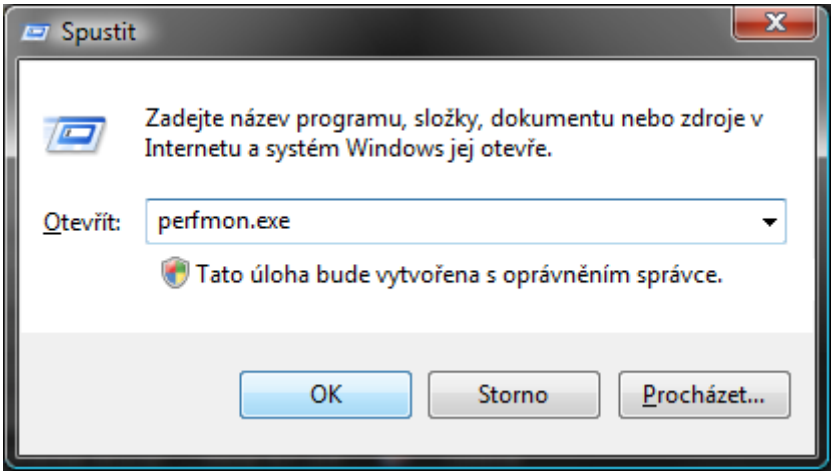
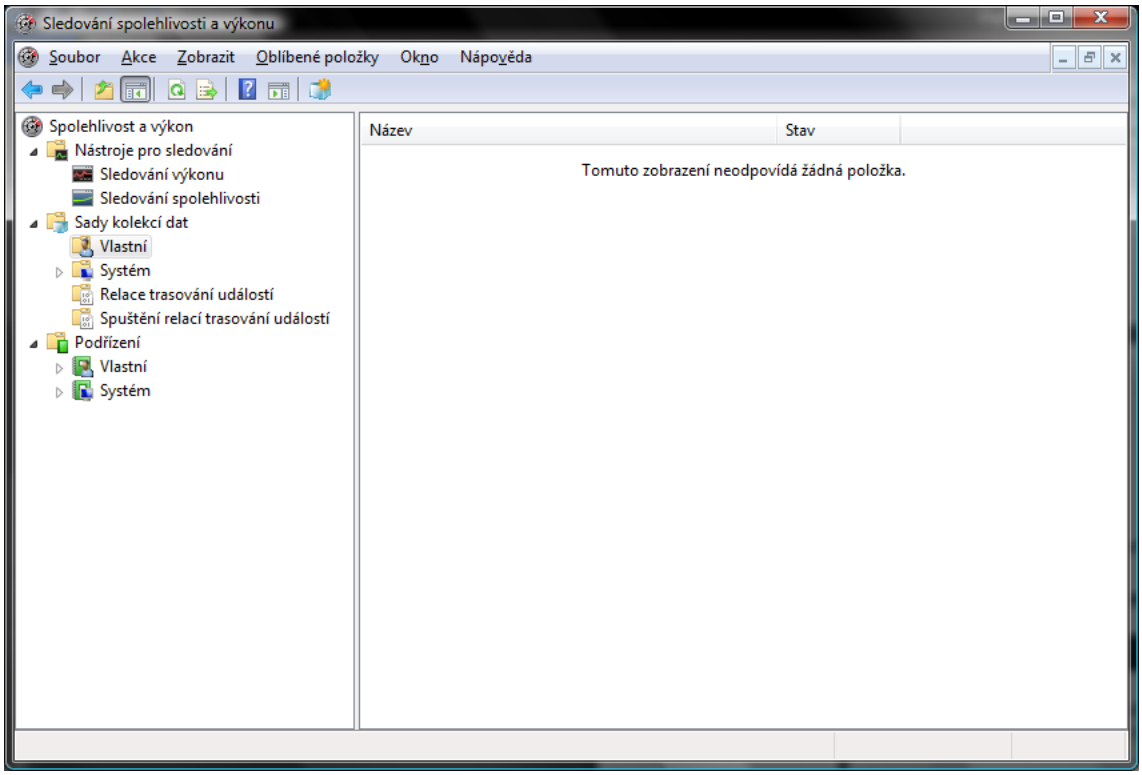
Príloha I: Nastavení aplikace perfmon.exe

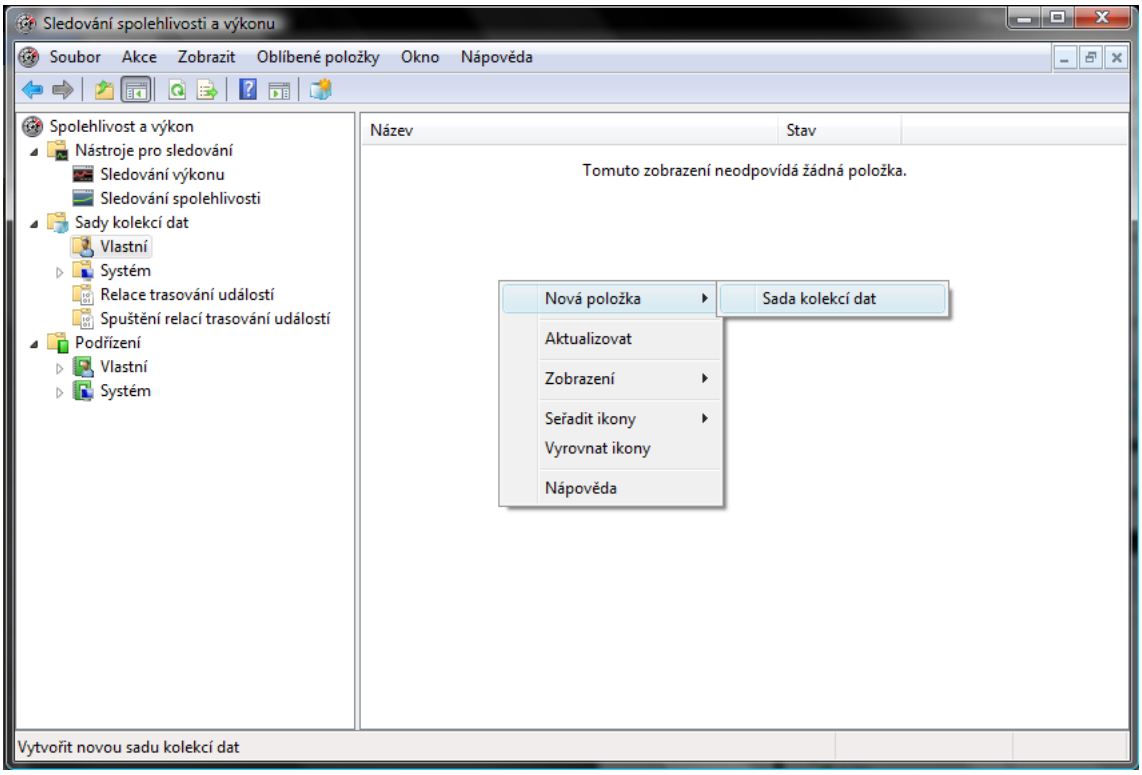
1. Spustíme aplikaci pomocí příkazu **perfmon.exe**



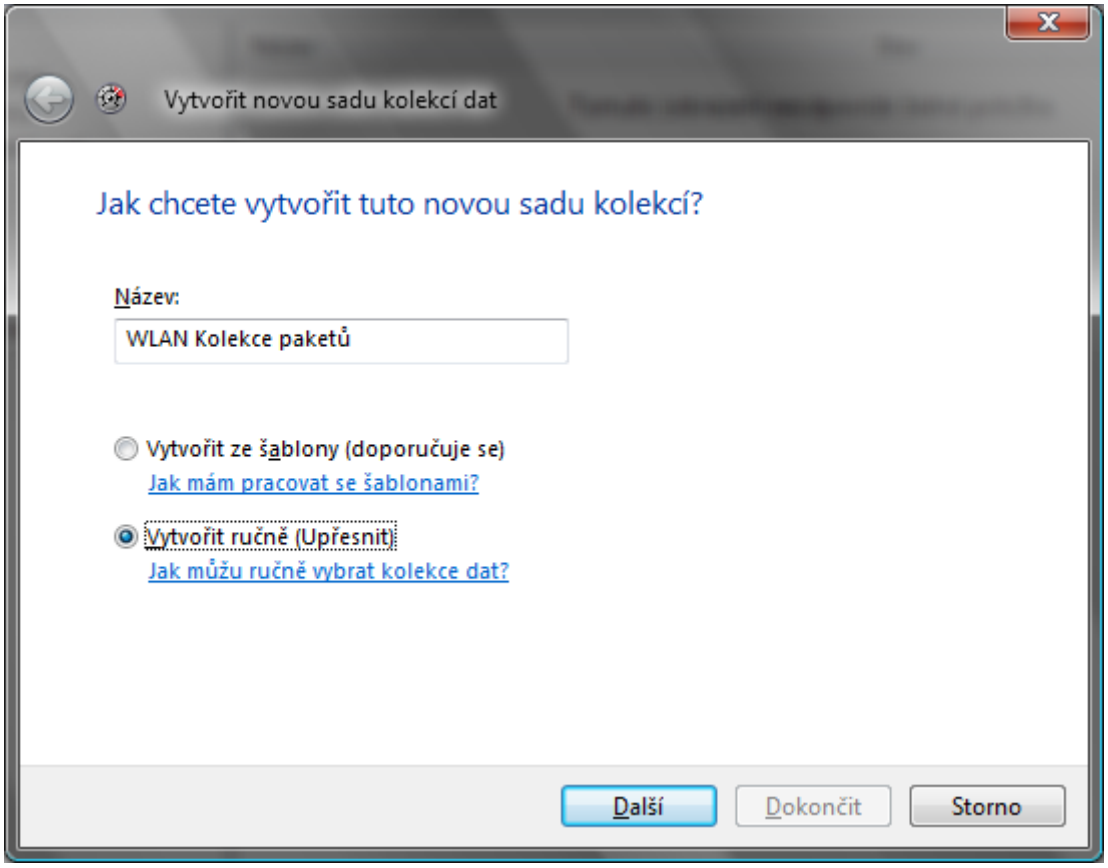
2. Klikneme na **„Sady kolekce dat“** a poté na **„Vlastní“**



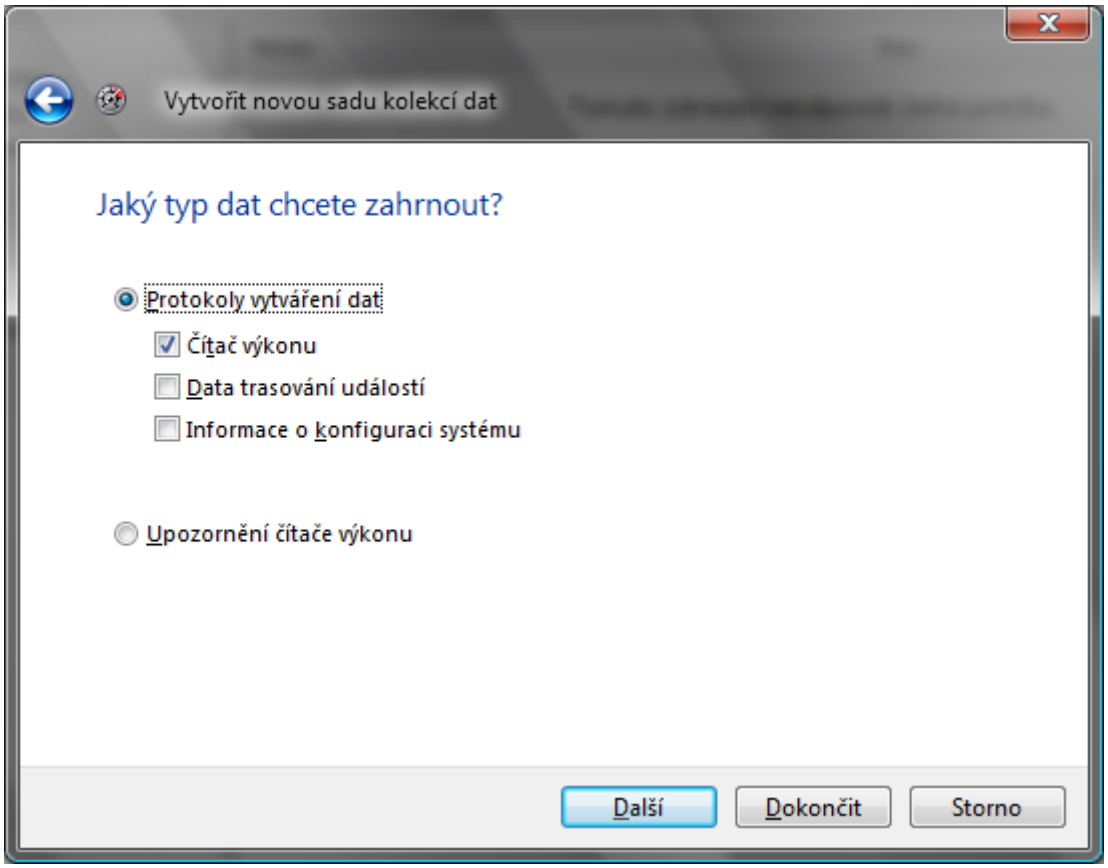
3. Klikneme pravím tlačítkem myši na prázdnou část levé části okna a vybereme z menu **„Nová položka“** a pak **„Sada kolekcí dat“**.



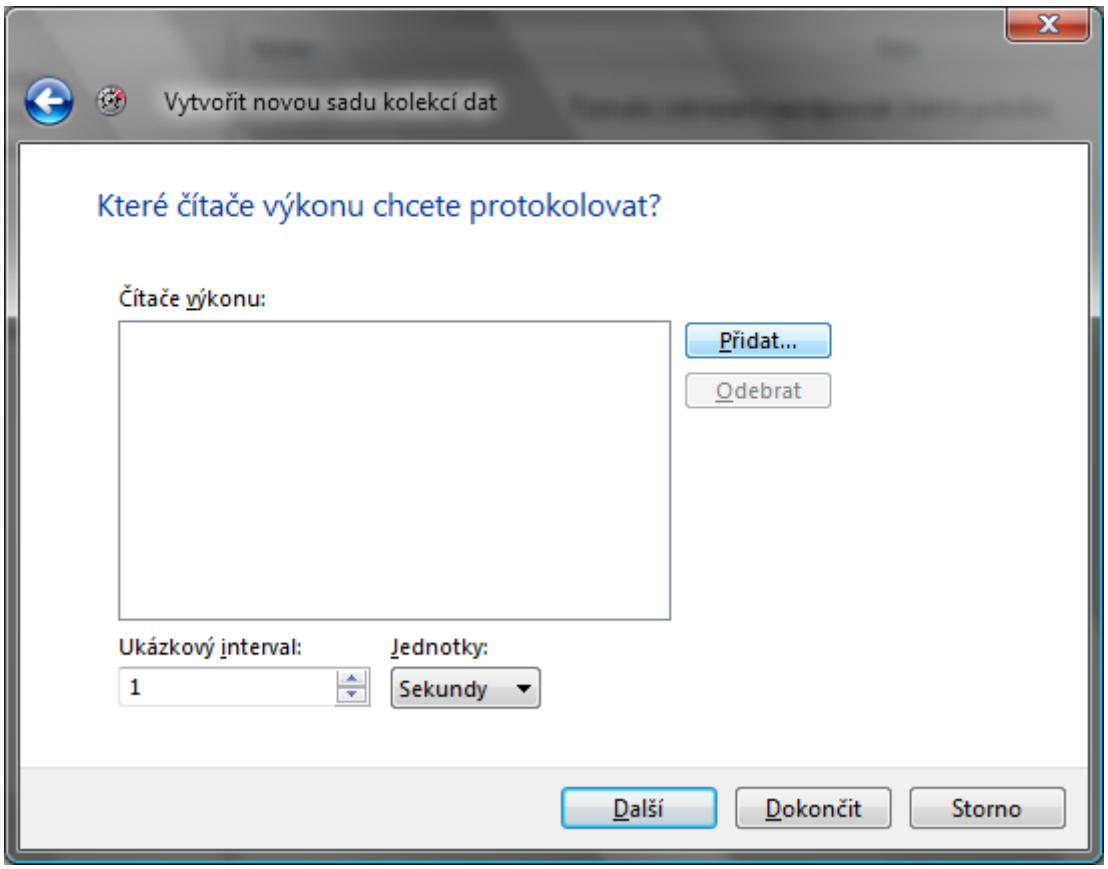
4. Teď zadáme nějaký **název** pro naši sadu kolekce, pak klikněme na **„Další“.**



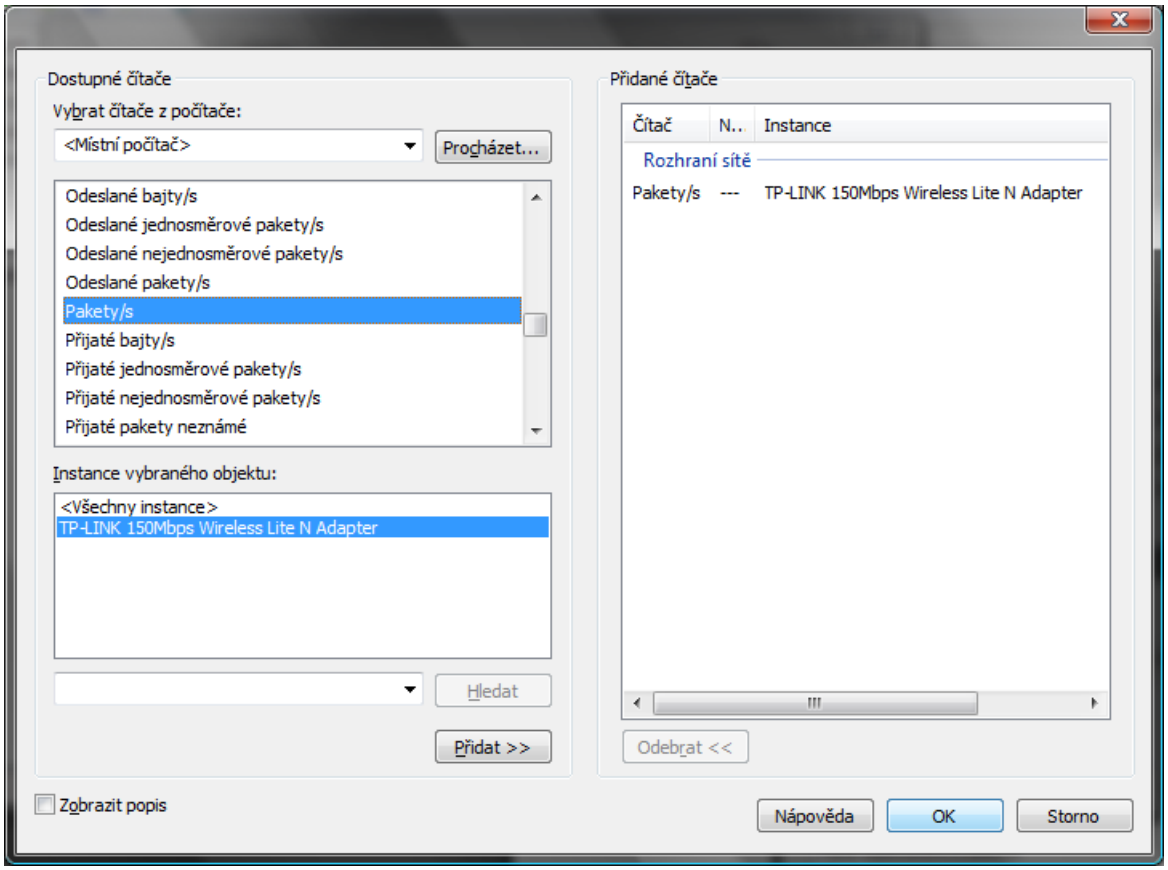
5. Tady nastavíme **protokol** na použití vytváření dat, tj. zatrhneme **„Čítač výkonu“** a klikneme na **„Další“.**

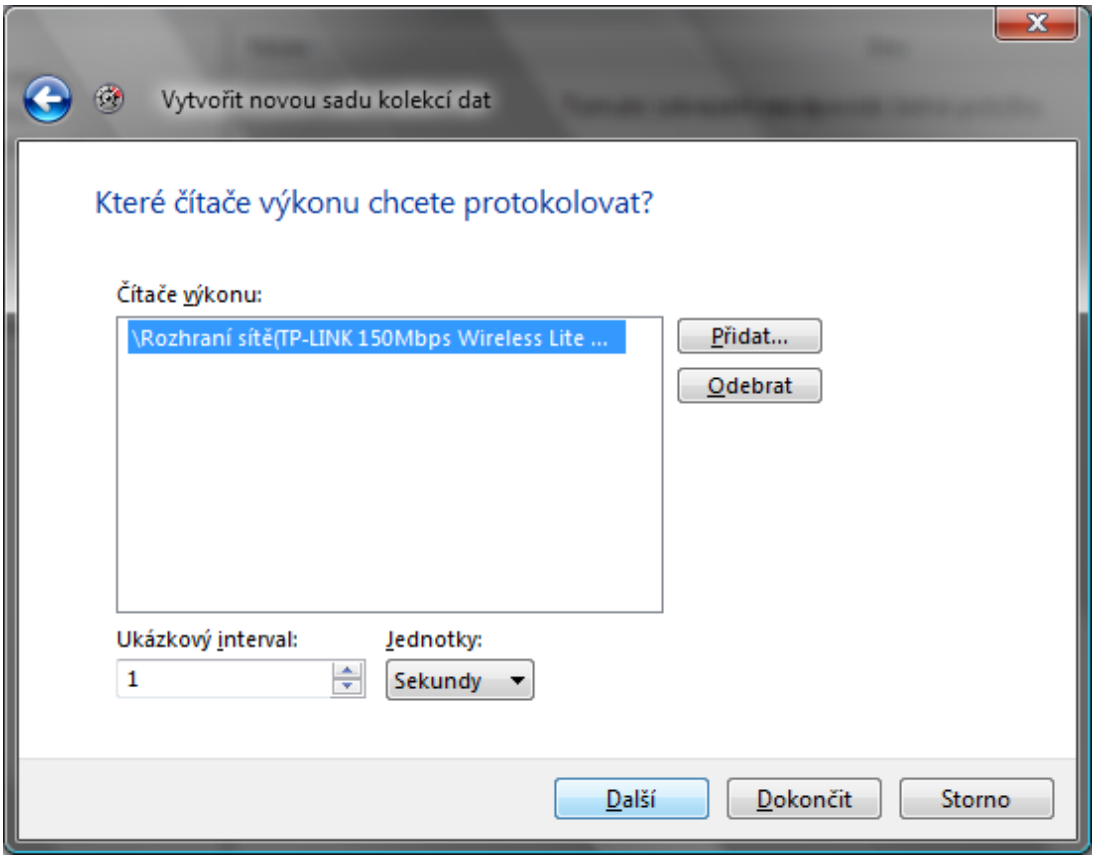


6. Teď se nastaví **časoví interval** pro kolekci a klikne se na tlačítko **„Přidat“.**

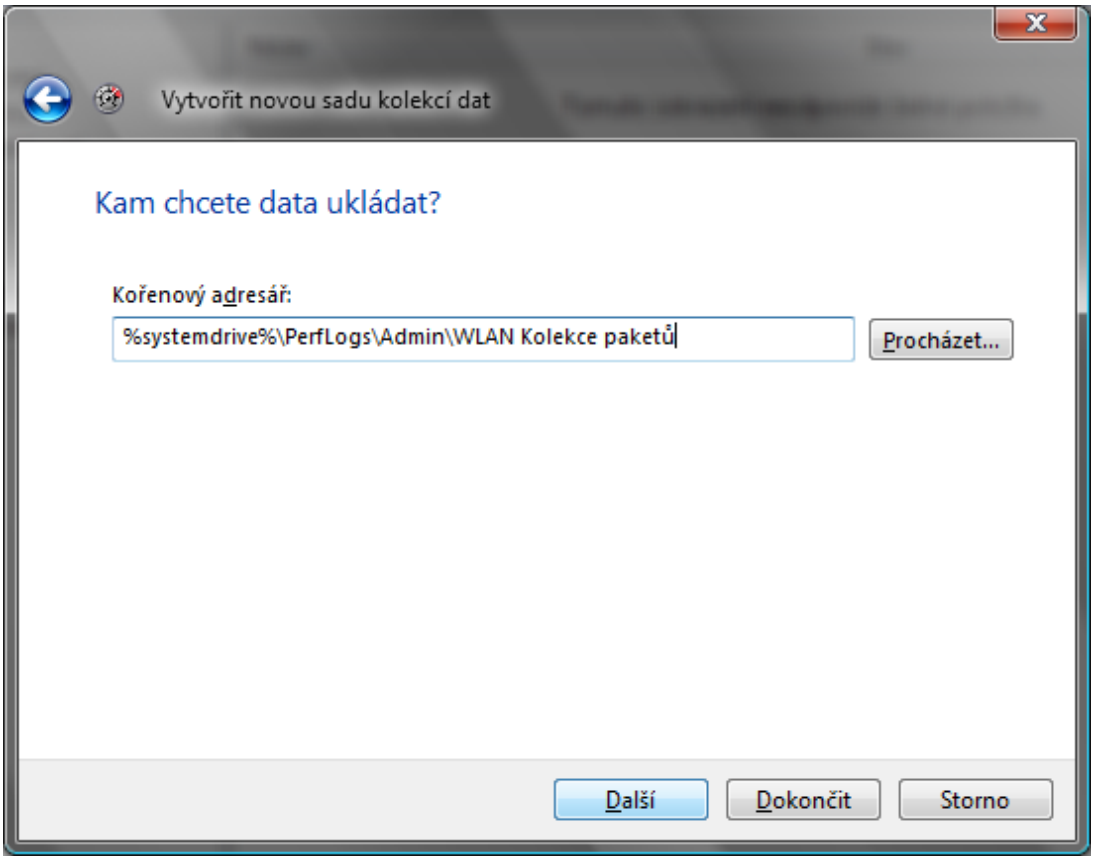


7. Vyhledáme a vybereme požadovaný čítač z levého horního okýnka (v našem případě **„Rozhraní sítě“**) a rozklikneme ji. Pak vyhledáme čítač **Pakety/s** a vybereme ji, poté v levém dolním okýnku vybereme požadovaný **interface** (síťová karta, na kterém budeme provádět kolekci dat) a klikneme na tlačítko **„Přidat“** a **„OK“**.

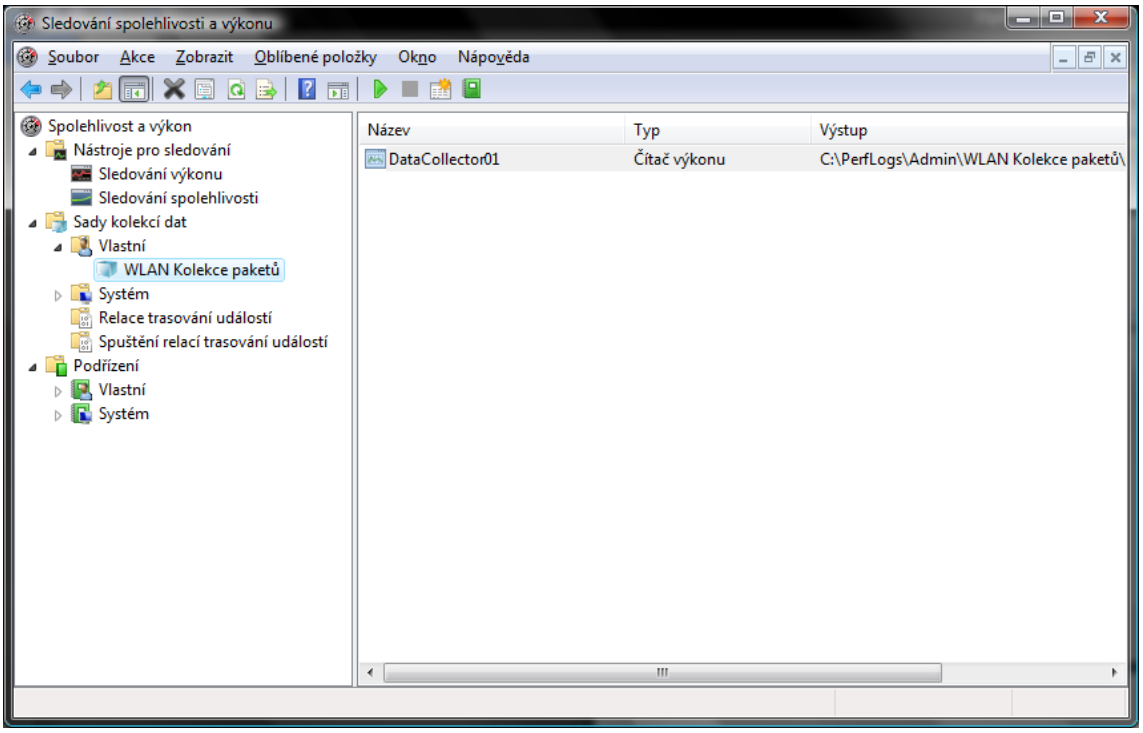




8. Už nám jenom zbývá vybrat umístnění, že kam chceme uložit datový soubor a dokončit průvodce.



9. Pak už kolekci můžeme spustit s malým zeleným tlačítkem **„Play“** na příkazovém panelu s nástroji.



Poznámka: Je ještě hromada možností nastavení výstupního adresáře, bezpečnosti, plánování, zastavovací podmínky a také spouštění požadované aplikace. S tímhle se dá bezprostředně zautomatizovat predikční proces. [9]

Príloha II: Filtrovací algoritmus – kód

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Filter the input file to be readable with UI_Prediction application %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc;
clear all;
[File,Path] = uigetfile('*.tsv','Tab separated value (*.tsv)'); %opening
the tab separated file containing the collected data
fID=fopen(strcat(Path,File)); %gets the file ID
r=fscanf(fID,'%c'); %scan the file as input string
s=regexp(r,'\t|\"','split'); %split the string into cells
q=str2double(s); %convert the cells to double values
t=size(q); %get the size of matrix containing the converted values
t=t(1,2);
j=0;
for i=1:t
    if q(i)>=0 %filtering the converted values matrix for usable numeric
values
        j=j+1;
        m(j,1)=q(i); %matrix containing the correct values in array
    end
end
p=fclose(fID); %closes the file with file ID fID
dlmwrite(strcat(Path,'data.txt'),m); %write to file for prediction
application
```

Príloha III: Hlavní aplikace – UI_prediction.exe – kód

```
function varargout = UI_prediction(varargin)
% UI_PREDICTION MATLAB code for UI_prediction.fig
%      UI_PREDICTION, by itself, creates a new UI_PREDICTION or raises the
existing
%      singleton*.
%
%      H = UI_PREDICTION returns the handle to a new UI_PREDICTION or the
handle to
%      the existing singleton*.
%
%      UI_PREDICTION('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in UI_PREDICTION.M with the given input
arguments.
%
%      UI_PREDICTION('Property','Value',...) creates a new UI_PREDICTION or
raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before UI_prediction_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to UI_prediction_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help UI_prediction

% Last Modified by GUIDE v2.5 28-Apr-2011 15:23:31

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @UI_prediction_OpeningFcn, ...
                   'gui_OutputFcn',  @UI_prediction_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before UI_prediction is made visible.
function UI_prediction_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to UI_prediction (see VARARGIN)

% Choose default command line output for UI_prediction
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes UI_prediction wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = UI_prediction_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes during object creation, after setting all properties.
function uipanel1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to uipanel1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called


function S_Callback(hObject, eventdata, handles)
% hObject    handle to S (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of S as text
%        str2double(get(hObject,'String')) returns contents of S as a
double


% --- Executes during object creation, after setting all properties.
function S_CreateFcn(hObject, eventdata, handles)
% hObject    handle to S (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function L_Callback(hObject, eventdata, handles)
% hObject    handle to L (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of L as text
%        str2double(get(hObject,'String')) returns contents of L as a
double


% --- Executes during object creation, after setting all properties.
function L_CreateFcn(hObject, eventdata, handles)
% hObject    handle to L (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in Show_result.
function Show_result_Callback(hObject, eventdata, handles)
% hObject    handle to Show_result (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

File_path=get(handles.Filepath,'String');
fID=fopen(File_path);                              %open the file
r=fscanf(fID,'%g');
r=r';                                              %transpose the input matrix
val=get(handles.popupmenu1,'Value');
str=get(handles.popupmenu1,'String');
S=str2num(get(handles.S,'String'));
S=(S/8)*1024^2;
L=str2num(get(handles.L,'String'));
average_r=mean2(r);                                %computes the r matrix
avarage value
gamma=average_r/S;
matrix_size=size(r);                               %get the size of the matrix
r
num_r=matrix_size(1,2);                            %get the number of colums
used in matrix r
time=zeros(1,num_r-1);
for i=1:(num_r-1)                                  %cycle - TIME SCALE [s]
    time(i+1)=i;                                   %create a time scale for
the 'x' base in the graph
end
average_time=mean(time);
switch val
    %%ON-OFF METHOD
    case 1
        for i=1:(num_r)                            %cycle - NORMING the r
matrix data
            if r(i)<average_r
                r_norm(i)=(average_r^2/r(i));
            else
                r_norm(i)=r(i);
            end
```

```
        end
        for i=1:(num_r)                         %cycle - COUNT the t1 and
t2 state's
            t_1(i)=(average_r*gamma)/(L*(1-(time(i)/num_r)));
            t_2(i)=S/r_norm(i);
            P(i)=t_1(i)+t_2(i);
        end
        average_P=mean2(P);
        for i=1:(num_r)                         %cycle - UNNORMING the
final equation P
            if P(i)>average_r
                P_u(i)=(P(i)*average_r)/average_P;
            else
                P_u(i)=P(i);                    %packet amount (piece/time)
            end
            y(i)=(P_u(i)*average_r)/1024;
        end
    %%MODIFIED ON-OFF METHOD
    case 2
        for i=1:(num_r)                         %cycle - NORMING the r
matrix data
            if r(i)<average_r
                r_norm(i)=(average_r^2/r(i));
            else
                r_norm(i)=r(i);
            end
        end
        for i=1:(num_r)                         %cycle - COUNT the t1 and
t2 state's
            t_1(i)=(average_r*gamma)/(L*(1-(time(i)/num_r)));
            t_2(i)=S/r_norm(i);
            P(i)=t_1(i)+t_2(i);
        end
        average_P=mean2(P);
        for i=1:(num_r)                         %cycle - UNNORMING the
final equation P
            if P(i)>average_r
                P_u(i)=(P(i)*average_r)/average_P;
            else
                P_u(i)=P(i);                    %packet amount (piece/time)
            end
            Pk(i)=average_r*P_u(i)/1024;        %packet size in kB/time
        end
        average_Pk=mean2(Pk);
        disp=average_Pk-(S/1024);
        for i=1:(num_r)
            if Pk(i)>average_Pk
                y(i)=Pk(i)+disp;
            else
                y(i)=Pk(i);
            end
        end
    %%CORRELATION METHOD
    case 3
        for i=1:(num_r)                         %cycle - DEVIATION
            dev_r(i)=(r(i)-average_r);
            dev_time(i)=(time(i)-average_time);
            m(i)=(dev_r(i)*dev_time(i));
        end
        cov=mean2(m);                           %computes the covariance
```

```
        variance=sqrt(var(r))*sqrt(var(time));  %computes the square roots
of varience multiplication
        rho=cov/variance;                       %computing the population
correlation coeficient
        y=rho*r;
        average_y=mean2(y);
        for i=1:(num_r)
            if y(i)>(average_y*rho)
                y(i)=y(i)+average_y;
            else
                y(i)=y(i);
            end
        end
        y=y*L/1024;
end
%%PLOT
max_y=max(y)+0.1*max(y);
plot(time,y,'Linewidth',2);
axis([time(1) time(num_r) 0 max_y]);
xlabel('time [s]')
ylabel('Packet size [kB]')
if val==1
    str=char(str(1,1));
end
if val==2
    str=char(str(2,1));
end
if val==3
    str=char(str(3,1));
end
title(strcat('Predikovaný síťový provoz pomocí metody: |', eval('str'),'|
[kB/s]'))

% --- Executes on button press in Browse.
function Browse_Callback(hObject, eventdata, handles)
% hObject    handle to Browse (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 [File,Path] = uigetfile('*.txt','Text File (*.txt)','Select the file
containing the values');
 set(handles.Filepath,'String',strcat(Path,File));

function Filepath_Callback(hObject, eventdata, handles)
% hObject    handle to Filepath (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Filepath as text
%        str2double(get(hObject,'String')) returns contents of Filepath as
a double


% --- Executes during object creation, after setting all properties.
function Filepath_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Filepath (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu1
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
popupmenu1

% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function figure1_ResizeFcn(hObject, eventdata, handles)
```