arriving requests while the PSS is busy. It includes r flows corresponding to the priorities 1,...,r. The time length between two consecutive arrivals of requests of the same flow is the exponential r.v. with parameter Intensity. Before proceeding to serve the request of priority k the PSS needs an orientation time period for preparation, if the previous request's priority differs from k. This duration is really proved to be a r.v. The orientation times corresponding to different priorities are distinct. After the orientation the PSS device proceeds to serve the request and, what's more, the service duration carries random character as well.

Thus, to define the priority level we use object type `PriorLevel` with the structure as follows:

Object type

`PriorLevel` = {

Ancestor :	
Fields :	`InputFlow, Orient, Service, Teta;`
Methods :	`Init;`
	`Done;`
	`SetInputFlow(InpFlow);`
	`GetInputFlow;`
	`SetOrient(RandVar1);`
	`GetOrient;`
	`SetService(RandVar2);`
	`GetService;`
	`SetTeta(tt);`
	`GetTeta;`

}

Description:

- `InputFlow`: pointer to an object of type `ExpRV`,
- `Orient`: pointer to an object of type `MyRV`,
- `Service`: pointer to an object of type `MyRV`,
- `Teta`: parameter,
- `SetInputFlow(InpFlow)`: sets `InputFlow`=`InpFlow`,
- `GetInputFlow`: returns `InputFlow`,
- `SetOrient(RandVar1)`: sets `Orient`=`RandVar1`,
- `GetOrient`: returns `Orient`,
- `SetService(RandVar2)`: sets `Service`=`RandVar2`,
- `GetService`: returns `Service`,
- `SetTeta(tt)`: sets `Teta`=`tt`,

- `GetTeta`: returns `Teta`.

We define the object type `PriorLevelColl` for the collection of priority levels:

Object type

$$\texttt{PriorLevelColl} = \begin{cases} \textit{Ancestor}: & \texttt{TCollection;} \\ \textit{Fields}: & \\ \textit{Methods}: & \texttt{FreeItem(P);virtual;} \end{cases}$$

Description:

- `FreeItem(P)`: delete a priority level from the collection by pointer `P`.

Now we are at the point to define the object type `Status` whose fields and methods will contain some information about the current state of the PSS and serve to use and change this information. This type had already been presented in [6] but now it suffers some changes due to our new approach to the interpretation of the priority system structure.

Object type

$$\texttt{Status} = \begin{cases} \textit{Ancestor}: & \\ \textit{Fields}: & \texttt{PriorDisc; OrientDisc;} \\ & \texttt{ServiceDisc; Regime;} \\ \textit{Methods}: & \texttt{Init; Done;virtual;} \\ & \texttt{SetPriorDisc(APD);} \\ & \texttt{SetServiceDisc(ASD);} \\ & \texttt{SetOrientDisc(AOD);} \\ & \texttt{SetRegime(iR);} \\ & \texttt{GetPriorDisc;} \\ & \texttt{GetServiceDisc;} \\ & \texttt{GetOrientDisc;} \\ & \texttt{SetRegime(Reg);} \\ & \texttt{GetRegime;} \end{cases}$$

Description:

- `PriorDisc`: keeps the priority discipline,
- `OrientDisc`: keeps the orientation discipline,
- `ServiceDisc`: keeps the service discipline,
- `Regime`: keeps the parameter Regime,
- `Init`: installs the default values for all fields,
- `Done`: destroys the object,
- All other methods install or return the values of corresponding parameters.

The necessity of the field `PriorDisc` to be added due to the appearance of some combinations of the kind (`ServiceDisc`, `OrientDisc`) which appear both in the discipline with `Absolute` priority and these of `DD`- class (`Absolute DD, Relative DD`) and it is the point that requires more precision. Generally, following [4], the priority discipline may be `Absolute, SemiAbsolute, Relative, SemiRelative, Absolute DD, Relative DD`; the mode determining the behavior of the system, while it is idle, has three alternatives: `Reset`, `LookAhead`, `WaitMostProb`.

So, having defined object types `PriorLevelColl` and `Status` we would construct now the object type `PSS` containing full background information on the priority queuing system with orientation:

Object type

$$\texttt{PSS} = \begin{cases} \textit{Ancestor}: & \texttt{Status;} \\ \textit{Fields}: & \texttt{Coll;} \\ \textit{Methods}: & \texttt{Init(Dim);} \\ & \texttt{Done;virtual;} \\ & \texttt{PriorLevel(i);} \end{cases}$$

Description:

- `Coll`: pointer to an object of type `PriorLevelColl`,

- `Init(Dim)`: installs the priority system with `Dim` priorities,
- `Done`: destroyed the object,
- `PriorLevel(i)`: returns the pointer to `i` priority level.

3.2 Object type BusyPeriod

We introduce object type `BusyPeriod` to provide a background for busy period characteristics. It includes the methods `Fast`, `FastIter1`, `FastIter2` which are of crucial importance in the calculation of all characteristics.

Object type

$$\texttt{BusyPeriod} = \begin{cases} \textit{Ancestor}: & \texttt{RV;} \\ \textit{Fields}: & \texttt{PSS;} \\ & \texttt{a, sigma;} \\ & \texttt{cnt, InitpkkDimension, Initpkk;} \\ & \texttt{pks, \_pks, \_pkks;} \\ & \texttt{pk1s1, pk1s2, \_pk1s2;} \\ & \texttt{precision, l, averarr;} \\ \textit{Methods}: & \texttt{Fast(k,s);} \\ & \texttt{FastIter(k,s,y1);} \\ & \texttt{FastIter2(k,s,y1,y2;} \\ & \texttt{n(k,s,x);} \\ & \texttt{h(k,s,x,bx,y);} \\ & \texttt{Integral1(k,t);} \\ & \texttt{Integral2(k,alfa,beta);} \\ & \texttt{Averadge1(k);} \\ & \texttt{Averadge5(k);} \\ & \texttt{Averadge6(k);} \end{cases}$$

Description:

- `PSS`: pointer to the object of type `PSS`,
- `a,sigma`: the arrays of input intensities and their partial sums,

- `cnt, InitpkkDimension, Initpkk`: some quantities for use in methods `Fast, FastIter1, FastIter2,`
- `pks`, `_pks`, `_pkks`, `pk1s1`, `pk1s2`, `_pk1s2` : the values $\pi_k(s)$, $\bar{\pi}_k(s)$, $\bar{\pi}_{kk}(s)$, $\pi_{k-1}(s+a_k)$, $\pi_{k-1}(s+a_k(1-\bar{\pi}_{kk}(s)))$, $\bar{\pi}_{k-1}(s+a_k(1-\bar{\pi}_{kk}(s)))$, respectively,
- `l`: selected priority,
- `averarr`: array of math. expectations π_{k1}, $\bar{\pi}_{k1}$, $\bar{\pi}_{kk1}$, π_{kk1}, h_{k1}, ν_{k1},
- `Init, Done`: constructor and destructor of an object of type `BusyPeriod`,
- `SetLevel(k)`: sets `l`=`k`,
- `Fast, FastIter1, FastIter2, n, h`: produce background calculation for all busy period characteristics,
- `Integral1, Integral2`: the methods for approximate calculating of integrals occurring in the formulae for `h` and the expectations,
- `Average1, Average5, Average6`: procedures for calculating the stationary mode characteristics of the system with `Absolute, Absolute DD, Relative DD` priority respectively.

3.3 Object types related to busy period characteristics

We treat any busy period characteristics as r.v. related to some object of type `BusyPeriod`. This allows the characteristics to access the fields and methods of the object for its own use. Now we define object types corresponding to some specific busy period characteristics. For this we only need to redefine the inherited methods `LaplaceF` properly.

Object type

$$\texttt{pk} = \begin{cases} Ancestor: & \texttt{BusyPeriod;} \\ Fields: & \\ Methods: & \texttt{LaplaceF(t);virtual;} \\ & \texttt{Mom1;virtual;} \end{cases}$$

Description:

- `FreeItem(P)`: delete a priority level from the collection by pointer `P`,
- `LaplaceF( t )`: runs the inherited procedure `Fast(k,t)` of the object type `BusyPeriod` and returns `pks`,
- `Mom1`: runs one of the methods Average in dependence of the kind of the priority discipline and returns either value averarr[1] where the value of math. expectation π_{k1} is entered or value -1 if the stationary mode is absent.

Object type

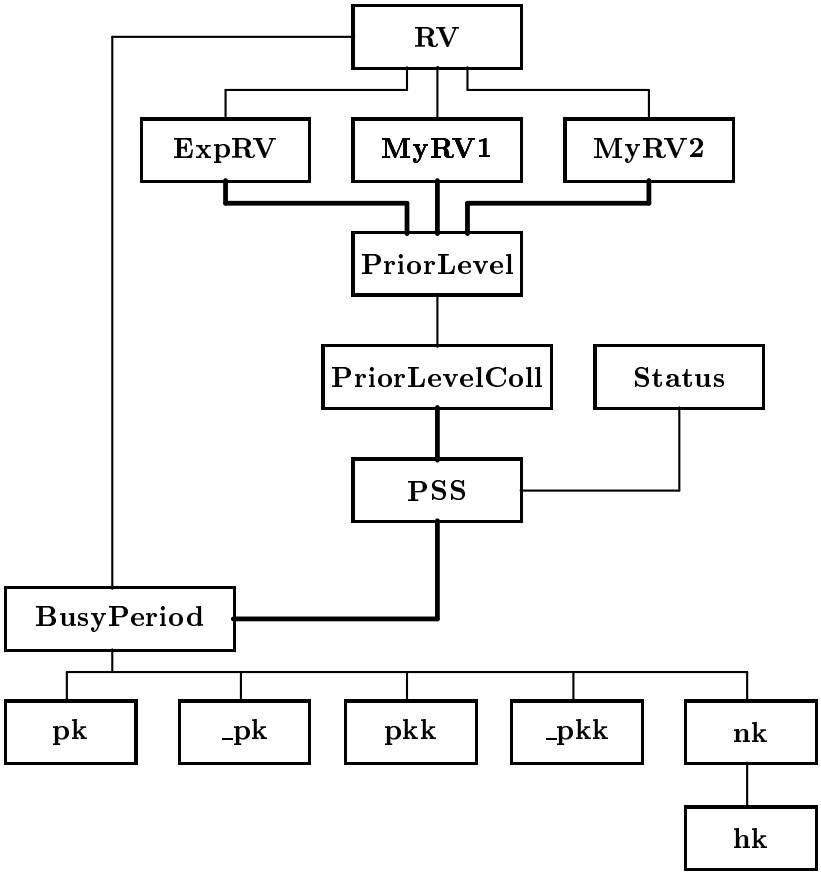
$$\texttt{nk} = \begin{cases} Ancestor: & \texttt{BusyPeriod;} \\ Fields: & \\ Methods: & \texttt{LaplaceF(t);virtual;} \\ & \texttt{Mom1;virtual;} \end{cases}$$

Description:

- `FreeItem(P)`: delete a priority level from the collection by pointer `P`,
- `LaplaceF( t )`: runs either the inherited procedure `n(l,t,pks)`, if the priority discipline is `Absolute` or `Absolute DD`, or `n(l,t,_pks)`, if the discipline is `Relative DD`,
- `Mom1`: runs one of the methods Average in dependence of the kind of the priority discipline and returns either value averarr[5] where the value of math. expectation ν_{k1} is entered or value -1 if the stationary mode is absent.

For other characteristics the similar object types are available. The

connection between object types is presented in the figure below.



The thin line shows the origin of the object while the thick line shows what objects it contains.

4 Numerical results

The above object types were used to evaluate the d.f. and Laplace-Stiltjes transforms of the d.f. of Π_k-period, $\bar{\Pi}_k$-period, Π_{kk}-period, $\bar{\Pi}_{kk}$-period, k-cycle of service, k-cycle of orientation and the expectations of these durations. In the following tables we present some numerical results. All calculations were performed at the computer IBM PC 286.

Table 1:

s	$\pi_k(s)$	$\bar{\pi}_k(s)$	$\pi_{kk}(s)$	$\bar{\pi}_{kk}(s)$	$h_k(s)$	$\nu_k(s)$
.00001	0.99953	0.99958	0.99927	0.99959	0.99971	0.99978
.01	0.95651	0.96107	0.93346	0.96194	0.97200	0.97828
.6	0.32253	0.32952	0.22514	0.44588	0.46314	0.52267
1.22	0.17899	0.18150	0.11870	0.31942	0.32308	0.38204
3.47	0.05221	0.05246	0.03293	0.16550	0.16814	0.20239
5.63	0.02562	0.02568	0.01584	0.11411	0.11557	0.14053
10.13	0.00979	0.00980	0.00593	0.06944	0.07000	0.08602
20.71	0.00273	0.00273	0.00162	0.03619	0.03635	0.04503

Table 2:

s	$\pi_k(s)$	$\bar{\pi}_k(s)$	$\pi_{kk}(s)$	$\bar{\pi}_{kk}(s)$	$h_k(s)$	$\nu_k(s)$
.00001	0.99954	0.99978	0.99920	0.99944	0.99964	0.99985
.01	0.95816	0.97952	0.92960	0.95032	0.96591	0.98529
.6	0.33163	0.60851	0.23920	0.43890	0.45510	0.56512
1.22	0.18267	0.46425	0.12574	0.31957	0.32888	0.40531
3.47	0.05263	0.25864	0.03393	0.16678	0.16974	0.20707
5.63	0.02573	0.18317	0.01612	0.11481	0.11629	0.14224
10.13	0.00981	0.11440	0.00598	0.06968	0.07025	0.08645
20.71	0.00274	0.06093	0.00163	0.03624	0.03640	0.04509

Table 3:

t	$\Pi_k(t)$	$\bar{\Pi}_k(t)$	$\Pi_{kk}(t)$	$\bar{\Pi}_{kk}(t)$	$H_k(t)$	$N_k(t)$
0.00001	0.00000	0.00000	0.00000	0.00000	0.00002	0.00001
0.01	0.00007	0.00007	0.00004	0.00795	0.00795	0.00993
0.6	0.12954	0.12995	0.08173	0.32711	0.33450	0.39790
1.22	0.30900	0.31222	0.20264	0.48320	0.49972	0.57504
3.47	0.65223	0.67226	0.46742	0.69536	0.73004	0.79541
5.63	0.77623	0.80336	0.60758	0.78794	0.83266	0.88301
10.13	0.88243	0.90741	0.77396	0.88606	0.93469	0.96052
20.71	0.96031	0.97151	0.92199	0.96320	0.99233	0.99679
Mom1	9.70943	3.75460	7.70946	4.45670	3.08279	3.37500

Table 4:

t	$\Pi_k(t)$	$\bar{\Pi}_k(t)$	$\Pi_{kk}(t)$	$\bar{\Pi}_{kk}(t)$	$H_k(t)$	$N_k(t)$
0.00001	0.00000	0.00004	0.00000	0.00002	0.00002	0.00001
0.01	0.00007	0.01370	0.00004	0.00795	0.00795	0.00993
0.6	0.13026	0.49728	0.08362	0.33122	0.33874	0.40915
1.22	0.31433	0.68936	0.21507	0.48728	0.50381	0.61859
3.47	0.67769	0.87317	0.50211	0.66443	0.69426	0.88751
5.63	0.80452	0.91766	0.63194	0.74264	0.78367	0.95667
10.13	0.90079	0.95364	0.77454	0.83953	0.89445	0.99259
20.71	0.96185	0.98101	0.90496	0.93196	0.97960	0.99999
Mom1	4.58904	2.24315	10.2117	5.63928	4.16667	1.50000

In the tables 1 and 2 we give the evaluations of the Laplace-Stiltjes transform of the d.f. corresponding to the characteristics of the Absolute DD and Relative DD disciplines respectively, but in the tables 3, 4 the d.f. corresponding to the above characteristics are evaluated.

References

[1] Jaiswal N.K. Priority queues. Academic Press, New York, 1968.

[2] Bogunovic N. Processes scheduling procedure for a class of real time computer systems. IEEE Trans. Ind. Electon., 1987, vol.34, No.1, p.29-34.

[3] Volkovinsky M.N., Kabalevsky A.N. The mixed priority service in the systems with the switching losses. Automatics and telemechanics, 1975, No.11, p.16-22. (Russian)

[4] Mishkoy G.K. Priority systems with orientation, their software and some applications. Doctor thesis, 1990 (Russian)

[5] Mishkoy G.K. Some results for the priority discipline DD with non-zero switching. Bulletin Ac. Sci. R.M., 1993, No.3, p.91-94.

[6] Grama I., Mishkoy G. The object oriented programming for queuing system. Computer Sci. Journ. of Moldova, 1993, vol.1, No.1, p.85-104

O.Trofilov, G.Mishkoy, I.Grama, V.Katruck
Institute of Mathematics,
Academy of Sciences of Moldova,
5 Academiei str.,
Chişinău, 277028, Moldova

Received 4 March, 1995