Optimization of update intervals in Dead-Peer-Detection using adaptive Fuzzy Logic

Vesselin Tzvetkov
vesselin.tzvetkov@arcor.net
Arcor AG&Co KG, Alfred-Herrhausen-Allee 1, 65760 Eschborn, Germany

Abstract- **Internet is major communication media and staying "online" becomes a key requirement. Commonly, the broadband access and enterprise networks involve devices with dynamic NAPT(NAT) and statefull firewalls functionality. Keeping open connection behind such an intermediate device is not a trivial task. The intermediate device closes the connection without sending any notify to participants. Until proactively detaching the change of connection status, all send packets get lost. Most applications involve deed-peer-detection mechanisms to verify the host availability, thus if the connection is active. The fast detection of connection lost is vital for the application and therefore short constant dead-peer-detection intervals are used. The update interval is a tradeoff between the network resources and fast detection of disconnection. The interval values are currently constant and set based on the general network experience and application requirements. In this paper is suggested a new framework for dynamic optimization of the update intervals in respect of fast disconnection detection and low network resources. The key idea is to make the update interval proportional to the probability of connection drop. The main issue is that the time point of disconnection can only be narrowed down to certain update interval, the exact time point can not be determined by the hosts. The proposed solution is based on adaptive Fuzzy Logic. The simulation results are presented to verify the performance of the new method. The optimization is relevant for wide spectrum of applications, like VoIP, routing, security protocols etc.**

Index Terms- **deed-peer-detection, adaptive Fuzzy Logic, NAT, NAPT, router, non-linear and non-Gaussian, network events, prediction, One Pass method, RLS.**

I. INTRODUCTION

The internet grows rapidly and its importance becomes significant for many business arias. At the same time, the structure and principles of internet are steadily evaluating. The majority of the internet protocols [IETF] are designed with the strict assumption, that IP layer connectivity is always present. Unfortunately, this condition is currently not applicable. The reasons are that: (1) the majority broadband and enterprise routers implements dynamic Network Address and Port Translation (NAPT) [NAT], (2) the spared of statefull inspection firewalls, (3) the mobile wireless access (for example WiFi host spots) leads to often interruption of the application connection.

NAPT enables private addresses to communicate with the public routable address space [NAT]. The connection thought NAPT device (dynamic NAT) is unidirectional and can be established only form inside to outside, thus usually LAN side to WAN(internet) side. For every connection is created a table entry, which contains information required for the packet forwarding (multiplexing and demultiplexing). The NAPT table entry is removed when: some idle timeout is reached, the public ip of the device has changes or the connection is terminated by the hosts. The entry removal leads to disconnection and it is done without notifying the connection participants. In the case of undesired removal, for example due idle timeout, the end hosts consider the connection as active. Practically the connection "hangs"(zombie connections) - all send packets get lost. The end hosts must detach these zombie connections and reset them. For most of the broadband internet access hosts (quasi ADSL) undesired disconnection happens at least ones a day, since the public ip is rotated ones a day. Zombie connection can be found in other popular scenario is, when the device is moving between different wireless access networks. The device will permanently change its network status between online and offline. The status change requires some action by the application, for example VoIP client. When the ip connection gets down (offline) simultaneously, no notify can be sent to the communication host at the internet side. The internet participant will keep its connection open until some timeout is reached. During this time, all send packets get lost.

In order to reduce the detection time the most application implement dead-peer-detection mechanism. Its main target is the proactive check, if the connection is active. The application sends at regular constant time intervals request packet to the communication host. The receiver of the dead-peer-detection (DPD) message replies to the sender including some payloads of the received packet. In this way the originator (sender) becomes feedback, if the peer is active or not. In general, the result of DPD execution is Boolean: active or not active. The exact time of becoming unreachable (offline) is unknown. It is between to last successful DPD execution and update failure. The result says, if the peer is currently reachable or not. The mechanism is widely used for example in: VoIP protocols SIP, H323, routing protocols like BGP, ISIS, OSPF, security protocols as IPSec. The notation is different and can be update, hello messages etc. In this paper we use the common name dead-peer-detection, since it is more intuitive.

The "hanging" connections lead to packet loses, so their existing time must be minimal. For fast detection of the connection failure commonly short DPD intervals are used. The fast detection is a tradeoff the network resources. More updates means more traffic and processing power.

The dead-peer-detection(DPD) mechanism implemented in majority of the applications suffers of inflexibility. The update time intervals are constant. Using expert knowledge of networks and knowing the application requirements, some time value is set. For example in VoIP SIP application traditionally short interval of 10-15 sec are used. The BGP routing protocols use 90-180 seconds. The set value does not change during the session.

In this paper we are using non-linear update intervals for minimizing the network resources and keeping the disconnection detection time low. The main goal is to make the size of the update interval dependent of the probability for connection drop. The update intervals close to the predicted disconnection time point must be small and update intervals with low probability must be large. The proposed method is based on adaptive Fuzzy Logic controller. Fuzzy controller is very suitable, because can handle expert experience and combine it with adaptive algorithms. The Fuzzy rule base is created with One Pass(OP) training method and expert knowledge. The membership functions are optimized for fast convergence with Recursive Least Square method (RLS) [FUT].

The challenge in optimizing the update intervals is that there are no exact measurable values – the time point of disconnection. The disconnection time point can only be narrowed down to the update interval. This is the time interval of the dead-peer-detection. Since the result of the DPD execution is Boolean active/not active, it is not possible to determine the exact disconnection time. If the update interval is 300 seconds, we can find the interval and determine the disconnection somewhere within this 300 sec.

In the classical filter theories based on Bayesian rule there are crisp(exact) measured values involved in the calculation. The measurement values consist of noise and equalized signal. The general task is to filter the noise from the signal. In this paper we overcome this issue by creating model which contains crisp values.

In the following chapter II the terminology and objectives are defined. Method overview is made in the next chapter III. The update distribution problem and its solution are presented in chapters IV, V and VI. The Fuzzy controller are briefly described in chapter VII. The Fuzzy rules creation through One Pass training and RLS optimization is described in IX and X. The last chapter concludes the results.

II. Model Therminology and Objectives

The implementations are slightly different, so we create an abstract model summarizing and representing all deployments. Without changing the nature of the dead-peer-detection we made the following abstraction and assumptions: The host sending the packets receives feedback with the execution result, thus exit code. The result is Boolean. The status *"true"* means the update was successful and *"false"* means it has failed. Practically if the reply is received in the dead-peer-detection the result is *"true"* - the connection is active. If no reply is received within certain time interval the exit status is considered as *"false"* – disconnection. We are concentrated in prediction of the *false* events. Let us further denote "update procedure" meaning execution of dead-peer-detection.

The execution time of the update procedure is denoted as *Update Time Point* (s. **Figure 2**). The update procedure practical can not be executes in zero time, so we consider the time point, when the result is received. The time point, where the disconnection occurs, is *Event Time Point* (ETP). *Disconnection Interval* (DI) is the time interval for disconnection detection. This is the interval between the event time point and the following first update time point. *Maximum Disconnection Interval* is time tolerable by the application without active connection (zombie connection). It is usually couple of seconds for real time applications, like VoIP and minute for not time critical applications as Email. The *maximum Disconnection Interval* value must not be exceeded. The interval between two following event time points is the *Event Interval* (EI). The interval between two following update time points is *Update Interval.* All definitions are shown in **Figure 2**.

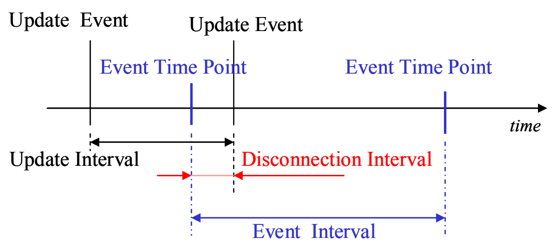

Figure 2 Terminology

The disconnection interval is less or equal to the smallest update interval, which surrounds the event time point (s. **Figure 2**). Less update interval, the less disconnection interval becomes.

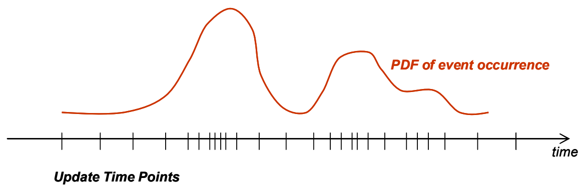

Figure 1 Update time points and event PDF

To reduce the network resources the update interval length must be inversely proportional to the probability of *false* event occurrence (PDF), as shown at **Figure 1**. The regions with high probability density must have small update intervals. The large update interval means low event probability.

III. Method Overview

The method consists of prediction and update phase. In the prediction phase the upcoming event interval (EI) is predicted. For this purpose, adaptive Fuzzy logic controller is involved. The controller has input parameter the posteriori estimated event intervals and the absolute time. The output is the predicted event interval. In the second phase using the predicted event interval the update time points are calculated. The update time points are calculated using transformation function. The transformation function converts constant linear update intervals to intervals proportional to the priori probability of event time point. The transformation function's input parameters are: the predicted event interval and the maximal disconnection interval. The function plays a key role in the proposed method and described in chapter VI.

The update procedure is executed sequentially at the calculated time points. If the execution result of the update procedure is *false*, then the update procedure is terminated. The estimate event time point is in the middle of the last update interval. The update sequence is started form the beginning using the obtained estimated event time point. The steps are summarized in following points:

1) The Fuzzy controller initialization. The initialization is done thorough One Pass (OP) training methods and expert knowledge. Chapters IX and X describe the procedure.
2) Using the last n posteriori event intervals, the new predicted event interval is calculated. This procedure is described in chapter VIII.
3) Using the transformation function the update time intervals are calculated. See chapters V and VI.
4) The updated procedure is executed until the result of the execution is *false*.
5) The update interval is estimated in the middle of the last update interval. The execution is next returning back to 2).

IV. PROBLEM OVERVIEW AND INTERVAL LENGHT

Implementing classical prediction theory in update procedure arises the problem, that there is no crisp measured values. The event time points can be narrowed down to a certain interval, which can not be handled directly in classical theory. The measurement represents in our case the event time point (ETP). To solve this problem we assume that the measured crisp value is in the middle of the update interval, in which disconnection occurs. The maximal absolute estimation error in this case is the half of the update interval. A further important assumption is that the probability for ETP is Gaussian distributed with standard deviation proportional to the length of the update interval. At **Figure 4** the PDF (probability density function) in update interval is shown.

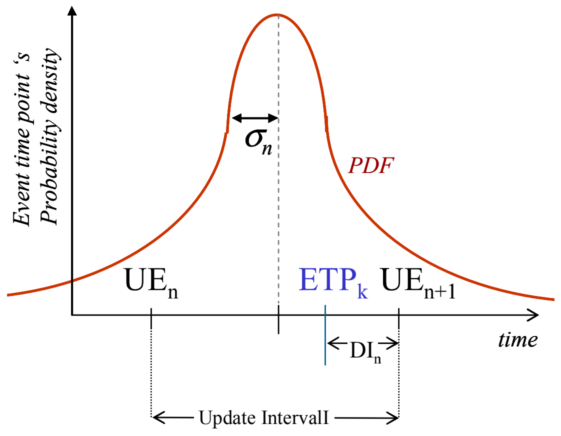

Figure 4 Gaussian distribution

It is important to stress that the Normal (Gaussian) distribution is infinite, so it is not possible to be limited in the update interval. The controversy is that practically outside the update interval there is zero probability for the event occurrence. It's sure that the event has happened in the update interval. Although the Normal distribution cannot satisfy the mentioned condition. It gives a smooth reduction of the probability and many natural phenomena are describe by it.

The length of the update interval must be inversely proportional to the probability density function (PDF) function, as defined in II. The PDF of the Normal distribution is the well known bell curve. At **Figure 3** is shown the abstract relation between the probability of ETP and the update interval length. The inversely proportional function of the update interval length is defined general by:

$$L_i = a - b.N(x_i,\mu,\sigma) \qquad (1)$$

, where *Li* is the length of the i-th interval, *a* and *b* are constant. $N(x_i,\mu,\sigma)$ denotes the normal distribution function at x_i, where μ,σ are the mean and standard deviation. Normal distribution is defined by

$$N(x_i,\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(x_i-\mu)^2}{2\sigma^2}\right)$$

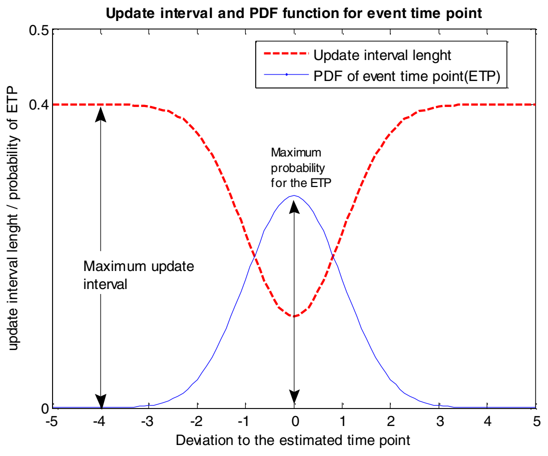

Figure 3 The ETP probability and interval length

V. UPDATE INTERVALS AND TRASFORMATION FUNCTION

In order to generate update interval according (1) we create transformation function T(). The function T() transforms constant update intervals to intervals proportional to the PDF of event time point. The transformation function depends of the following parameter: maximal disconnection interval defined by the application, predicted event interval value and normal distribution coefficients. These parameters define fully the transformation function.

Let us first derivate the equation for the transformation function T(). The difference between two transferred time points is the update interval: $T(x_1) - T(x_2) = L_i$, where $x_{1,2}$ are input time points, L_i is the update interval length and $T()$ the transformation function. The interval L fulfills the defined in equation (1) for inverse proportion to ETP's PDF, so it can be found that:

$$T(x_1) - T(x_2) = L_i <=> \frac{\partial T(x)}{\partial x} = L_i$$

$$\frac{\partial T(x)}{\partial x} = L_i = a - b.N(x_i,\mu,\sigma)_i$$

$$T(x) = \int_n^m a - b.N(x_i,\mu,\sigma)dx$$

$$T(x) = \int_n^m a - b.\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)dx$$

$$T(x) = ax - \frac{b\sqrt{2}}{\sigma\sqrt{\pi}}erf\left(\frac{x}{\sigma} - \frac{x}{\mu}\right) + C$$, where

$$erf(u) = \frac{2}{\sqrt{\pi}}\int e^{-u^2}du$$

The erf() denotes error function and C is constant due the integration. The coefficients in the equation can be rewritten to simpler form:

$$T(x) = ax - u.erf\left(\frac{x}{k}\right) + C, \qquad (2)$$

where a,u,k and C are the coefficients.

VI. Approximation of the Transformation Function

The function (2) is the analytically derivate and optimal in all aspects. Unfortunately the transformation function (2) can not be used in the suggested form. Implementable transformation function should be invertible, thus the inverse transformation $T^{-1}(x)$ is required. The transform function (2) consist of polynomial and trigonometric terms and it's inverse can't be found analytically. To overcome this shortcoming we approximate the transformation function to set of invertible functions.

Analyzing the function (2) and curve at **Figure 5** it's easy to notice, that for small x values the domination term is $u.erf\left(\frac{x}{k}\right)$ and for big x values the term ax dominates the result. The transformation function can be spitted to:

$$T(x) = \begin{cases} u.erf(x/k) + C_1, x \in (-l, +l) \\ ax + C_2, \forall x \in \Re \notin (-l, l) \end{cases}$$

Then the inverse function is:

$$T^{-1}(x) = \begin{cases} k.erf^{-1}((x - C_1)/u), x \in (-l^{tr}, +l^{tr}) \\ (x - C_2)/a, \forall x \in \Re \notin (-l^{tr}, l^{tr}) \end{cases}$$

At the next step the coefficient of new T() equations are derivate. The frame requirements include objective and subjective arguments:

1) The fist condition is that the transformation function and its inverse must be continuous (not interrupted). Otherwise for some values is not possible to find the inverse and visa versa. This gives relation between the two spitted functions at the crossover point - *l*:

$$u.erf((l/k)) + C_1 = a.l + C_2$$

The function must have the same value at this point.

2) The cross point *l* is defined using subjective knowledge. The cross over point defines the region (*-l,l*), where the intervals becomes inverse proportional to the probability. Outside this interval the updates are constant, since the transformation function there is linear. The predicted event interval is a reasonable to be equal to *l*, thus $l = P_k^+$, where P_k^+ is the predicted event interval. The k index denote the prediction cycle. The plus sign stress that it's the predicted values – priori estimation.

3) The linear update interval is the maximal disconnection interval defined by the application. It is the input of the transformation function. This value must not be exceeded. The value depends of the application requirements as described in chapter II.

3) The output of the transformation function (the non linear update interval) must also not exceed maximal disconnection interval. Considering condition 2) it follows, that the derivate of T(x) must be bigger then one for $x_k \in (-P_k^+, P_k^+)$:

$$\frac{\partial}{\partial x}u.erf(x_k/k) \geq 1, \quad x_k \in (-P_k^+, P_k^+)$$

4) The erf() function is an integral of the famous bell curve(Normal distribution). Observing the curve can be noticed, that the erf() reaches it's maximum at the infinity. On the other hand the function reaches 95% of it's maximum quite rapidly at ca. x=1,38k, see equation (2). The last 5% of the increase can be approximated to line, which is almost horizontal. The linear part of the function is not interesting for our transformation. With this empirical and subjective consideration the condition is defined: The erf function riches 95% of its maximum at the cross point between the components functions, thus:

$$P_k^+ + C_2 = 0.95u$$

5) The transformation function has its minimum at the predicted event time point, since there the probability is highest. The event time point is set as point zero and all points in relation to it. The transformation function is odd function returning positive values. This condition facilitates the practical implementation and do not restrict properties.

These five conditions deliver the solution for the coefficients of the transformation function. Without going deeper in mathematical solution steps we calculate the transformation function:

$$T(x) = \begin{cases} u.erf(x/k), x \in [-P_k^+, +P_k^+] \\ x + C_2, \forall x \in \Re \notin (-P_k^+, P_k^+) \end{cases}$$

$$T^{-1}(x) = \begin{cases} k.erf^{-1}(x/u), x \in (-P_k^+, +P_k^+) \\ x - C_2, \forall x \in \Re \notin x \in (-P_k^+, P_k^+) \end{cases}$$

with

$$k = \frac{P_k^+}{erf^{-1}(0.95)}, \quad u = \frac{k\sqrt{\pi}}{2}e^{2erf^{-1}(0.95)},$$

$$C_2 = 0.95.\frac{\sqrt{\pi}}{2}e^{erf^{-1}(0.95)} - P_k^+$$

Approximated transformation function with update linear interval of 1 and estimation of 2 is shown at **Figure 5**. The zero point at the abscise is predicted event time point(ETP). The time values at abscise are relative to ETP, see condition 5). The function obeys the required conditions: it become linear at infinite and for values near to the ETP is not linear.

The intervals become smaller nearing the zero point, thus the probability becomes higher – the intervals smaller.

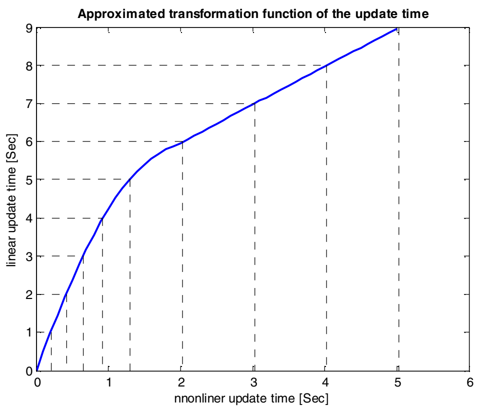

Figure 5 Transformation function

VII. Fuzzy logic Overview

By exploring phenomena generally there are two types of descriptions - objective and subjective. The objective knowledge creates system model involving univalent mathematical equations based on known physical properties. The second possibility for system description is using subjective knowledge, which represents expert experience. The discipline for defining, creating operations and processing subjective knowledge is called Fuzzy Logic. Fuzzy Logic is mainly used for solving very complex problems, where an exact solution in objective mathematical sense can not be achieved in the required technical frame. Here we are not giving details on fuzzy logic, more detailed can be found among other in [FUM][FUN].

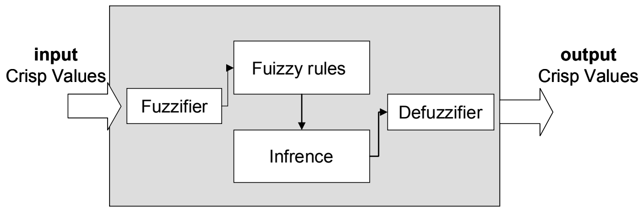

Figure 6 Fuzzy system

Fuzzy system consists of the following blocks: fuzzifier, fuzzy rules, inference and defuzzifier. The crisp input values are transformed by the fuzzifier in fuzzy set. Using the fuzzy rules and inference the result fuzzy set is calculated. The defuzzifier converts the fuzzy set back to crisp value.

The abstract crisp output of one fuzzy controller is:

$$f = D(O) = D\left(\bigwedge_{l=1}^{M} \left[\mu_o^l(y) * \operatorname*{T}_{k=1}^{n} \mu_Q(x_{k,\sup}^l) \right] \right)$$

, where μ_o and μ_Q is the membership functions of consequent and antecedent rules. The Operation * is a single T-norm operation and T denotes sequence of T-norm operation. The system has n input variables and y is the centre of the consequent membership function. The Λ denotes the T-Conorm operation and M is the number of rules. The D() denotes the defuzzfication function.

If centroid method is used for defuzzification and singletone output membership functions, then the crisp output is denoted as:

$$f = \frac{\sum_{l=1}^{M} y^l \left[\operatorname*{T}_{k=1}^{n} \mu_Q(x_k^l) \right]}{\sum_{l=1}^{M} \operatorname*{T}_{k=1}^{n} \mu_Q(x_k^l)} = \sum_{l=1}^{M} y^l p_{fb}^l(x) \qquad (3)$$

$$p_{fb}^l(x) = \frac{\operatorname*{T}_{k=1}^{n} \mu_Q(x_k^l)}{\sum_{l=1}^{M} \operatorname*{T}_{k=1}^{n} \mu_Q(x_k^l)}$$

The p variable is Fuzzy Basis Function as defined in [FFB] and it will play important role in the creation of the fuzzy controller.

VIII. Prediction with Fuzzy controller

The fuzzy controller uses the last posteriori event intervals to predict the upcoming event interval. The input values are used in antecedent part of the fuzzy rules. Using training method the exact rules are created, see IX. Before starting the fuzzy controller the training phase must be finished. An additional antecedent element is the absolute time. The absolute time helps to make estimation if it is high probability time - rush hour (see IX). The Fuzzy controller is working as black box for every input values in gives predicted interval.

IX. Creation of the rule base

The rule base plays decisive role in the performance of the model. It is crated using two methods: first by using the expert knowledge and second by using training methods.

A. Expert knowledge

The experience by observing the network brings the rudimental knowledge involved in rules creation. In most of the cases DPD is used with normal repeating cycle. For working employee in business days there are general two working maximums at the morning and after lunch. The Monday's maximums are higher then at the Friday. At the weekend there is a reduced activity. Typical diagrams for day and week activity is at Figure 7. (Internet exchange point PARIX) This knowledge builds the first rules in the rules base.

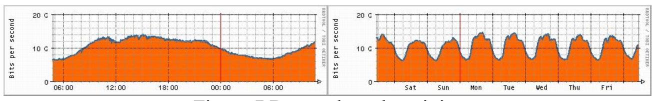

Figure 7 Day and week activity

Profile with rules build the rudimentary sets depending of the customers behavior. The base rules can be for example:

IF 7:oo h < time AND time < 20:oo h THEN Update Interval IS small
IF Monday < day AND day < Friday THEN Update Interval IS small

Multiple profiles can be created as for example: traveler, office employee, holiday etc. The user can switch manually between the profiles for increasing the performance.

B. *Training methods*

In many cases there are no sufficient expert information to create exact membership functions and rules, thus no acceptable results can be achieved. If there are already observed actions-reactions of the system, then they can be used to create fuzzy rules and adjust the membership functions. Involving known system input and output values to adjust the controller parameters is called *training*. We are using One Pass method with Recursive Last Squares (RLS) optimization.

One Pass method

One Pass(OP) is simple and fast method for creation of fuzzy rules using known input-ouput pairs [FWM]. The generated rules are used further in the controller. The given values are $(x_1^{(1)}, x_2^{(1)}, ...x_n^{(1)}, y^{(1)}), (x_1^{(2)}, x_2^{(2)}, ...x_n^{(2)}, y^{(2)})$, where x is the crisp input values and y is the corresponding output value. The index denotes the pair number. There are n input variables and single output variable.

1) The intervals, where the input/outpu variables are expected to lie are: $[x_1^-, x_1^+]; [x_2^-, x_2^+] ... [x_n^-, x_n^+]; [y^-, y^+]$. The intervals can be adjusted dynamical, but at first stage for simplicity they are fixed. Each interval is divided into N regions, where the number of regions can be different for every variable. The number depends of the required precision for the variable. More important variable should have more regions. To every region a membership function is assigned. The membership function is not zero within the region and zero outside it.

2) Every pear of input-output values is evaluated to fuzzy rule. Each input-output set is assigned to the region with the highest membership function. Proceeding in this way for every training set(pair) one rule is created. If there are n pairs, then n rules are created.

3) For every rule degree of strength is calculated in order to resolve conflicting rules. The degree S of l-th rule is calculated as follows:

$$S_l = \mu_Y(y^l) \prod_{i=1}^{n} \mu_{Xi}(x_i^l)$$

, where $\mu_Y, \mu_{X1}, ...\mu_{Xn}$ are the membership functions for the rule. Conflicting rules have the same antecedent memberships, but different consequent membership. For specific combination of antecedent members only one rule must be selected - for one combination for input values is possible only one output. If there are conflicting rules the rule with the highest degree is kept and the other one is discarded.

4) After evaluating all set as described in 2) and 3) the rules base is ready for use. The new crisp input values can be evaluated in the rule base. Using defuzzification the crisp output value is calculated.

Recursive Last Squares Method

Major drawbacks of the One Pass(OP) method are the form and the centre of the membership functions. The values are predefined mostly without sufficient knowledge for the system. The result performance of OP is not optimal. The main idea of the RLS method is minimizing the sum of square error between the crisp fuzzy output and the real training data.

Let us consider, that the rules are already developed by some method as OP. The Fuzzy output f() according the equation (3) is:

$$f = \sum_{l=1}^{M} y^l p_{fb}^l(x) = Y^T P^{fb}(x)$$

, where Y and P are the matrixes of column values. The RLS algorithm can be applied to this equation, since *f()* is linear function in respect to Y. For every training data set k then input-output pairs must be evaluated [FAF]:

$$P_{k+1} = P^{fb}{}_{k+1,*}, \qquad G_n = \frac{L_k P_{k+1}}{\lambda + P_{k+1}^T L_k P_{k+1}}$$

$$e_{k+1} = d_{k+1} - P_{k+1}^T Y_k, \qquad Y_{k+1} = Y_k + G_k e_{k+1}$$

$$L_{k+1} = \lambda^{-1} L_k - \lambda^{-1} G_{k+1} P_{k+1}^T L_k$$

The $P^{fb}{}_{k+1,*}$ denotes the fuzzy basis function at k-th training pair. It is column with M values, one for each input. The G_k is the gain vector at step k. L_k denotes the inverse correlation matrix at step k. L is initialized MxM identity matrix. The value e_k is the priory error. Y is the matrix with the centers of the consequent membership function. It is initialized with M zero values. More information on the mathematical background can be found in [FUM].

X. Training data collection

The training data is collected using constant update intervals. This intervals must be enough small, so the DI(disconnection interval) is tolerable for the application. The estimate ETP (estimate time point) is the middle of the update interval, when the event has occured. Reasonable update interval size in our studies was half of the maximum disconnection interval. It is very important, that the training date includes all significant states and in this way to representative rule to be created.

XI. Conclusion

The simulation results in Appendix A proof that the proposed algorithm has strong advantages over the constant update. Involving past disconnection events for prediction of the coming events is strong base for good results. A key point for achieving good results is enough big training period. The training must include in best case all possible states and transitions of the system. Another very important point is the right number of past values as input. Chousing more values increase the number of fuzzy rules needed for adequate results. The number of rules plays must be sufficient to represent all combination input-output variables.

Appendix A – Simulation Results

An implementation and practical simulation of the update procedure was done in order to proof the qualities of the developed method. In this section the results briefly overviewed. The simulation was made with Matlab 7.0 on PC with 1.1 GHz and 512 RAM. The simulation was offline using intervals values, so the absolute time was not relevant. In this way the simulation code is simplified significantly. Unfortunately it is not possible to use expert knowledge to create basic rules.

For the simulation we used 4500 event intervals and training in the first 1000 samples. The membership functions was 2 to 3 time the maximum disconnection interval, which is 10s.

To determine quality in exact manner, we compare the method to constant update intervals. Constant update is the most popular method, where the update interval are monotone constant length. In order to achieve fair comparison the both method fuzzy and constant run with the same resources. This is fair way for objective results. The same resources means the same number of updates during the simulation time. Using the same number of updates in the same time the both methods deliver different disconnection times. The minimal, maximal and mean values of the disconnection interval are compared.

A.1 Non liner function

Non liner signal with white noise is the big challenge for the algorithm. The N=4500 random values $\underline{r}$ are generated with the recursive equation used also in [PNO][PPA]

$$r_k = \frac{r_k}{2} + \frac{25 r_{k-1}}{1 + r_{k-1}^2} + 8\cos(1.2k) + v_k^r + 100$$

$$v_k^r = N(0, \vartheta)$$

The standard deviation of the noise component is 10. The fuzzy rule base includes 100 rules The 3 last values are used for the prediction and the membership function is 3 times the maximum disconnection interval of 10s.

The generated Event Intervals (EI) and the estimated event interval are show at the **Figure 8**. The values are not linear and subjective there are not dependencies.

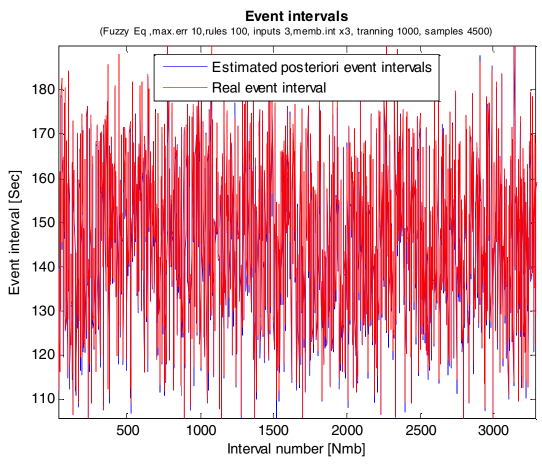

Figure 8 Event intervals case A.1

The histogram of the update intervals is presented at **Figure 9**. Using the same network resurces (updates) the suggested method delivers is in 96,34% smaller disconnection interval to constant update interval. The disconnection interval(DI) is maximum equal or less to the update interval Die DI is not measurable in the praxis and we compare the update intervals.

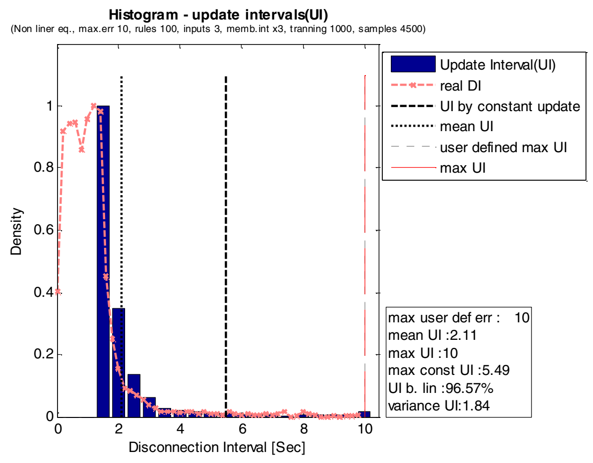

Figure 9 Histogram case A.1

The mean update interval of new method is 2.11s and in the constant updates - 5.49s. There is over performance of 260%. There is clear advantage of the developed method to the classical constant method. The update interval for every same are shown at **Figure 10**.

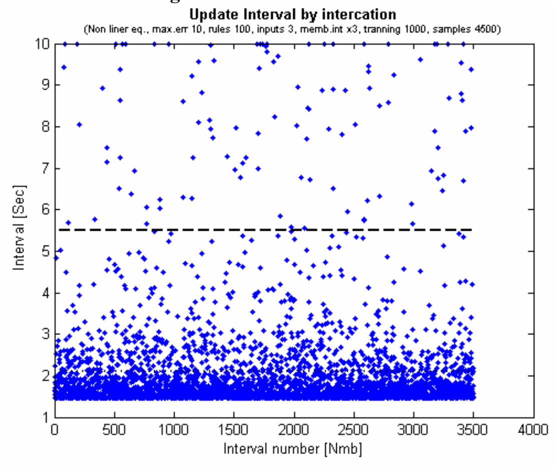

Figure 10 Update interval values, case A.1

A.2 Real data

In this case we tested with real network data. The data is represent the dialin intervals in one administrative domain. In order to represent the worst case there were missing data regions and sporadic abnormal activities.

Even in this difficult case the suggested algorithm is in 89,09% of all estimations better then the constant updates(Figure 11). The mean disconnection intervals by here presented method (4.14s) and the constant update (5.55s) shows an performance of 134%.

A.3 Sinus based signal

In contrast to previous case a simulation with strong dependence. The disconnection intervals are combination of sinus signal and white noise:

$$r_k = \vartheta.\sin(4\pi k / \max(k)) + 0.7.\vartheta.\cos(1.5\pi k / \max(k))$$
$$+ sc(k / \max(k)) + 200 + v_k$$
$$v_k^r = N(0, \vartheta); \vartheta = 10$$

The function max() denotes the maximum sample elements, thus 4500. The data is presented at the **Figure 12**. This is best case scenario for the adaptive fuzzy logic algorithm.

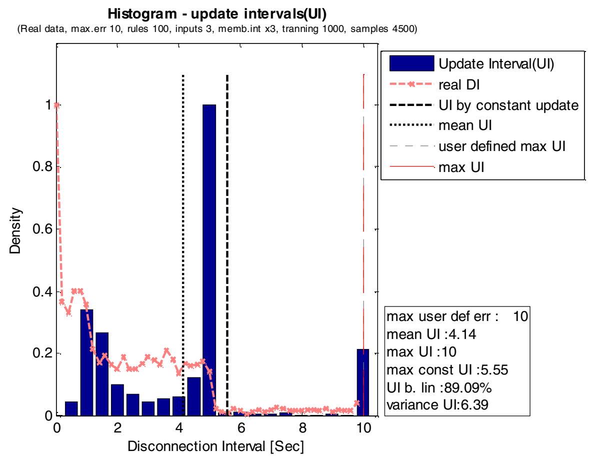

Figure 11 Simulation with real data

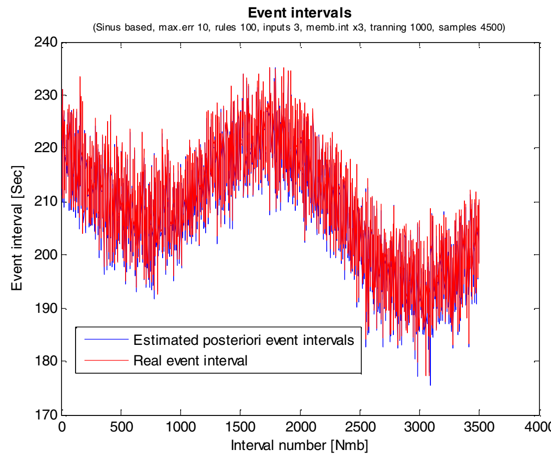

Figure 12 Event time intervals case A.3

The advantages of the suggested method are clearly presented at the **Figure 13**. The suggested algorithm achieves mean update interval of 1.6s and the classical approach 5.91s. There is clear over performance of 369%. In 99.49% of all update the performance was better then the classical constant update. This is the best case for the new method.

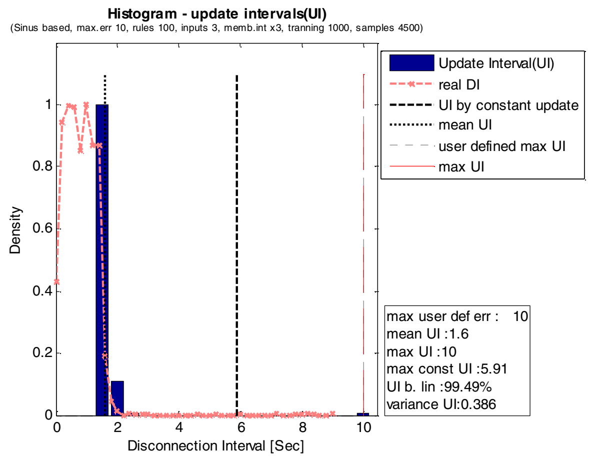

Figure 13 Results of sinus base signal, case A.3

In is important to underline, that the good performance is a result of optimal parameters, like: number of rules (100), number input valuable (3) and membership function size (3x max DI). The training interval is very critical for conversing best results. If the parameters are chosen wrong than the performance will significantly decrease.

The performance decrement is shown by repeating the last simulation using NOT optimal parameters: 50 rules, membership function 2x max DI. (The same sinus based signal – "best case"). The update intervals are shown at **Figure 14**.

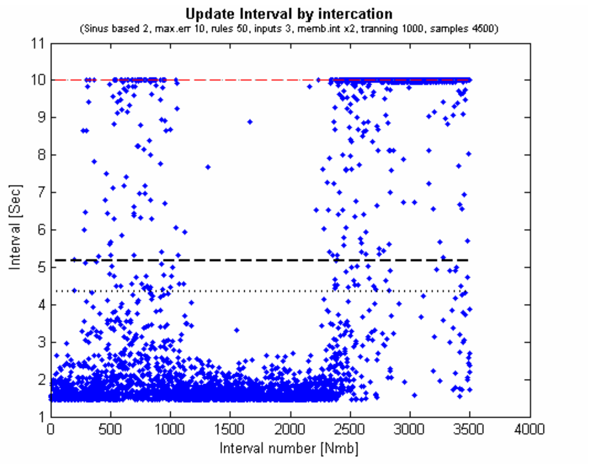

Figure 14 Sinus based signal with not optimal parameters

It is clear to see that in samples 300 to 700 and 2500 to 3500 there is big gab in the prediction algorithm. The values increase to the maximum Disconnection interval and the performance becomes poor. The fuzzy controller does not include sufficient rules to cover all input values combination and the performance decrease. How the parameters must be optimal chosen is part of a future work.

References

[FUT] "A First Course in Fuzzy Logic", H. Nguyen, E. Walker, Chapman & Hall/CRC; 3rd , 2005

[FUM] "Fuzzy Logic Systems for Engineering: a Tutorial" , IEEE Proc., Vol. 83, pp. 345-377, March 1995.

[FUN] "Designing fuzzy logic systems," G. Mouzouris, co-author), IEEE Trans. on Circuits and Systems, Part II, vol. 44, pp. 885-895, Nov. 1997.

[FTN] "Triangular Norms" Erich Peter Klement, Radko Mesiar and Endre Pap; Springer; July 1, 2000

[FFB] "Fuzzy Basis Functions, Universal Approximation, ...", L. Wang, co-author, IEEE Trans. on Neural Networks, vol. 3, pp. 807-814, Sept. 1992.

[FWM] "Generating fuzzy rules by learning from examples", L. X. Wang and J. Mendel, IEEE Transactions on Systems, Man, and Cybernetics, vol. 22, July 1992.

[FAF] "Adaptive Filter Theory", Simon Haykin, Prentice Hall, 2002,

[PNO] "Novel approach to nonlinear and non-Gaussian Bayesian state estimation", N. Gordon, et.al, Proc. Inst. Elect. Eng., vol. 140, 1993.

[NAT] RFC 3022 "Traditional NAT", January 2001

[IETF] www.ietf.org

[PPA] "A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking", M. S.Arulampalam et al, IEEE, VOL. 50, NO. 2, 2002