ACCPndn: Adaptive Congestion Control Protocol in Named Data Networking by learning capacities using optimized Time-Lagged Feedforward Neural Network

Amin Karami*

Computer Architecture Department (DAC), Universitat Politècnica de Catalunya (UPC), Campus Nord, Jordi Girona 1-3, 08034 Barcelona, Spain

Abstract

Named Data Networking (NDN) is a promising network architecture being considered as a possible replacement for the current IP-based Internet infrastructure. However, NDN is subject to congestion when the number of data packets that reach one or various routers in a certain period of time is so high than its queue gets overflowed. To address this problem many congestion control protocols have been proposed in literature which, however, they are highly sensitive to their control parameters as well as unable to predict congestion traffic well enough in advance. This paper develops an Adaptive Congestion Control Protocol in NDN (ACCPndn) by learning capacities in two phases to control congestion traffics before they start impacting the network performance. In the first phase -adaptive training- we propose a Time-Lagged Feedforward Network (TLFN) optimized by hybridization of particle swarm optimization and genetic algorithm to predict the source of congestion together with the amount of congestion. In the second phase -fuzzy avoidance- we employ a non-linear fuzzy logic-based control system to make a proactive decision based on the outcomes of first phase in each router per interface to control and/or prevent packet drop well enough in advance. Extensive simulations and results show that ACCPndn sufficiently satisfies the applied performance metrics and outperforms two previous proposals such as NACK and HoBHIS in terms of the minimal packet drop and high-utilization (retrying alternative paths) in bottleneck links to mitigate congestion traffics.

Keywords: Named Data Networking, Congestion Control, Time-Lagged Feedforward Network, Particle Swarm Optimization, Genetic Algorithm, Fuzzy Set

1. Introduction

Information-Centric Networking (ICN) [1, 2, 3] has been proposed as a solution for a viable and vital replacement for the current IP-based Internet due to the fundamental limitations of the Internet in supporting today's content-oriented services [4, 5, 6, 7]. Named Data Networking (NDN) [8] is a prominent example and ongoing research effort of ICN design. The main goal of NDN is to support the dissemination of named content rather than the current host-centric (end-to-end) delivery of content to a named host. In NDN, a consumer asks for a *Content* (*Data*) by sending an *Interest* request using name prefix (content identifier) instead of today's IP prefix (content location). An Interest packet is routed towards the location of the content's origin where it has been published. Any router (intermediate node) on the way checks its cache for matching copies of the requested content. The requested content is returned by any node that holds a copy of the content in its cache. On the way back, all the intermediate nodes store a copy of the content in their caches to satisfy subsequent users interested in that content (i.e., in-network caching). Congestion takes place in NDN routers when the number of arrival data packets is higher than the queue's capacity which causes an overflow in the routers' buffer [9, 10, 11]. When this happens a high data packet loss and increase in the end-to-end delay occur affecting negatively on the performance, stability and robustness of the network [12, 13]. This leads to under-utilization of the available resources and degradation of throughput and quality of service [14, 15].

This difficulty has recently motivated researchers to explore ways of congestion control in NDN. Some of the relevant contributions are [9, 10, 16, 17, 18, 19, 20]. The main weak points of the proposed methods are: a too high sensitivity to their control parameters as well as the

*Corresponding author, Telephone: 0034-934011638
Email address: amin@ac.upc.edu (Amin Karami)

inability to predict congestion traffic well enough in advance. This will often bring unfair bandwidth sharing, network instability, packet loss, additional delay and so on [21, 22]. The first goal of any method against congestion can be the early detection (ideally long before the problematic traffic builds up) of its existence. If the congestion problem can be recognized in advance, changing network parameters can possibly prevent such costly network breakdowns. Network traffic prediction plays an important role in guaranteeing quality of service in computer networks [23]. The prediction of network traffic parameters is feasible due to a strong correlation between chronologically ordered values [21]. Their predictability is mainly determined by their statistical characteristics including self-similarity, multiscalarity, long-range dependence (LRD) and a highly non-linear nature [24]. Prediction algorithms can be embedded into network communications to improve the global performance of the network by anomaly detection, proactive congestion detection (or avoidance), and allow a better quality of service by a balanced utilization of the resources [23, 25, 26]. Contributions from the areas of operational research, statistics and computer science have lead to forecasting methods. In particular, the field of Time Series Forecasting (TSF) deals with the prediction of a chronologically ordered values [27, 28]. The goal of TSF is to model a complex system as a black-box in order to predict the systems behavior based on the historical data [21, 29].

In this paper, we develop ACCPndn (Adaptive Congestion Control Protocol in Named Data Networking) which is a new congestion control protocol with learning capacities. The ACCPndn focuses on two phases for congestion control before building up in NDN. **The first phase -adaptive training-** learns from the past breakdowns to how to detect the problem beforehand. This phase allows to identify the source of the congestion together with the amount of congestion. This phase uses Time-Lagged Feedforward Neural Network (TLFN) approach. The TLFN adopts a multilayer perceptron ANN (Artificial Neural Network) and time series forecasting (TSF) [30, 31]. The major advantages of neural networks in time series forecasting are their flexible nonlinear modeling capability that there is no need to specify a particular model form and high data error tolerance [32, 33]. A Back-Propagation is a most popular NN algorithm (BPNN) to determine and adjust network parameters, weights and biases. Despite the advantages of BPNN, it has some drawbacks that the most important one being their poor trainability. It might fall to local minima and cause overfitting and failure of the network training [34, 35]. There is a recent trend to train BPNN with bio-inspired optimization algorithms for different applications [36, 37, 38]. In this paper, in order to improve the performance of BPNN, a new combined algorithm namely Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) is presented to optimize the weights and the biases of network, and to prevent trapping in local minima. The results show that our proposed combination of PSO/GA with TLFN (TLFN+PSO-GA) performs better than the GA/PSO, PSO and GA in terms of the applied performance criteria.

When the source(s) and the amount of congestion are identified, they are sent to the second phase for congestion control before building up. **The second phase -fuzzy avoidance-** performs a fuzzy decision-making approach to proactively respond to network congestion rather than simply wait for a congested queue to overflow and the tail drop all subsequently arriving data packets. The application of fuzzy decision-making techniques to the problem of congestion control is suitable due to the difficulties in obtaining a precise mathematical (or a formal analytical) model, while some intuitive understanding of congestion control is available [39, 40]. Its use allows to regulate effectively the incoming Interest packets in each routers' interface.

The main objective of the proposed protocol is to enable a stable equilibrium and satisfy some basic requirements which are characterized by the utilization of multiple paths and few packet drops. The second objective is to present a scalable and fast convergence properties with respect to varying bandwidths, traffic patterns, and number of users at different times utilizing the network. The evaluation through simulations shows that ACCPndn can quickly and effectively respond against congestion problems in a timely manner and performs successfully even in the large networks as compared to two recent congestion control mechanisms namely NACK [20] and HoBHIS [9] in terms of the applied performance metrics.

The rest of the paper is organized as follows. Section 2 describes the background materials of NDN. Section 3 gives an overview of the time series forecasting approach. Sections 4 and 5 describe the PSO and the GA, respectively. In Section 6, multilayer perceptron neural network is described in detail. Section 7 provides a overview of fuzzy set theory. The proposed method (ACCPndn) is completely presented in Section 8. Section 9 presents the evaluation setup. The results of simulation is also presented in Section 10. Finally, Section 12 draws conclusions.

2. Named Data Networking (NDN)

NDN communication protocol is receiver-driven and data-centric. All communication in NDN is performed using two distinct types of packets: `Interest` and `Data`. Both types of packets carry a `name`, which uniquely identifies a piece of data that can be carried in one data packet [41, 42]. Data names in NDN are hierarchically structured, e.g., eight fragment of a youtube video would be named `/youtube/videos/A45tR7Kg5/8`. In addition to the data name, each Interest packet also carries a random `nonce` generated by the consumer. A router checks both the name and nonce of each received Interest packet. If a newly arrived Interest packet carrying the same name as a previously received Interest packet from a different consumer, or a previously forwarded Interest looped back, the router drops the Interest packet. Therefore Interest packets cannot loop. Each NDN router maintains three major data structures [43, 44]:

1. The Pending Interest Table (PIT) holds all not yet satisfied Interest packets that were sent upstream towards potential data sources. Each PIT entry holds one or multiple incoming physical interfaces and their corresponding Interest packets.
2. The Forwarding Information Base (FIB) forwards Interest packets to one or multiple physical network interfaces based on the forwarding strategies. The strategy module makes forwarding decisions for each Interest packet.
3. The Content Store (CS) or buffer memory temporarily buffers data packets for data retrieval efficiency.

When a NDN router receives an Interest packet, it first checks its CS (cache). If there is no copy of the requested content, it looks up its PIT table. If the same name is already in the PIT and the arrival interface of the present Interest is already in the set of arrival interface of the corresponding PIT entry, the Interest is discarded. If a PIT entry for the same name exists, the router updates the PIT entry by adding a new arrival interface to the set. The Interest is not forwarded further. Otherwise, the router creates a new PIT entry and forwards the present Interest using its FIB. When an Interest packet is satisfied by the content's origin where it was published, on the way back, all the intermediate nodes store a copy of content in their caches to answer to probable same Interest requests from subsequent requester [45, 43].

3. Time series analysis

A time series is a sequence of data points or time ordered observations $(y_1, y_2, ..., y_t)$ in which each period recorded at a specific time t. A time series forecasting is the use of a model to predict future values based on previously observed values [29, 46]. For time series feature extraction, a trace (set of events) should be converted into *time series* with the regular time intervals. This will be used as an input for the purpose of prediction. The time series would be described by the following formula [47]:

$$x(t+\tau) = f(x(t), x(t-\tau), ..., x(t-n\tau)) \quad (1)$$

Where, f is a Time Series Forecasting (TSF) method, τ is specified time delay and n is some integer values. The TSF methods have found applications in very wide area including finance, business, computer science, medicine, physics, chemistry and many interdisciplinary fields. Most time series modeling methods provide only a reasonable, but limited accuracy and suffer from the assumptions of stationarity and linearity [48]. To improve TSF with nonlinear characteristics, several researchers have successfully employed artificial neural networks [49, 50].

4. Particle Swarm Optimization (PSO)

The PSO was firstly introduced by Kennedy and Eberhart in 1995 [51]. It was inspired by the social behavior of a bird flock or fish school. It is a computational method that tries to optimize a problem iteratively by improving a candidate solution with regard to a given measure of quality as utility (cost) function. PSO optimizes a problem by having a population (called a swarm) of candidate solutions (called particles) and moving these particles around in the search space according to simple mathematical formula over the particle's position and velocity. The movements of the particles are guided by their own best known position (called local best) as well as the entire swarm's best known position (called global best) [52]. The structure of the velocity and the position updates is shown in Fig. 1. In the first movement, particle moves slightly toward the front in the previous direction with the same speed. Afterward, it moves slightly toward the previous local best. Finally, it moves slightly toward the global best [44, 53]. At each iteration, the velocity and the position of each particle are defined according to Eqs. (2) and (3), respectively:

$$\begin{aligned} V_i(t) = \omega * V_i(t-1) + c_1\varphi_1(P_i - X_i(t-1)) \\ +c_2\varphi_2(G - X_i(t-1)) \end{aligned} \quad (2)$$

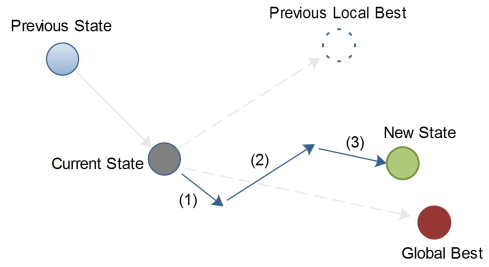


Figure 1: The velocity and the position updates in PSO for a 2-dimensional parameter space [43]

$$X_i(t) = X_i(t-1) + V_i(t) \tag{3}$$

Where i is the number of the particles in the swarm, $V(t)$ is the particle's velocity at time t, $X(t)$ is the particle's position at time t, P_i is the personal best of the ith particle, G is the global best of the entire swarm, ω is the inertia weight (a larger ω performs more efficient global search and smaller one performs more effective local search), c_1 (cognitive parameter) and c_2 (social parameter) are constants which control the search space between the local best position and the global best position (generally $c_1 = c_2 = 2$ [54]). Parameters φ_1 and φ_2 are random numbers uniformly distributed within [0 1]. Eberhart and Shi [55] present a formula to properly balance between exploration (global search) and exploitation (local search) over ω:

$$\omega = \omega_{max} - t \cdot \frac{(\omega_{max} - \omega_{min})}{T} \tag{4}$$

Where $\omega_{max}, \omega_{min}, T$ and t denote the maximum inertia weight, the minimum inertia weight, the total and the current number of iterations, respectively. According to Eq. 4, the inertia weight is uniformly varying from its maximum value to the minimum one.

5. Genetic Algorithm (GA)

Genetic Algorithm (GA) is a search heuristic and stochastic optimization technique based on biological evolution theory and genetic principles developed by Holland on 1975. GA adopts a group of simulated encoded chromosomes and calculates the fitness function of these chromosomes. GA applies three kinds of genetic operators: selection, crossover and mutation to produce next generation. This evolution process continues until the stopping criteria are met. The selection operator chooses chromosomes from a population for later breeding (recombination or crossover). The crossover operator combines (mates) two chromosomes (parents) to produce a new chromosome (offspring). The idea behind crossover is that the new chromosomes might be better than both of the parents if it takes the best characteristics from each of the parents. The mutation operator alters one or more gene values in a chromosome from its initial state. This can result in entirely new gene values being added to the gene pool. With these new gene values, the genetic algorithm might be able to arrive at better solution than was previously possible. Mutation helps the genetic algorithm to avoid being trapped in a local optimal. GA is appropriate for large-sized and nonlinear space problems which solution is unpredictable [56]. One of the main advantages of the use of GA is that it is less likely to fall into a certain local minimum or maximum [57, 58].

6. MLP Neural Network

In the last years, various neural network models have been developed for different applications including signal processing, pattern recognition, system modeling and so on [59]. The multi-layer perceptron (MLP) with back-propagation learning is the most popular and commonly used neural network structure due to its simplicity, effectiveness and excellent performance in many applications that require to learn complex patterns [60, 61]. Multi Layer perceptron (MLP) is a feed-forward neural network with one or more hidden layers between input and output layer. Feed-forward means that data flows are in the forward direction, from input to output layer. MLP can solve problems which are not linearly separable [62]. A graphical representation of a MLP is shown in Fig. 2. In the training phase

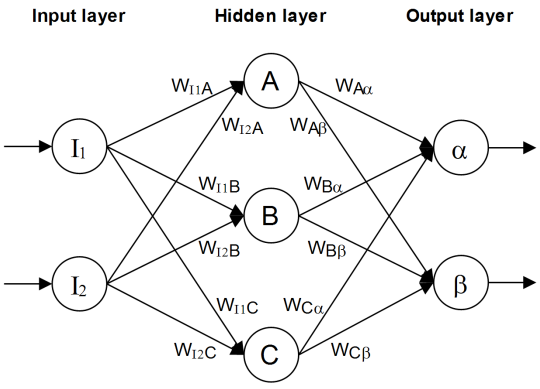


Figure 2: Structure of a three-layer MLP

of the MLP, the training set is presented at the input layer and the parameters of the network (weights and

biases) are dynamically adjusted using gradient-descent based delta-learning rule (back-propagation learning) to achieve the desired output [63, 64]. The training process of MLP neural network is defined as follows:

`Step 1:` *Network initialization.* The connection weights and bias of the network are initialized randomly, setting up the network learning rate η, the error threshold ε, and the maximum iterations T.

`Step 2:` *Data preprocessing.* Data samples are usually partitioned into three sets: training, validation and test. The training sets are used for training (to adjust the weights and biases) the network; the validation sets are the part that assesses or validates the predictive ability of the model during the training to minimize overfitting; the test sets are used for independent assessment of the model's predictive ability (generalization performance) after training.

`Step 3:` *Training network.* input the training sets into MLP, compute network predicted output values, and calculate the error E between output and the target value. The error function is defined as follows:

$$E = \frac{1}{2}\sum_{k}^{m}(\hat{y}(k) - y(k))^2 \tag{5}$$

Where,

$$\hat{y}(k) = \phi_k(\sum_{i=1}^{m} W_{ik}h(i)) \quad k = 1, 2, ..., l. \tag{6}$$

Where, ϕ_k is an activation function for neuron k in hidden layer, $h(i)$ is the output value for neuron (node) i in the hidden layer, W_{ik} is weight connection between neuron i in hidden layer and neuron k in output layer, l and m are the number of neurons for output layer and the hidden layer, respectively. $h(i)$ is calculated by:

$$h(i) = \phi_i(\sum_{j=1}^{n} W_{ij}X_j + b_i) \quad i = 1, 2, ..., m \tag{7}$$

Where, ϕ_i is an activation function for neuron i in hidden layer, W_{ji} is the weight connection between neuron i and input j, X is input value, and b_i is the bias connection of neuron i in hidden layer.

`Step 4:` *Updating the weights and biases.* update network weights and biases according to the prediction error E, making the predictive value of the network as close to actual values through a Back-propagation algorithm.

`Step 5:` *Judgment of whether the end condition is met.* If $E <= \varepsilon$, network training is stopped and go to step 7.

`Step 6:` *Judgment of whether an overfitting has occurred.* If accuracy of the validation error has not been satisfied network training is stopped and go to step 7; otherwise, return to step 3 to continue training.

`Step 7:` *Judgment of generalization performance.* Run test data set by trained network for generalization performance measurement.

`Step 8:` *Further usage.* if the prediction error of the network is acceptable, use the network for further usage; otherwise, go to the Step 1 and train the network again until an ideal network with desire accuracy is found.

7. Fuzzy Set

In classical set theory, an element either belongs or not to a set of elements. Therefore, the membership evaluation is boolean. A more flexible approach would be fuzzy set theory, where elements belong to sets with certain degree of membership that takes its value in the interval [0 1]. This makes fuzzy set theory suitable for complex models where some things are not either entirely true nor entirely false and where the problems are somehow ambiguous or it is needed to manage subjective judgments or opinions [65]. In our scenario, it could be used to decide things like "Is the PIT entry rate average or very high?". Therefore, fuzzy set theory can be successfully employed when most of the decision making attributes are qualitative in nature, with the possibility of subjective assessment [66]. In fuzzy set theory, a *linguistic variable* is a variable whose values are words or sentences in natural or artificial language [67]. A fuzzy rule is defined as a conditional statement in the form:

$$IF\ x\ is\ A \quad THEN\ y\ is\ B \tag{8}$$

Where x and y are linguistic variables; A and B are linguistic values determined by fuzzy sets on the universe of discourse X and Y, respectively. These rules are then mathematically represented by a membership function. The membership provides a measure of the degree of presence for every element in the set [68]. A fuzzy system often consists of four main parts: fuzzification, rules, inference engine, and defuzzification [43]. In the fuzzification step, a crisp set of input data is converted to a fuzzy set using fuzzy linguistic terms and membership functions. In step 2, a list of fuzzy statements are constructed to create what is called "rule base". That rule base will be used to process the fuzzy data by a computational unit, which will output again fuzzy sets. In the defuzzification step, that fuzzy output is mapped to a crisp (non-fuzzy) output using the membership functions.

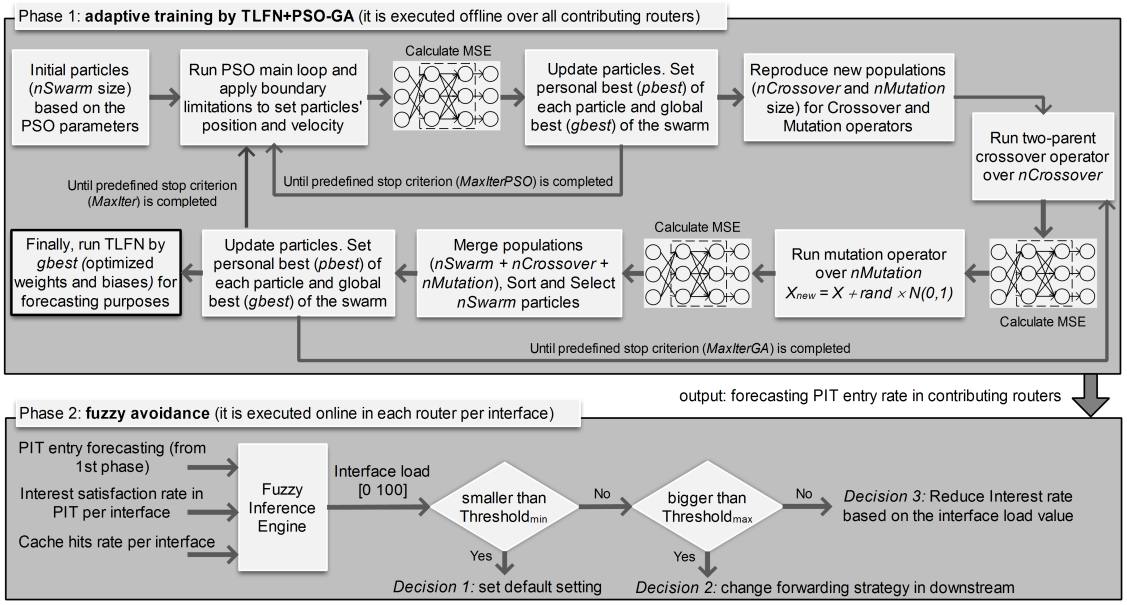


Figure 3: Two steps of the ACCPndn

8. The proposed method: ACCPndn

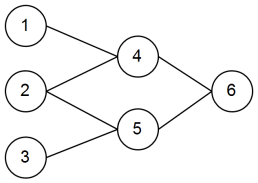
In this section, we introduce our proposal, ACCPndn: a two phase framework for congestion control. The first phase being adaptive training and the second one fuzzy avoidance. This proposal aims to avoid congestion before it builds up. A diagram of the two phases of the ACCPndn is shown in Fig. 3.

8.1. Phase 1: Adaptive Training

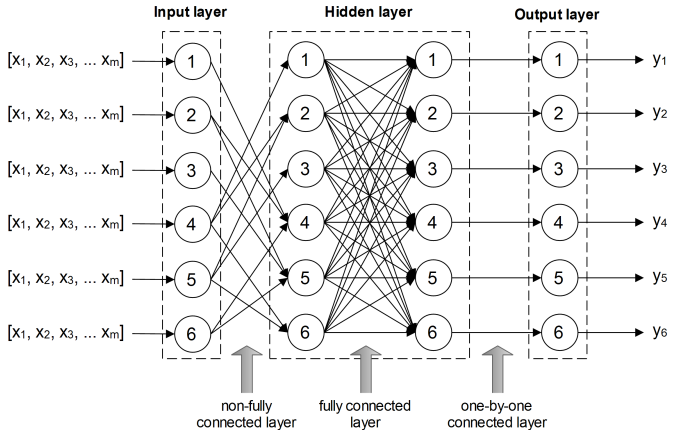
For TLFN modeling, we try to forecast the rate at which entries are added to the PIT table in NDN routers (In the rest of the paper we use the term *PIT entries rate* interchangeably). Since all exchange in NDN is Interest/Data (one Interest packet will be replied back with one data packet), the rate of new PIT entries (the expected amount of returned data) could be a good indicator of a future congestion in the router's buffer. With the prediction of new PIT entries rate in the next time interval, the arrival rate of data packets at that are susceptible to create congestion can be forecast in a timely manner. In this phase, routers learn what are the kind of many low and high frequent traffic patterns which cause an overflow in the routers' buffers and create congestion.

We adopt the nodes connectivity of the NDN routers for defining the number of neural network layers, the connectivity of layers and the number of neurons in TLFN. Fig. 4 shows the logic of the proposed neural network connectivity. The neural network used consists of $m \times n$ input nodes, two hidden layers and one output layer containing n node. The m denotes the number of features in the input layer and n denotes the number of the contributing routers. The input features correspond to the PIT entries rates for a set of consecutive time intervals.

For the connectivity between input layer and first hidden layer, the neural network reflects the connectivity of the data network by only allowing links between neurons representing adjacent nodes. For instance in Fig. 4a, n would be six because only six nodes are actually contributing traffic. Hence, the connection between node 1 in the input layer and node 4 in the first hidden layer derived from their connectivity in the data communication network. We only allow connectivity between nodes neighbors from input neurons (nodes) to represented neurons (nodes) in the hidden layer. On the other hand, according to the definition of the NDN data communication, when a node's cache cannot satisfy Interest packets, the node forwards Interest packets toward the origin's content through intermediate nodes. Thus, there are data communications by routing Interest and returning back data packets through intermediate routers. To address this issue, we provide an extra layer in the hidden layer by a fully-connection communication. The output of the neural network is a representation of PIT entries rate forecasting in each considered routers which are suspected causing the problem in the next time interval. For instance, an output of [50 0 0 30 90 0] would



(a) example of data communication network



(b) neural network representing the network connectivity

Figure 4: Reflection of the connectivity of data communication network in a neural network considered

mean the first, fourth and fifth routers will be faced with the new PIT entries with the probability of 50%, 30% and 90% in the next time interval, respectively.

During the training and analysis, our assumption is that the all network elements (routers and links) are switched on and the network topology is constant. We plan to investigate an approach as switching off network elements while still guaranteeing the least packet drop and maximum link utilization during congestion in the future work.

The constructed TLFN is trained offline in order to create a pattern by learning the PIT entries rate in contributing routers in the next time interval. Afterward, we create a control agent containing this trained neural network to being placed in the simulation environment. A higher level view of our architecture is a network with a control agent existing somewhere on a node in the network. This controller should easily gather required input information (historical PIT entries rate) from contributing routers in a predefined time interval. When the controller predicts the rate of PIT entries in contributing routers, it sends the prediction rate to the corresponding routers. Then, each router per interface performs a fuzzy decision making to control or prevent the probable packet drop in a timely manner (see section 8.2).

For TLFN modeling, we propose a new technique, an hybrid of particle swarm optimization and genetic algorithm during the TLFN training. The TLFN+PSO-GA integrates PSO and GA to tune (optimize) weights and biases of TLFN. The computational procedures for the proposed TLFN+PSO-GA are as follows:

`Step 1:` Normalize data samples into [0 1]:

$$X = \frac{X_i - min(X_i)}{max(X_i) - min(X_i)} \tag{9}$$

Where $min(X_i)$ and $max(X_i)$ are the minimum and maximum value of data samples and X_i denotes the real value of each vector.

`Step 2:` Define some initial parameters: $c_1 = c_2 = 2$, $\omega_{min} = 0.3$, $\omega_{max} = 0.9$, $nSwarm = 25$ (number of the particles), $MaxIter = 500$ (maximum number of the main iteration), $MaxIter_{PSO} = 4$ (maximum number of the PSO iteration), $MaxIter_{GA} = 2$ (maximum number of the GA iteration), $Var_{min} = 0$ (lower bound of variables -particles' position-) and $Var_{max} = 1$ (upper bound of variables).

`Step 3:` Randomly initialize a group of particles in size of $nSwarm$. Each particle includes position (the weights and the biases of TLFN) and velocity.

`Step 4:` Calculate the particles' fitness value. The performance (fitness) function is Mean Square Error (MSE) between the actual target and output of the neural network. Afterwards, update the *pbest* (personal best of each particle) and the *gbest* (global best of the swarm).

`Step 5:` Repeat the following loop until the target or maximum sub-iteration of PSO ($MaxIter_{PSO}$) is reached:

`Step 6:` Apply PSO main loop:

1. Update velocity by Eq. 2.
2. Apply velocity limits: If the velocity of a particle exceeds the minimum or maximum allowed speed limit, it should bring such particle back into the

search space:

$$\begin{aligned} Velocity &= max(Velocity, Vel_{Min}) \\ Velocity &= min(Velocity, Vel_{Max}) \end{aligned} \quad (10)$$

Where $Vel_{Min} = 0.1 \times (Var_{Max} - Var_{Min})$ and $Vel_{Min} = -Vel_{Max}$ are the minimum and maximum values of the particles' velocity.

3. Update position by Eq. 3.
4. Apply velocity reflection: it allows those particles' position that move toward the outside the boundary $[Var_{Min}\ Var_{Max}]$ to move back into the search space multiplying particles' velocity by -1.
5. Apply position limits: If the position of the particle exceeds the boundary of search space, this such particle should bring back into the feasible search space:

$$\begin{aligned} Position &= max(Position, Var_{Min}) \\ Position &= min(Position, Var_{Max}) \end{aligned} \quad (11)$$

6. Evaluate fitness function.
7. Update the personal best (*pbest*) and the global best (*gbest*).

`Step 7:` If $MaxIter_{PSO}$ is not reached to its predefined value go to the `step 6`; otherwise, if all particles updated and $MaxIter_{PSO}$ is reached to its predefined value go to the next step.
`Step 8:` Apply real-coded GA operators: reproduction, crossover, mutation, selection:

1. *Reproduction:* reproduce a number of individuals (chromosomes) for crossover and mutation:

$$nCrossover = \left\lfloor (pCrossover \times \frac{nSwarm}{2}) \times 2 \right\rfloor \quad (12)$$

Where, $pCrossover$(=0.7) is crossover percentage, $nCrossover$ is the number of parents (Offsprings).

$$nMutation = \lfloor pMutation \times nSwarm \rfloor \quad (13)$$

Where, $pMutation$(=0.2) is mutation percentage, $nMutation$ is the number of mutants.

2. *Crossover:* apply two-point crossover over two random selected particles for the number of $nCrossover$ particles (individuals). It creates new population set as $pop_{Crossover}$. Calculate fitness function for $pop_{Crossover}$.
3. *Mutation:* apply mutation over random selected particle for the number of $nMutation$ particles:

$$X_{new} = X \times rand \times N(0, 1) \quad (14)$$

It creates new population set as $pop_{Mutation}$. Calculate fitness function for $pop_{Mutation}$.

4. *Selection:* merge populations ($[nSwarm\ pop_{Crossover}\ pop_{Mutation}]$), sort merged populations based on their fitness values, and select the first $nSwarm$ particles.

`Step 9:` Update *pbest* and *gbest*.
`Step 10:` If the sub-iteration of GA algorithm ($MaxIter_{GA}$) is not reached to its predefined value go to the `step 8`; otherwise, go to the next step.
`Step 11:` Update ω by Eq. 4.
`Step 12:` If the maximum iteration ($MaxIter$) or predefined target is not reached, go to the `step 5`; otherwise, the *gbest* includes the optimized parameters (weights and biases) of TLFN+PSO-GA and the network can be used for forecasting.

8.2. Phase 2: Fuzzy Avoidance

As we explained earlier, a controller based on the trained TLFN+PSO-GA is placed in the network to gather required information for PIT entries rate forecasting in the contributing routers. In this phase, a Fuzzy Inference System (FIS) is applied to prevent probable packet drop in susceptible routers to congestion problem before building up. We deploy a combination of three criteria where each interface in contributing routers gathers them in each time interval:

1. PIT entries rate forecasting in each router through the first phase (training module).
2. Interest satisfaction rate in PIT per interface.
   We take into consideration the unique feature of NDN, i.e., one Interest packet will only return back one data packet in reverse path of the Interest packet. In a NDN router, if the number (rate) of incoming Interest packets in PIT be varied widely with the number (rate) of incoming data packets for Interest satisfaction, there might be some abnormal Interests or congestion. If the number of incoming Interest packets be more than the PIT size, PIT will apply its replacement policy for new incoming Interest packets. If the PIT removes old PIT entries to accommodate new Interest packets, returned data packets for removed PIT entries become *unsolicited*. It might be led to congestion due to the crowding of unsolicited data packets. On the other hand, current unsatisfied Interest packets in PIT table may also reach their timeout (lifetime expiration) and become *dangling state* [20]. Such dangling state can potentially block other Interest packets. When a router is congested, it can potentially lead to dangling state for unsatisfied Interest packets. We maintain the Relative Strength Index

(RSI) for every interface of a router to reflect the Interest satisfaction ratio in PIT:

$$RSI = \frac{\hat{I}_n}{\hat{I}_n + \hat{D}_n} \tag{15}$$

Where $\hat{I}_n$ and $\hat{D}_n$ are the average number of the placed Interests in the PIT table and the incoming data packets of an interface at the nth time, respectively. We apply the standard Exponentially Weighted Moving Average (EWMA) with α coefficient (a lower α counts widely earlier observations) [69] to calculate the placed Interest packets in PIT and the incoming data packets periodically, e.g., once a second:

$$\begin{aligned} \hat{I}_n &= \alpha.I_n + (1-\alpha)\ \hat{I_{n-1}} \\ \hat{D}_n &= \alpha.D_n + (1-\alpha)\ \hat{D_{n-1}} \end{aligned} \tag{16}$$

Where I_n and D_n are the total number of incoming Interest in PIT and incoming data packets of an interface in the nth period. Generally, the reasonable RSI of every interface should be around 50%.

3. Cache hits rate per interface.
If an interface satisfies the most arrival Interest packets by cache, it should be significantly considered in the decision making. Otherwise, if an interface of a suspected router to congestion just fills up PIT table, it should be negatively considered in decision making. We apply Exponential Moving Average (EMA) to calculate the new average of the cache hits ratio in the recent nth time interval. It applies weighting factors which decrease exponentially (the weighting for each older datum decreases exponentially):

$$Cache\ hits = \frac{C_1 + (1-\alpha)C_2 + ... + (1-\alpha)^{n-1}C_n}{1 + (1-\alpha) + ... + (1-\alpha)^{n-1}} \tag{17}$$

Where, C_1 denotes to the number of current cache hit and C_n is the number of the cache hits in the recent nth time.

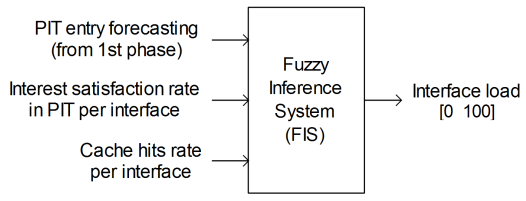


Figure 5: Proposed FIS for fuzzy congestion control

These three criteria themselves involve fuzziness because of bringing vague, imprecise or uncertain information along in problem solving. For instance, the exact value 0.7 (or 70%) cannot show that it is very high percentage or partially high percentage of occurrence an event (e.g., one of the three applied criteria). With the uncertainty modeling, fuzziness subjective, incomplete and imprecise data can be described. Thus, a fuzzy control system can construct a control system in terms of many-values logic dealing with reasoning, i.e., approximate rather than fixed or exact.

The output of proposed FIS is the amount of interface load in a router. Interface load means the portion of the corresponding interface in filling PIT table entries up in that router. The structure of the proposed fuzzy inference system is depicted in Fig. 5. When the controller forecasts PIT entries rate and sends them to the corresponding routers, the proposed fuzzy control system is called in each contributing router to apply three types of control per interface including (1) readjust Interest packets rate, (2) effect on forwarding strategy in the downstream and (3) set default configuration. We set two thresholds ($threshold_{min}$ and $threshold_{max}$) to make decision regarding to the crisp output of the proposed FIS. We set $threshold_{min}$ and $threshold_{max}$ to 20% and 80%, respectively. The procedure of the fuzzy decision making approach is depicted in Fig. 6. According to Fig. 6, if the output of FIS from R_{ij} is bigger than $threshold_{max}$, there is more likely that interface j in router i will face to a malignant congestion (very high packet drop in router's buffer) and the best decision can be changing the interface j status to the unavailable (cannot bring data back) and will be deactive for a predefined time interval t in order to the downstream (neighbor router) does not send Interest packets. It allows to downstream to forward its Interest packets via other available links. If the output of the FIS is between [$threshold_{min}$ $threshold_{max}$] in an interface j, there is more likely by controlling the rate of Interest packets in the downstream in a predefined time interval t, the probability of packet drop reduces considerably. Finally, if the output of FIS in an interface j in router i is lower than $threshold_{min}$, there is more likely that there is no congestion (packet drop) in the next time interval and the configuration of an interface j can be set to its default values: set original Interest packet rate and/or make available (can bring data back) the downstream interface j.

We apply an ascending penalty for definition of time interval t during two restrictions (Interest packets rate and interface unavailability). If in an interface j in a time T, a restriction is needed, counter sets to 1 sec. If in

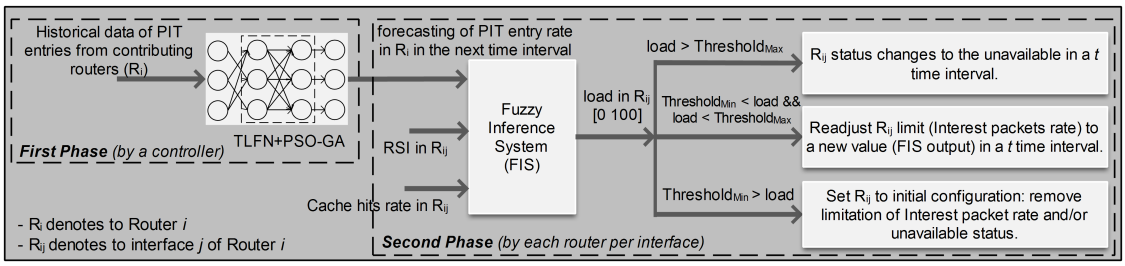


Figure 6: ACCPndn: the first phase by a controller and the second phase by routers per interface

the next time interval $T + 1$ a same restriction again be reported, counter sets to 2 sec. Our ascending penalty method is in $2^{counter}$. Initially, counter sets to 0 and increase linearly in each time interval. The counter will set to the initial value when the output of FIS be lower than $threshold_{min}$ in the next time interval. This ascending penalty intensifies the restriction to avoid packet drop in the long-term.

9. Evaluation Setup

We use simulations to quantify effect of congestion and its countermeasure. In this work, we used the open-source ndnSIM [70] package, which implements NDN protocol stack for NS-3 network simulator (`http://www.nsnam.org/`), to run simulations for evaluating the performance of proposed method. ndnSIM simulation environment reproduces the basic structures of a NDN node (i.e., CS, PIT, FIB, strategy layer, and so on). The proposed adaptive training method (first phase) was implemented by the MATLAB software on an Intel Pentium 2.13 GHz CPU, 4 GB RAM running Windows 7 Ultimate. This algorithm deployed to C++ project integrating as a C++ shared library using the MATLAB compiler. Then, this C++ program was integrated with ndnSIM environment to be able to adjust in the simulation environment. The proposed fuzzy avoidance phase was also implemented with C++ in ndnSIM environment. We choose two metrics to quantify the effectiveness of our countermeasure. First criterion is the average of utilization of multiple paths (retry alternative paths) to mitigate congestion in bottleneck links. The indicator for evaluating the utilization of bottleneck links and alternative links is the rate of *InData*. InData denotes a number of arrival data in an interface. InData guarantees that this amount of data packet was actually transferred over the channel during the congestion. Second criterion is the average of packet drop rate. If the proposed ACCPndn considerably decreases or totally removes packet drops, it can be concluded our proposed method is highly effective at mitigating/removing packet drops.

Our experiments are performed over two topologies shown in Fig. 7. Fig. 7a corresponds to DFN-like (Deutsche Forschungsnetz as the German Research Network) [71], and Fig. 7b corresponds to the Switch-like (Swiss Education and Research Network) [72]. We use the symbols `Cx`, `Px`, and `Rx` to represent `x`-th consumer, producer and router, respectively. We leave the investigation of the most complex arbitrary network topologies to future work. The properties of the underlying topologies are listed in Table 1, which are defined as follows:

1. Nodes/Edge Nodes: The number of the nodes and the edge nodes.
2. Links: The total number of the links between nodes.
3. Dia. (diameter of the topology): This is the length (number of hops) of the longest path that an Interest packet can be satisfied by a producer/router.
4. Bottleneck: The number of the slowest links in terms of the bandwidth.
5. $Degree_{avg}$: The average of the outgoing links from a node connected to adjacent nodes.
6. $Degree_{std}$: The standard deviation of the outgoing links from a node connected to adjacent nodes.

In spite of various arguments and experiments, there is no typically and properly justification for NDN parameters and they have specified based on authors' experiences and designs [6]. Therefore, the applied control parameters of the ACCPndn are iteratively learned under various network environments to make a real data communication in considered topologies. For scalability reasons, it is important for a congestion control protocol to be able to maintain their properties as network characteristics change. We thus set nodes' PIT size to a range of [700 1000] randomly, while the Interest expiration time was set to the default timeout of 4000 ms. We set the link delay and queue length parameters to

Table 1: Properties of the applied network topologies

Topology	Nodes/Edge Nodes	Links	Dia.	Bottleneck	$Degree_{avg}$	$Degree_{std}$
DFN-like	26/15	36	8	6	2.769	2.25
Switch-like	34/17	45	14	9	2.647	2.028

different values for every node in the simulated topologies. In particular, we set delay and queue length to about 2ms and the range of 200-500 for both considered topologies, respectively. A various PIT entries replacement policies (i.e., perform different actions when limit on number of PIT entries is reached) were adopted randomly over the nodes in both considered topologies including persistent (new entries will be rejected if PIT size reached its limit) and least-recently-used (the oldest entry with minimum number of incoming faces will be removed). Moreover, The nodes' cache capacity was set to 1000 contents, while the caching replacement policies were set to randomly over the nodes including least-recently-used, FIFO (first-input first-output) and random policies. We ran various traffic patterns within the randomize and Zip-Mandelbort (α is in range of [0.4 0.9]) distribution. For both distribution methods, we applied uniform and exponential patterns of distribution. The expected frequency of Interest packets generation sets to a range of 100-1000 packets per second. Each consumer changes five times the frequency randomly during simulation run. We assign some bottleneck links with yellow dash lines in both considered network topologies in Fig. 7. We set bandwidth in the range 1 Mb/s to 3 Mb/s randomly. Table 2 is also shown the Interest-Data communications. Finally, we investigate the transient behavior of the utilization and the packet drop rate at the bottleneck links and alternative paths for congestion control and/or avoidance during simulation run.

Table 2: Interest-Data communications

DFN-like		Switch-like	
Consumer	Producer	Consumer	Producer
C1	P1	C1	P1
C2	P2	C2	P2
C3	P5	C3	P3
C4	P4	C4	P1, P4
C5	P2, P3	C5	P3
C6	P3	C6	P2, P6
C7	P4, P6	C7	P5
C8	P1, P5	C8	P5, P6
C9	P4, P6	C9	P7

10. Evaluation Results

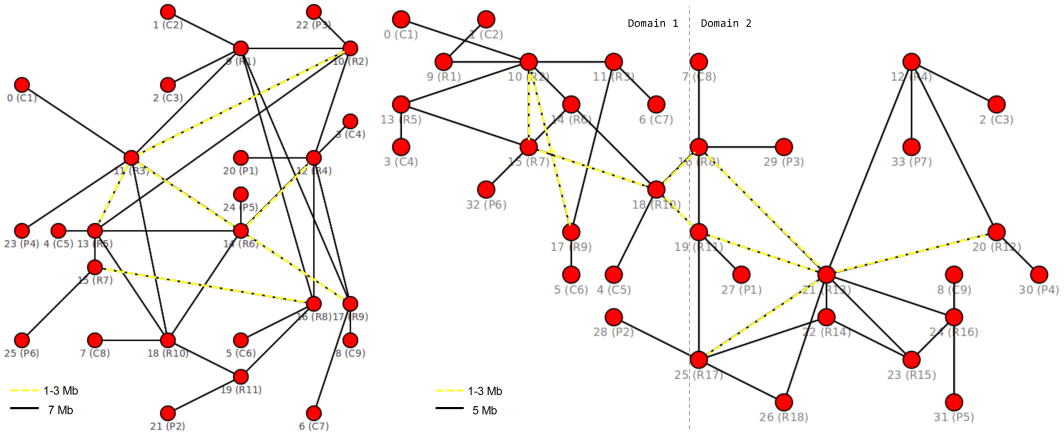
In this section we report the results of evaluation of ACCPndn presented in Section 8.

10.1. Phase 1: adaptive training

Training phase consists of a collection of time ordered observations of PIT entries in contributing routers from two considered NDN network topologies in Fig. 7. Depending on the time scale, there are four main forecasting types including real-time, short-term, middle-term and long-term [73]. The real-time forecasting is the most appropriate type of PIT entries forecasting where samples not exceed a few seconds and requires an on-line forecasting and reaction in a timely manner. The choice of the input time intervals has a crucial effect in the PIT entries forecasting performance. A small number of time intervals will provide insufficient information, while a high number of intervals will increase the probability of irrelevant input features [29]. Several configurations based on our observations of PIT entries fluctuation in considered network topologies were set. Five different sliding windows were empirically adopted based on the predefined time interval (we set 1 sec):

- **DFN-like:**
  1. {1 2 3 6 7 8 11 12 13 24 25 26 38 39 40}
  2. {1 2 3 6 7 8 11 12 13 24 25 26}
  3. {1 2 3 4 5 8 9 10 11 12}
  4. {1 2 3 7 8 9 12 13 14 26 27}
  5. {1 2 3 6 7 8 10 11 12 23 24 25}
- **Switch-like:**
  1. {1 2 3 4 8 9 10 11}
  2. {1 2 3 4 5 10 11 12 13 14}
  3. {1 2 3 6 7 8 10 11 12}
  4. {1 2 3 10 11 12 20 21 22}
  5. {1 2 3 7 8 9 16 17 18}

Due to the application of different configuration settings on considered network topologies, we run the experiments 20 times independently to evaluate the proposed training method (see section 8.1) in terms of applied



(a) DFN-like topology

(b) Switch-like topology

Figure 7: Considered network topologies

performance metrics. The performance of the forecasting model in training phase is evaluated by the Mean Square Error (MSE) and Symmetric Mean Absolute Percent Error (SMAPE):

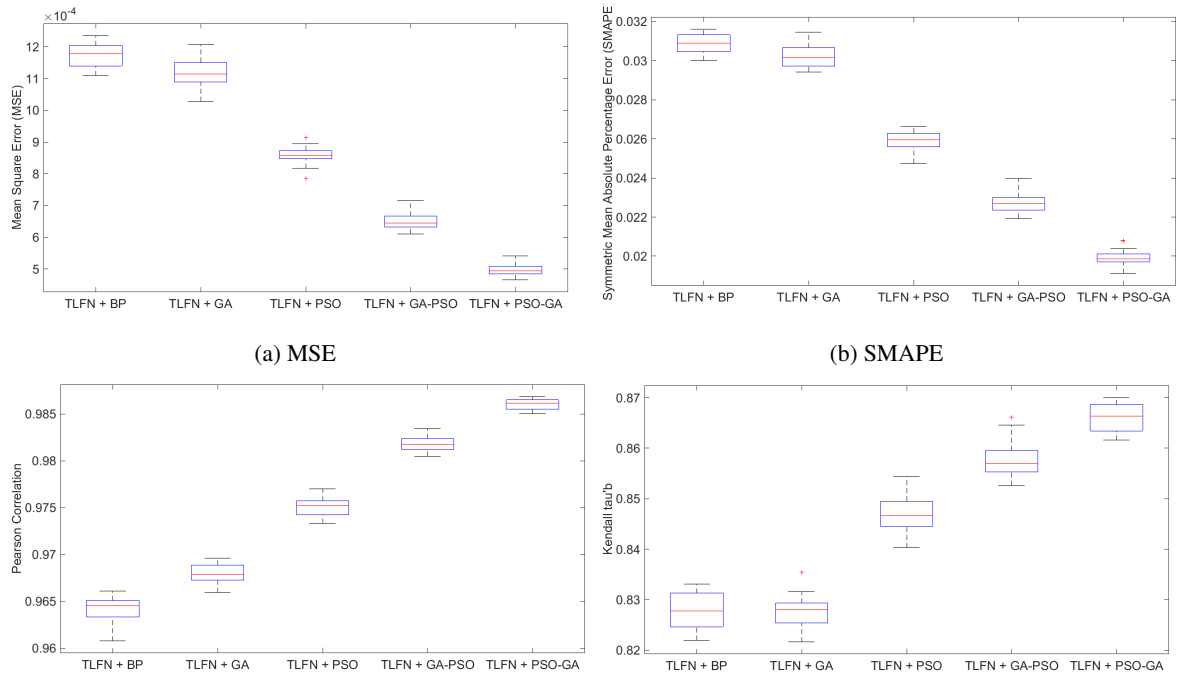
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(T_i - O_i)^2 \tag{18}$$

$$SMAPE = \frac{\sum_{i=1}^{n}|O_i - T_i|}{\sum_{i=1}^{n}(O_i + T_i)} \tag{19}$$

where T is the actual value and O is the forecast value. The MSE quantifies the difference between values implied by an estimator and the true values of the quantity being estimated. The SMAPE is an alternative to Mean Absolute Percent Error (MAPE) when there are zero or near-zero demand for items. It is a measure of accuracy of a method for constructing fitted time series values in statistics. In case of the 20 simulation runs, forecasting methods likely yield different results. Therefore, the forecasting results are investigated by statistical tests if these differences are significant [74]. We have used Pearson correlation coefficient and Kendall's tau'b with 99% of confidence level implemented by MATLAB software. Moreover, the time series data from considered NDN topologies were divided into three contiguous blocks as training (70% of the series) to fit (train) the forecasting models, validation (next 15% of the series) to evaluate the forecasting accuracy during the training and test (remaining 15% of series) to confirm the forecasting accuracy after training.

10.1.1. DFN-like topology

Fig. 8 shows the optimal accuracy derived from the first constructed sliding windows as the best configuration setting. The box plot of TLFN + PSO-GA in Figs. 8a and 8b is comparatively short as compared to other methods. This suggests that overall MSE and SMAPE values are relatively small and have a high level of agreement within 20 runs. TLFN + PSO-GA provides better results than TLFN + GA-PSO which it confirms that GA performs a good local search for better particles movement in the swarm to minimize significantly both applied cost functions. Indeed, TLFN + PSO and TLFN + GA obtain quite good forecasting errors as compared to TLFN + BP by MSE and SMAPE in the almost all 20 runs, respectively. As expected, the hybridization of the optimization algorithms reveal a better performance as compared to standalone TLFN trained by BP.
The short lower and upper whisker in the Figs. 8c and 8d mean that the results of several runs are not varied. As shown, the correlation between different applied algorithms with 99% of confidence level is strong and positive which is statistically significant. The significant statistical correlation between 20 runs is the proposed forecasting method in ACCPndn about 98.5% as compared to other algorithms ranging 96%-98.3%. Moreover, the values for concordance coefficient from Kendall's tau'b results are close to +1, which means that there is a large agreement between the forecasting results. The concordance coefficient of the proposed forecasting method in ACCPndn is better than other algorithms ranging 86%-88%.



(a) MSE

(b) SMAPE

(c) Pearson correlation

(d) Kendall tau'b

Figure 8: The forecasting results in DFN (1st sliding window)

10.1.2. Switch-like topology

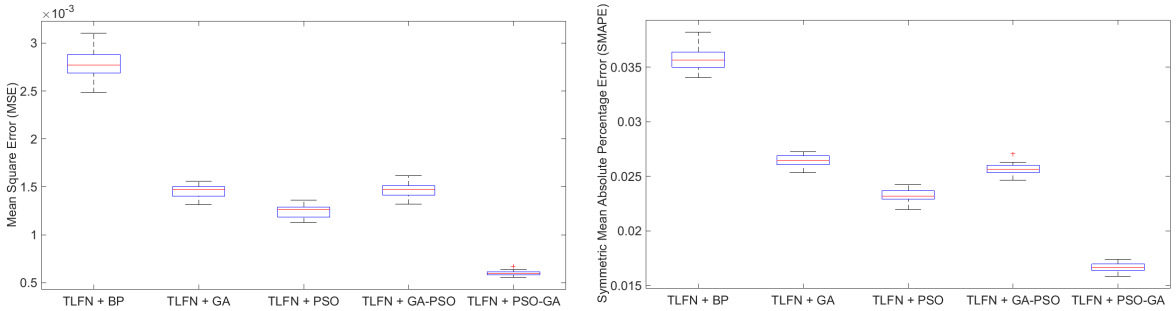
This large network can be formed and decomposed to two separated clusters (or domains). The reason is that the constructed TLFN would face high generalization error and non-proper training due to the poor-trainability and the over-fitting against larger and more complex network topologies. Therefore, we decompose the Switch-like topology graph experimentally to two clusters from the edge routers. This decomposition is depicted in Fig. 7b by vertical dot points. First cluster consists of eight routers (R1, R2, R3, R5, R6, R7, R9 and R10) and the rest (ten routers) appears in the second cluster.

The optimal forecasting performance of the first and the second cluster derived from the third and the second sliding windows is depicted in Figs. 9 and 10, respectively. The box plots clearly illustrate that the proposed TLFN + PSO-GA is able to provide roughly appropriate performance in terms of the MSE and SMAPE.
The extensive analysis in Figs. 9c-9d and Figs. 10c-10d demonstrate that the correlation and the concordance coefficient between results by the proposed training algorithm is more significant than other applied hybridization during 20 runs. Similar to the results of DFN-like topology (see section 10.1.1), TLFN by PSO-GA training satisfies performance criteria more appropriate than by GA-PSO in Switch-like topology. Moreover, as we expected, the application of the optimization algorithms can perform a better performance as compared to standalone training by BP.

10.2. Phase 2: fuzzy avoidance

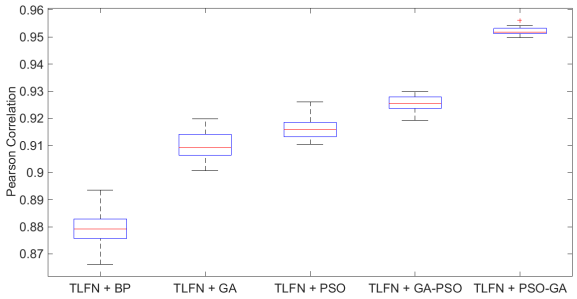
In this study, MATLAB fuzzy logic toolbox is used for fuzzy rule based decision-making regarding to congestion control. The second phase is structured with following components:

1. Three fuzzy set of input variables: (1) RSI rate in R_{ij}, (2) Predicted PIT entries rate in R_i and (3) Cache hits rate in R_{ij}; membership functions: *Low, Medium, High*.
2. A fuzzy set of output variable: Interface load; membership functions: *Negligible, Small load, Moderate load, Overloaded*.
3. Fuzzy membership functions: Since the *sigmoid* membership function [75, 76] is inherently open to the right or to the left; thus, it is appropriate for representing concepts such as "Low", "High" or "Negligible", "Overloaded". The *guass2mf* is also employed for middle linguistic values ("Medium",

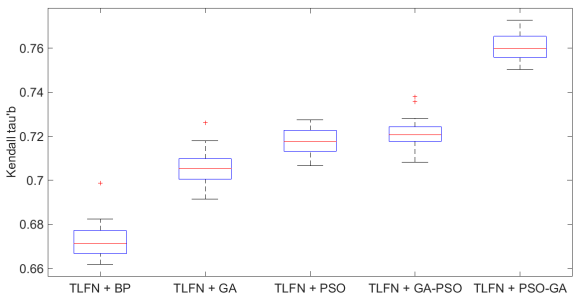


(a) MSE

(b) SMAPE

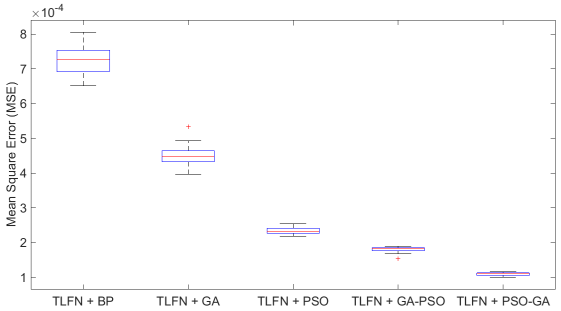


(c) Pearson correlation

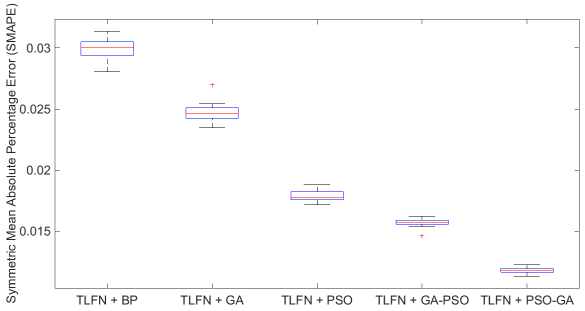


(d) Kendall tau'b

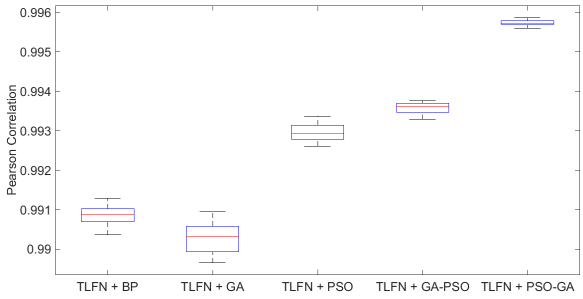
Figure 9: The forecasting results in 1st cluster of Switch-like (3rd sliding window)



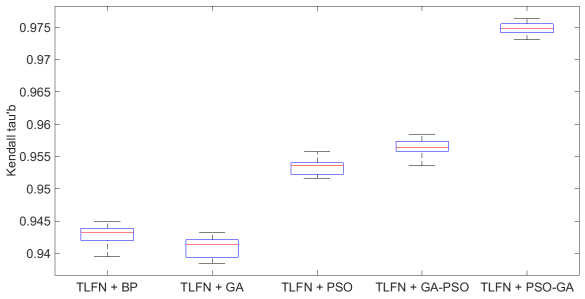
(a) MSE



(b) SMAPE

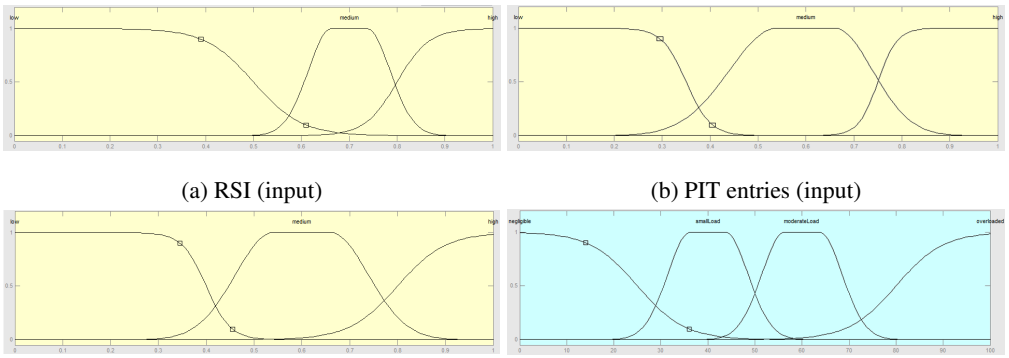


(c) Pearson correlation



(d) Kendall tau'b

Figure 10: The forecasting results in 2nd cluster of Switch-like (2nd sliding window)



(a) RSI (input)

(b) PIT entries (input)

(c) Cache hits (input)

(d) Interface load (output)

Figure 11: The membership functions

"Small load", "Moderate load"). The *gauss2mf* is a kind of smooth membership functions, so the resulting model has a high accuracy. It also covers the universe sufficiently which leads to the completeness of a fuzzy system [77]. The membership functions of input and output variables are shown in Figs. 11a-11d.

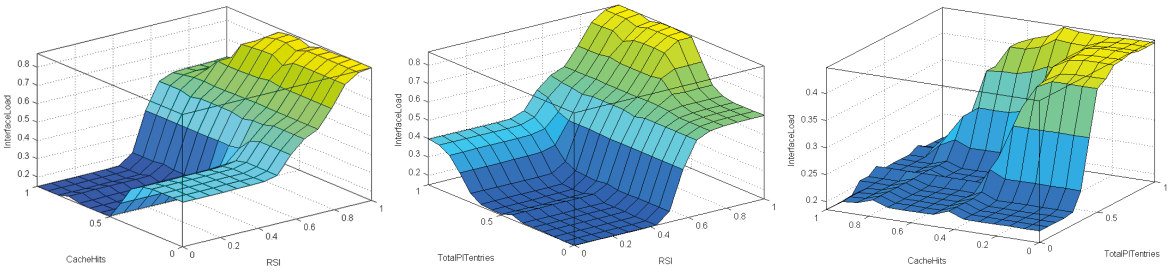
4. Fuzzy rules: 27 rules. The nonlinear control-decision surface is shaped by the constructed rule base and the linguistic values of the inputs and output variables in Fig. 12. According to Fig. 12, the cache hit ratio plays an important role for decision making next to the RSI and PIT entries forecasting, while, the high cache hit ratio might bring the not highly interface load and the low cache hit ratio might bring the highly interface load. Moreover, the RSI criterion plays a main role as far as the increasing the RSI will lead to high interface load. A sample of constructed rule base is as follows:
   ❶ if `CacheHits` is *low* and `PITentry` is *high* and `RSI` is *high* ➔ `InterfaceLoad` is *Overloaded*
   ❷ if `CacheHits` is *medium* and `PITentry` is *high* and `RSI` is *medium* ➔ `InterfaceLoad` is *ModerateLoad*
   ❸ if `CacheHits` is *high* and `PITentry` is *high* and `RSI` is *medium* ➔ `InterfaceLoad` is *SmallLoad*
   ❹ if `CacheHits` is *high* and `PITentry` is *medium* and `RSI` is *low* ➔ `InterfaceLoad` is *Negligible*
5. Inference: Mamdani fuzzy inference by fuzzy set operations as max and min for OR and AND, respectively.
6. Defuzzifier: Center of Gravity algorithm:

$$Center\ of\ Gravity = \frac{\int_{min}^{max} u\ \mu(u)\ d(u)}{\int_{min}^{max} \mu(u)\ d(u)} \qquad (20)$$

Where, u denotes the output variable, μ is the membership function after accumulation, *min* and *max* are lower and upper limit for defuzzification, respectively. A sample solution area (fuzzy inference) of proposed fuzzy detection phase is given in Fig. 13.

10.3. Results and Observations

In this section, we demonstrate through simulations that ACCPndn satisfies applied performance criteria as compared to NACK [20] and HoBHIS [9] methods. The Interest NACK mechanism enables NDN routers to perform quick recovery per interface rate limit to avoid congestion on a local outbound interface. The Hop-by-hop Interest Shaping (HoBHIS) is also a congestion control mechanism by shaping the rate of the Interest which is currently sending towards content providers with routers. NACK, HoBHIS and ACCPndn also have a fundamental difference in the implementation of the algorithm. ACCPndn controls or avoids congestion traffic through an hybridization of TLFN, metahuristics and non-linear fuzzy logic-based control system to predict future PIT entries and perform an adaptive recovery whereas NACK and HoBHIS apply a rate limiting after arriving congestion traffic which prevents the link between the two nodes from being congested. The results demonstrate that ACCPndn outperforms NACK and HoBHIS mechanisms sufficiently. In the training phase of ACCPndn we select the first, the third and the second time intervals configurations for DFN-like, the first cluster and the second cluster of Switch-like topologies, respectively. These time intervals perform better than others in (near) optimal configuration of TLFN + PSO-GA predictor based on the applied performance metrics within 20 runs.



(a) RSI vs. Cache hits (b) RSI vs. Total PIT entries (c) Total PIT entries vs. Cache hits

Figure 12: The surface of the proposed fuzzy control system

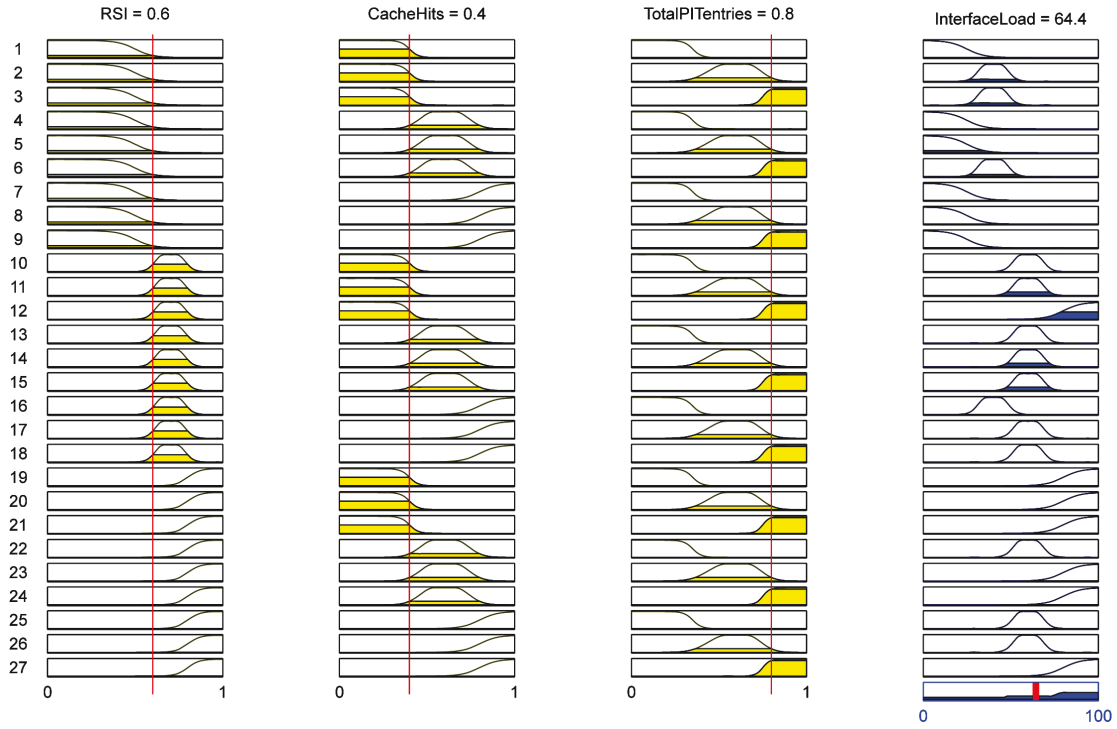
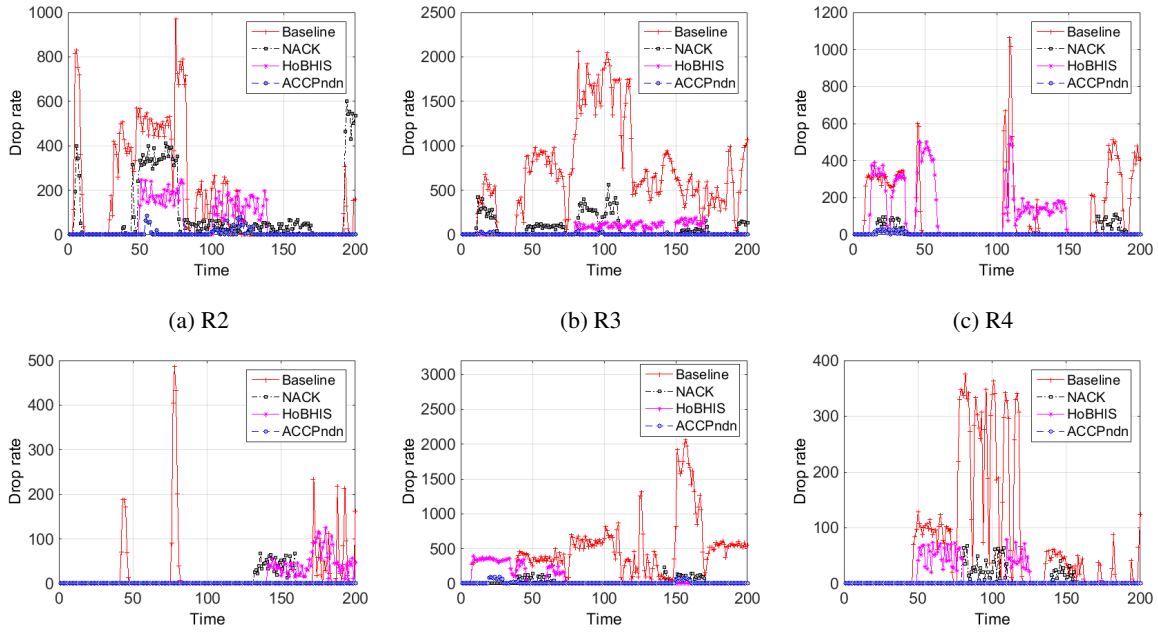


Figure 13: The sample solution area (fuzzy inference) of proposed fuzzy decision-making system

This TLFN + PSO-GA runs iteratively (we set 1 sec) to gather historical information of PIT entries in contributing routers in defined sliding windows in order to predict the PIT entries in the next time interval (see sections 8.1 and 10.1). These amount of predictions are sent to the corresponding routers to run second phase of ACCPndn, i.e., a nonlinear fuzzy control system per interface to control/avoid packet losses to mitigate congestion (see sections 8.2 and 10.2). When the controller runs initially, some time intervals are not available, that we set zero to those time intervals until their time reaches.

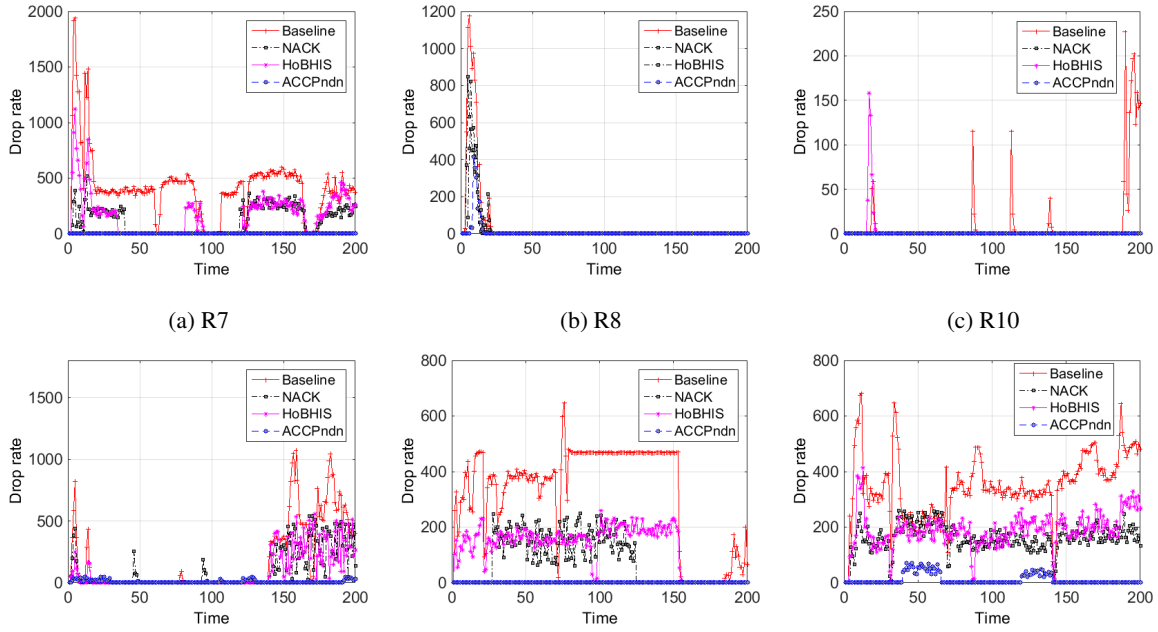
We show the results in four conditions (Baseline, NACK, HoBHIS and ACCPndn) in the bottleneck links to confirm the effectiveness and efficiency of ACCPndn in terms of the applied performance metrics. Figs. 14 and 15 demonstrate the average Data packet drop within 10 runs in DFN-like and Switch-like topologies, respectively. As shown in these figures, there is a considerable benefits of the proposed countermeasure implemented by ACCPndn in reduction of the packet drop. Tables 3 and 4 illustrate the statistics of packet drop rate in DFN-like and Switch-like topologies, respectively. According to these tables, the average number of packet drop



(a) R2 (b) R3 (c) R4

(d) R5 (e) R6 (f) R7

Figure 14: Average of Data drop in contributing routers' buffer in DFN-like topology



(a) R7 (b) R8 (c) R10

(d) R11 (e) R12 (f) R13

Figure 15: Average of Data drop in contributing routers' buffer in Switch-like topology

and its boundary have considerably decreased by ACCPndn as compared to the baseline, NACK and HoB-

Table 3: statistics of packet drop in DFN-like topology (mean of 10 runs)

Routers	Methods	No. drop	Drop boundary	Mean	Std.	SEM (95%)	μ (95%)	σ (95%)
R2	Baseline	108	[1 971]	173.2488	237.2683	32.8018	[140.2479 206.2497]	[216.1195 263.0411]
	NACK	145	[1 600]	95.8657	151.1534	20.8966	[74.8422 116.8891]	[137.6805 167.5722]
	HoBHIS	69	[59 247]	52.5373	78.0588	10.7914	[41.6804 63.3943]	[71.101 86.5377]
	ACCPndn	37	[1 84]	5.1045	14.77	2.0419	[3.0502 7.1588]	[13.4535 16.3744]
R3	Baseline	179	[5 2063]	719.3085	536.6871	74.1958	[644.6624 793.9546]	[488.8498 594.9836]
	NACK	111	[1 565]	81.9303	116.5032	16.1063	[65.7263 98.1344]	[106.1188 129.1582]
	HoBHIS	83	[50 193]	46.7761	60.3131	8.3381	[38.3874 55.1649]	[54.9371 66.8644]
	ACCPndn	29	[1 39]	2.2289	7.2545	1.0029	[1.2199 3.2379]	[6.6078 8.0425]
R4	Baseline	89	[1 1065]	122.0796	193.2822	26.7208	[95.1966 148.9626]	[176.0541 214.2771]
	NACK	42	[15 109]	12.398	27.8455	3.8496	[8.5251 16.2709]	[25.3635 30.8701]
	HoBHIS	92	[1 528]	100.3881	146.412	20.2411	[80.0241 120.752]	[133.3616 162.3157]
	ACCPndn	23	[1 29]	1.8657	6.0495	0.83633	[1.0243 2.7071]	[5.5103 6.7067]
R5	Baseline	40	[1 487]	20.7413	67.6333	9.3502	[11.3344 30.1482]	[61.6048 74.9798]
	NACK	28	[13 68]	5.9801	16.0652	2.221	[3.7456 8.2146]	[14.6332 17.8102]
	HoBHIS	62	[3 125]	14.0348	26.4268	3.6534	[10.3592 17.7104]	[24.0712 29.2973]
	ACCPndn	0	-	-	-	-	-	-
R6	Baseline	194	[17 2066]	494.1692	404.1683	55.8754	[437.9547 550.3836]	[368.143 448.0703]
	NACK	52	[1 232]	20.6368	43.2485	5.979	[14.6215 26.6521]	[39.3936 47.9463]
	HoBHIS	57	[1 395]	71.1791	128.9646	17.8291	[53.2418 89.1164]	[117.4694 142.9731]
	ACCPndn	27	[12 98]	7.9652	22.1243	3.0586	[4.888 11.0424]	[20.1522 24.5275]
R7	Baseline	115	[1 376]	72.209	109.1693	15.0924	[57.0249 87.393]	[99.4386 121.0276]
	NACK	46	[1 68]	6.194	14.4975	2.0042	[4.1776 8.2104]	[13.2053 16.0722]
	HoBHIS	47	[21 79]	11.5522	22.5388	3.1159	[8.4174 14.6871]	[20.5298 24.9871]
	ACCPndn	0	-	-	-	-	-	-

Table 4: statistics of packet drop in Switch-like topology (mean of 10 runs)

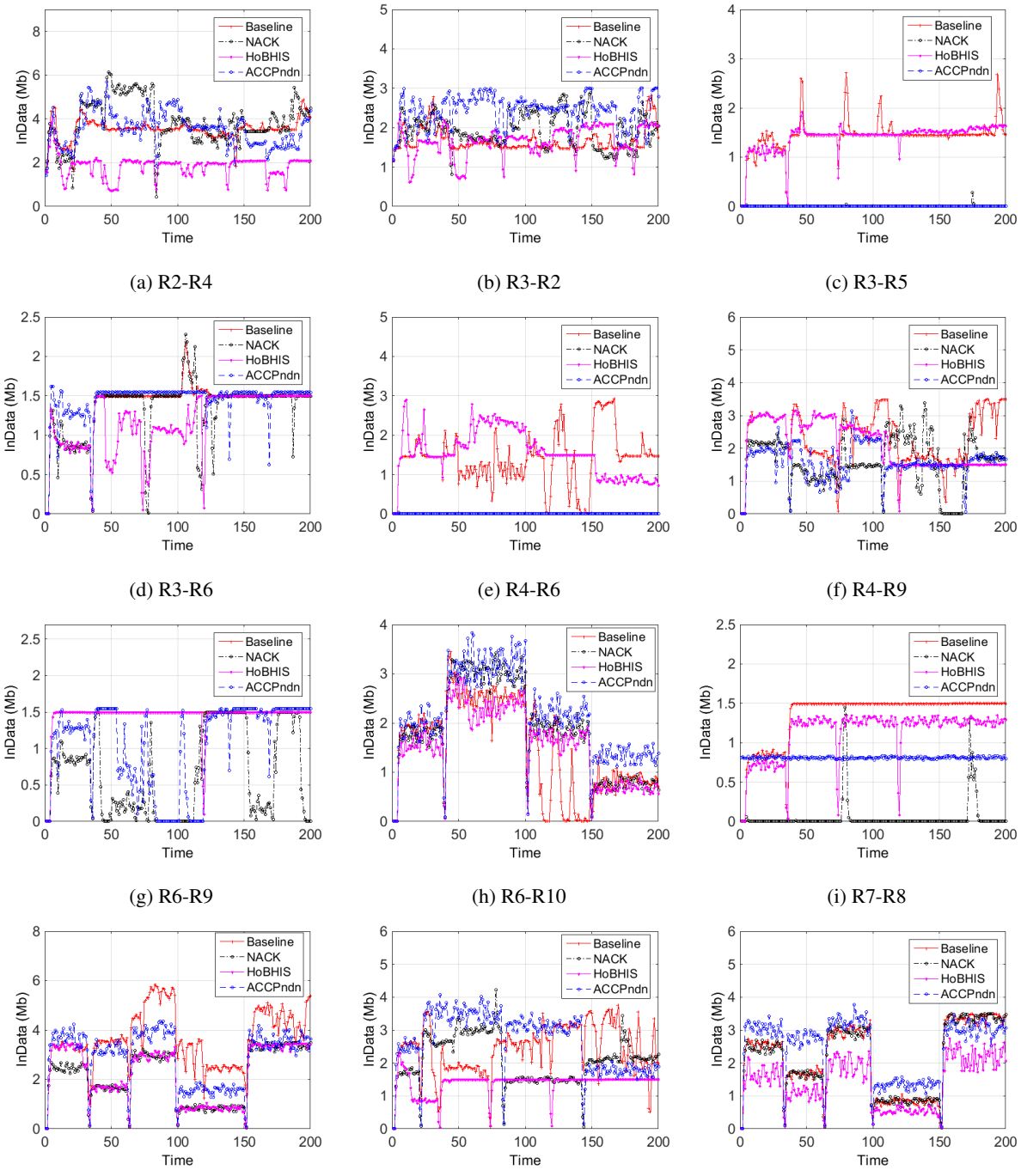
Routers	Methods	No. drop	Drop boundary	Mean	Std.	SEM (95%)	μ (95%)	σ (95%)
R7	Baseline	182	[1 1943]	418.0249	294.9422	40.7751	[377.0023 459.0475]	[268.6527 326.9797]
	NACK	113	[3 521]	118.0498	120.7619	16.6951	[101.2534 134.8461]	[109.9979 133.8794]
	HoBHIS	123	[5 1120]	163.1791	184.8821	25.5595	[137.4644 188.8938]	[168.4028 204.9646]
	ACCPndn	0	-	-	-	-	-	-
R8	Baseline	21	[1 1177]	43.2786	185.2503	25.6104	[17.5127 69.0445]	[168.7382 205.3728]
	NACK	18	[2 564]	19.5224	85.344	11.7986	[7.6522 31.3926]	[77.7369 94.6143]
	HoBHIS	15	[1 848]	23.7861	116.7227	16.1366	[7.5515 40.0207]	[106.3187 129.4014]
	ACCPndn	10	[5 414]	7.6915	46.6156	6.4445	[1.2079 14.1752]	[42.4606 51.6791]
R10	Baseline	27	[1 227]	11.0896	38.4164	5.311	[5.7463 16.4328]	[34.9922 42.5893]
	NACK	0	-	-	-	-	-	-
	HoBHIS	6	[4 158]	2.1045	15.5169	2.1452	[-0.053718 4.2627]	[14.1338 17.2024]
	ACCPndn	0	-	-	-	-	-	-
R11	Baseline	78	[2 1068]	178.796	282.9273	39.1141	[139.4446 218.1475]	[257.7088 313.6597]
	NACK	79	[1 542]	98.2687	159.9374	22.111	[76.0235 120.5138]	[145.6815 177.3103]
	HoBHIS	74	[2 559]	94.7811	157.2892	21.7449	[72.9042 116.658]	[143.2693 174.3744]
	ACCPndn	48	[11 49]	6.7413	13.1428	1.817	[4.9133 8.5693]	[11.9713 14.5704]
R12	Baseline	172	[4 647]	316.8458	184.8103	25.5496	[291.1411 342.5505]	[168.3374 204.885]
	NACK	97	[62 250]	74.9851	85.8511	11.8687	[63.0443 86.9258]	[78.1988 95.1765]
	HoBHIS	155	[1 260]	126.3781	80.6666	11.152	[115.1585 137.5978]	[73.4764 89.4288]
	ACCPndn	0	-	-	-	-	-	-
R13	Baseline	198	[15 683]	350.1343	117.5996	16.2579	[333.7778 366.4909]	[107.1175 130.3737]
	NACK	197	[5 259]	160.6468	50.5236	6.9848	[153.6196 167.6739]	[46.0202 56.0116]
	HoBHIS	198	[9 413]	189.7811	64.9344	8.977	[180.7496 198.8126]	[59.1465 71.9877]
	ACCPndn	47	[22 70]	10.1642	19.6443	2.7158	[7.4319 12.8964]	[17.8933 21.7781]

HIS within 10 runs. We also show the benefit of the ACCPndn by Mean, Standard Deviation (Std.), Standard Error of the Mean (SEM) and lower and upper boundaries of the 95% confidence interval in probability distribution of the amount of packet drop in contributing routers. The total average packet drop rate in both considered topologies is illustrated in Fig. 16. In Figs. 17 and 18 we show the average utilization of the bottleneck links and retrying alternative links in four conditions. Accordingly, we observe that ACCPndn achieves the highest and the better average utilization and retrying alternative links as compared to NACK and HoBHIS.

These highlights confirm that the ACCPndn is effective and efficient in presence of bottleneck links and congestion problems, and outperforms the NACK and the HoBHIS sufficiently.

11. Discussion

In this paper, an Adaptive Congestion Control Protocol in Named Data Networking by learning capaci-



(a) R2-R4 (b) R3-R2 (c) R3-R5

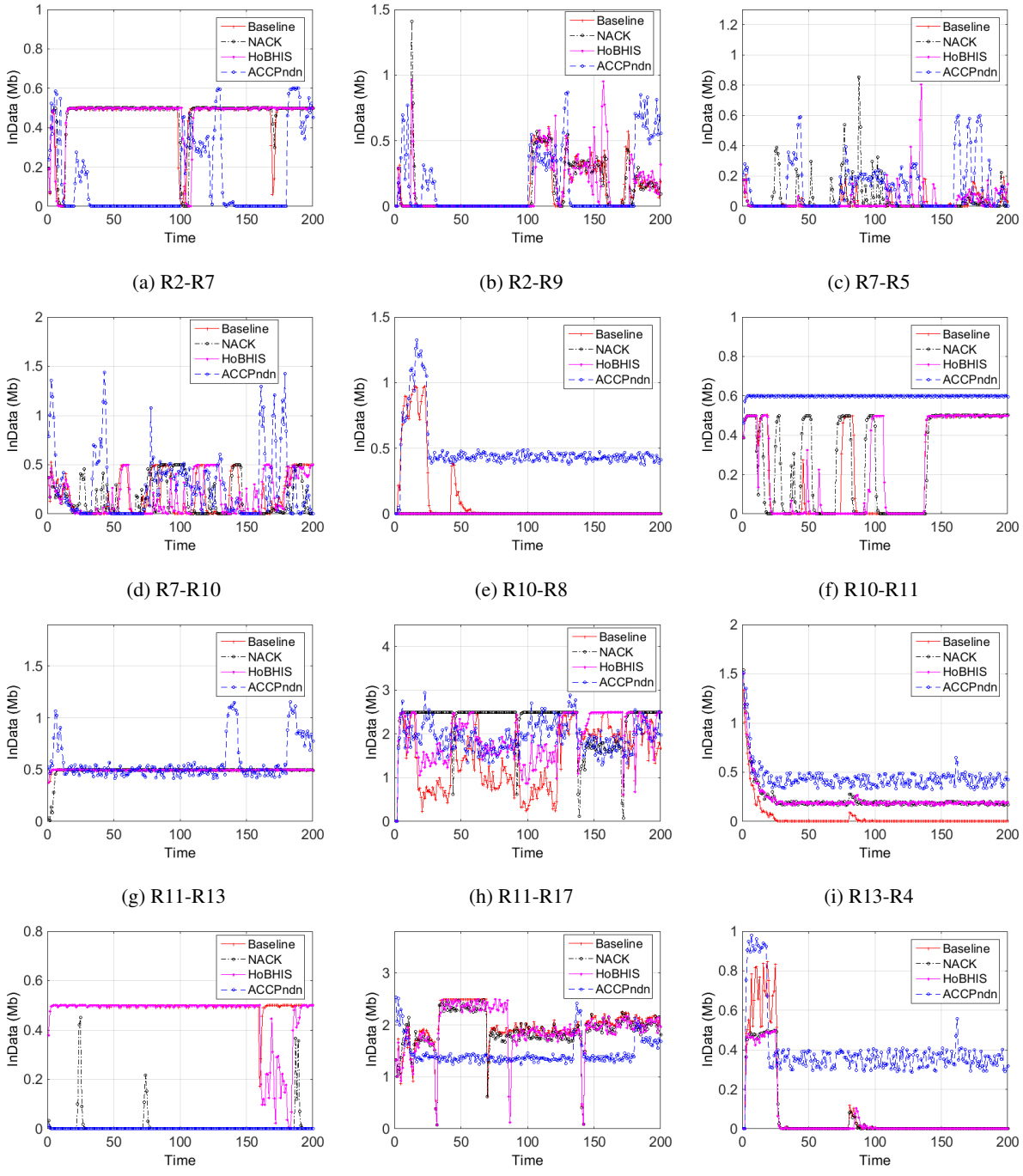
(d) R3-R6 (e) R4-R6 (f) R4-R9

(g) R6-R9 (h) R6-R10 (i) R7-R8

(j) R8-R1 (k) R8-R4 (l) R8-R11

Figure 17: Average of InData in contributing routers in DFN-like topology

ties called ACCPndn has been proposed to detect and respond to the congestion traffic before degrading data delivery performance. ACCPndn works in two phases: (1) *congestion forecasting by an adaptive training* to identify the source of the congestion together with the amount of congestion in the next time interval through



(a) R2-R7 (b) R2-R9 (c) R7-R5

(d) R7-R10 (e) R10-R8 (f) R10-R11

(g) R11-R13 (h) R11-R17 (i) R13-R4

(j) R13-R12 (k) R13-R16 (l) R13-R17

Figure 18: Average of InData in contributing routers in Switch-like topology

TLFN+PSO-GA, and (2) *congestion avoidance by a non-linear fuzzy logic-based control system* to make a proactive decisions against congestion traffic well enough in advance.

Scalability of the ACCPndn is studied over two frequently used network topologies in NDN as DFN-like

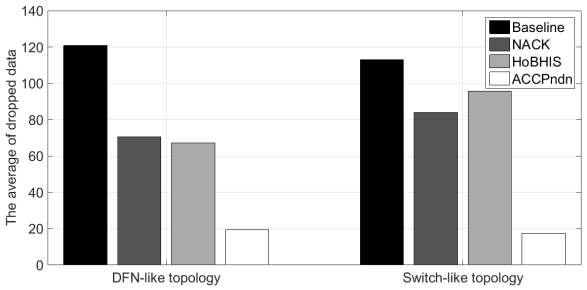


Figure 16: The total average packet drop rate in both considered topologies

and Switch-like (Fig. 7). In particular, we were interested in decomposing the Switch-like network topology graph to two clusters for two reasons: (1) avoiding poor trainability and high generalization error by TLFN over larger and more complex topology graphs, and (2) evaluating feasibility and accuracy of the ACCPndn over two interconnected clusters forming a larger topology graph. There will be several future works, such as applying ACCPndn over larger, more complex and arbitrary network topologies by several separated clusters to assess and measure the robustness and the feasibility of the ACCPndn in more than two interconnected NDN clusters. Another future work for scalability measurement is directly towards investigating the combination of different status of the routers and links (e.g., link failure, connecting and disconnecting nodes to the network, changing the topology of the clusters or the entire graph) in a dynamic network environment.
On the other hand, the implementation of ACCPndn needs more time and effort than two applied preexisting algorithms. Because ACCPndn works in two phases (i.e., offline training and online avoidance), in which the first phase needs more effort for configuring the network parameters as well as evaluating the effect of tuning parameters on performance of the training. Putting more effort into finding well-tuned parameters (e.g., sliding windows for TLFN and logic-based rules for fuzzy control system) leads to forecast congestion traffic with an appropriate performance. In spite of being more time consuming to build TLFN in the first phase of the ACCPndn, we enabled a suitable compromise by ACCPndn between overhead of implementation and the applied performance metrics including the rate of data packet drop and the number of arrival data packet during congestion (see section 10.3). The results confirm that the ACCPndn for congestion control in NDN can yield high accuracy as compared to NACK and HoBHIS without very much computational cost.

Finally, in case of choosing preliminary appropriate sliding windows for constructing TLFN, we applied empirically different sliding windows in order to find the most appropriate sliding windows. Accordingly, we employed the best configuration setting in the first phase of the ACCPndn (see section 8.1).

12. Conclusion

Our main contribution is to develop an Adaptive Congestion Control Protocol in Named Data Networking (ACCPndn) that works in two phases. The first phase - adaptive training- forecasts the source of congestion together with the amount of congestion in NDN routers with a Time-Lagged Feedforward Network (TLFN) optimized by hybridization of PSO and GA. The second phase -fuzzy avoidance- employs a non-linear fuzzy logic-based control system based on the outcomes of first phase, which it makes a proactive decision in each router per interface to control and/or prevent packet drop well enough in advance. Extensive simulations and results show that ACCPndn sufficiently satisfies the performance metrics and outperforms two previous proposals such as NACK and HoBHIS in terms of the minimal packet drop and high-utilization (retrying alternative paths) in bottleneck links to mitigate congestion traffics. In addition, it is found to be scalable with respect to varying bandwidths, packet generation, and replacement policies in cache and PIT table.

13. Acknowledgments

This work was partially supported by projects TIN2013-47272-C2-2 and SGR-2014-881.

References

[1] Jacobson, V., Smetters, D.K., Thornton, J.D., Plass, M.F., Briggs, N.H., Braynard, R.L.. Networking named content. In: Proceedings of the 5th international conference on Emerging networking experiments and technologies. New York, NY, USA: ACM; 2009,.

[2] Ahlgren, B., Dannewitz, C., Imbrenda, C., Kutscher, D., Ohlman, B.. A survey of information-centric networking. Communications Magazine, IEEE 2012;50(7):26 – 36.

[3] Dannewitz, C., Kutscher, D., Ohlman, B., Farrell, S., Ahlgren, B., Karl, H.. Network of information (netinf) - an information-centric networking. Computer Communications 2013;36(7):721 – 735.

[4] Conti, M., Gasti, P., Teoli, M.. A lightweight mechanism for detection of cache pollution attacks in named data networking. Computer Networks 2013;57(16):3178 – 3191.

[5] Lee, H., Nakao, A.. User-assisted in-network caching in information-centric networking. Computer Networks 2013;57(16):3142 – 3153.

[6] Rossini, G., Rossi, D.. Evaluating ccn multi-path interest forwarding strategies. Computer Communications 2013;36(7):771 – 778.

[7] Li, C., Liu, W., Okamura, K.. A greedy ant colony forwarding algorithm for named data networking. In: Proceedings of the Asia-Pacific Advanced Network; vol. 34. 2012, p. 17 – 26.

[8] Jiang, X., Bi, J.. Technical report: Named content delivery network. Tech. Rep.; 2013.

[9] Rozhnova, N., Fdida, S.. An effective hop-by-hop interest shaping mechanism for ccn communications. In: IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). 2012, p. 322 – 327.

[10] Saino, L., Cocora, C., Pavlou, G.. Cctcp: A scalable receiver-driven congestion control protocol for content centric networking. In: Communications (ICC), 2013 IEEE International Conference on. 2013, p. 3775 – 3780.

[11] Karami, A., Guerrero-Zapata, M.. An anfis-based cache replacement method for mitigating cache pollution attacks in named data networking. Computer Networks 2015;80:51–65.

[12] Qian, X., Jing, Y., Tian, J.. Network congestion avoidance strategy with particle filter. Computer Communications 2008;31(9):1723 – 1726.

[13] Lestas, M., Pitsillides, A., Ioannou, P., Hadjipollas, G.. Adaptive congestion protocol: A congestion control protocol with learning capability. Computer Networks 2007;51(13):3773 – 3798.

[14] Xu, Q., Sun, J.. A simple active queue management based on the prediction of the packet arrival rate. Journal of Network and Computer Applications 2014;42:12 – 20.

[15] Li, F., Sun, J., Zukerman, M., Liu, Z., Xu, Q., Chan, S., et al. A comparative simulation study of tcp/aqm systems for evaluating the potential of neuron-based {AQM} schemes. Journal of Network and Computer Applications 2014;41:274 – 299.

[16] Fu, T., Li, Y., Lin, T., Tan, H., Tang, H., Ci, S.. An effective congestion control scheme in content-centric networking. In: Parallel and Distributed Computing, Applications and Technologies (PDCAT), 2012 13th International Conference on. 2012, p. 245 – 248.

[17] Saucez, D., Grieco, L.A., Barakat, C.. Aimd and ccn: Past and novel acronyms working together in the future internet. In: Proceedings of the ACM Workshop on Capacity Sharing. CSWS '12; New York, NY, USA; 2012, p. 21 – 26.

[18] G. Carofiglio M. Gallo, L.M., Papalini, L.. Multipath congestion control in content-centric networks. In: Proceedings of the IEEE INFOCOM NOMEN. 2013,.

[19] Xia, C., Xu, M., Wang, Y.. A loss-based tcp design in icn. In: Wireless and Optical Communication Conference (WOCC). 2013, p. 449 – 454.

[20] Yi, C., Afanasyev, A., Moiseenko, I., Wang, L., Zhang, B., Zhang, L.. A case for stateful forwarding plane. Computer Communications 2013;36(7):779 – 791.

[21] Barabas, M., Boanea, G., Rus, A.B., Dobrota, V., Domingo-Pascual, J.. Evaluation of network traffic prediction based on neural networks with multi-task learning and multiresolution decomposition. In: IEEE International Conference on Intelligent Computer Communication and Processing (ICCP). 2011, p. 95 – 102.

[22] Bonald, T., Martin, M., Bolot, J.C.. Analytic evaluation of red performance. In: Proceedings in Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies; vol. 3. 2000, p. 1415 – 1424.

[23] Cortez, P., Rio, M., Rocha, M., Sousa, P.. Multi-scale internet traffic forecasting using neural networks and time series methods. Expert Systems 2012;29:143 – 155.

[24] Dharmadhikari, V.B., Gavade, J.D.. An nn approach for mpeg video traffic prediction. In: 2nd International Conference on Software Technology and Engineering (ICSTE); vol. 1. 2010, p. 57 – 61.

[25] Li, Z., Lei, Q., Kouying, X., Xinyan, Z.. A novel bp neural network model for traffic prediction of next generation network. In: Fifth International Conference on Natural Computation (ICNC'09); vol. 1. 2009, p. 32 – 38.

[26] Alarcon-Aquino, V., Barria, J.. Multiresolution fir neural-network-based learning algorithm applied to network traffic prediction. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 2006;36(2):208 – 220.

[27] Makridakis, S.G., Wheelwright, S.C., Hyndman, R.J.. Forecasting: Methods and Applications. New York, USA: Wiley; 1997.

[28] Claveria, O., Torra, S.. Forecasting tourism demand to catalonia: Neural networks vs. time series models. Economic Modelling 2014;36:220 – 228.

[29] Cortez, P., Rio, M., Rocha, M., Sousa, P.. Internet traffic forecasting using neural networks. In: International Joint Conference on Neural Networks (IJCNN'06). 2006, p. 4942 – 4949.

[30] Peralta Donate, J., Cortez, P., Sanchis de Miguel, A., Gutierrez Sanchez, G.. Evolving time-lagged feedforward neural networks for time series forecasting. In: Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation. GECCO '11; New York, NY, USA: ACM; 2011, p. 163 – 164.

[31] Kourentzes, N., Barrow, D.K., Crone, S.F.. Neural network ensemble operators for time series forecasting. Expert Systems with Applications 2014;-(-):–.

[32] Ren, C., An, N., Wang, J., Li, L., Hu, B., Shang, D.. Optimal parameters selection for {BP} neural network based on particle swarm optimization: A case study of wind speed forecasting. Knowledge-Based Systems 2014;56:226 – 239.

[33] Khashei, M., Hejazi, S.R., Bijari, M.. A new hybrid artificial neural networks and fuzzy regression model for time series forecasting. Fuzzy Sets and Systems 2008;159(7):769 – 786.

[34] Das, G., Pattnaik, P.K., Padhy, S.K.. Artificial neural network trained by particle swarm optimization for non-linear channel equalization. Expert Systems with Applications 2014;41(7):3491 – 3496.

[35] Wong, W., Xia, M., Chu, W.. Adaptive neural network model for time-series forecasting. European Journal of Operational Research 2010;207(2):807 – 816.

[36] Yogi, S., Subhashini, K.R., Satapathy, J.. A pso based functional link artificial neural network training algorithm for equalization of digital communication channels. In: Industrial and Information Systems (ICIIS), 2010 International Conference on. 2010, p. 107 – 112.

[37] Malviya, R., Pratihar, D.K.. Tuning of neural networks using particle swarm optimization to model {MIG} welding process. Swarm and Evolutionary Computation 2011;1(4):223 – 235.

[38] Lee, C.H., Lee, Y.C.. Nonlinear systems design by a novel fuzzy neural system via hybridization of electromagnetism-like mechanism and particle swarm optimisation algorithms. Information Sciences 2012;186(1):59 – 72.

[39] Chrysostomou, C., Pitsillides, A., Hadjipollas, G., Sekercioglu, A.. Fuzzy logic congestion control in tcp/ip best-effort networks. In: Australian Telecommunications Networks and Applications Conference (ATNAC). Melbourne, Australia; 2003, p. 8 – 10.

[40] Chrysostomou, C., Pitsillides, A., Sekercioglu, Y.. Fuzzy explicit marking: A unified congestion controller for best-effort and diff-serv networks. Computer Networks 2009;53(5):650 – 667.

[41] Afanasyev, A., Mahadevan, P., Moiseenko, I., Uzun, E.,

Zhang, L.. Interest flooding attack and countermeasures in named data networking. In: IFIP Networking Conference, 2013. 2013, p. 1 – 9.

[42] Ran, J., Lv, N., Zhang, D., Ma, Y., Xie, Z.. On performance of cache policies in named data networking. In: International Conference on Advanced Science and Electronics Information (ICACSEI). Atlantis Press; 2013, p. 668 – 671.

[43] Karami, A., Guerrero-Zapata, M.. A fuzzy anomaly detection system based on hybrid pso-kmeans algorithm in content-centric networks. Neurocomputing 2015;149(Part C):1253–1269.

[44] Karami, A., Guerrero-Zapata, M.. A hybrid multiobjective rbf-pso method for mitigating dos attacks in named data networking. Neurocomputing 2015;151(Part 3):1262–1282.

[45] Carofiglio, G., Gallo, M., Muscariello, L.. performance of bandwidth and storage sharing in information-centric networks. Computer Networks 2013;.

[46] Frank, R.J., Davey, N., Hunt, S.P.. Time series prediction and neural networks. Journal of Intelligent and Robotic Systems 2001;31(1-3):91 – 103.

[47] Gluszek, A., Kekez, M., Rudzinski, F.. Web traffic prediction with artificial neural networks. In: Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments III; vol. 5775 of *Society of Photo-Optical Instrumentation Engineers (SPIE)*. 2005, p. 520 – 525.

[48] Jain, A., Kumar, A.M.. Hybrid neural network models for hydrologic time series forecasting. Applied Soft Computing 2007;7(2):585 – 592.

[49] da S. Gomes, G.S., Ludermir, T.B.. Optimization of the weights and asymmetric activation function family of neural network for time series forecasting. Expert Systems with Applications 2013;40(16):6438 – 6446.

[50] Zhao, Y., Weng, T., Small, M.. Response of the parameters of a neural network to pseudoperiodic time series. Physica D: Nonlinear Phenomena 2014;268:79 – 90.

[51] Kennedy, J., Eberhart, R.. Particle swarm optimization. In: Proceedings in IEEE International Conference Neural Networks; vol. 4. 1995, p. 1942 – 1948.

[52] Urade, H.S., Patel, R.. Dynamic particle swarm optimization to solve multi-objective optimization problem. Procedia Technology 2012;6:283 – 290.

[53] Shakibian, H., Charkari, N.M.. In-cluster vector evaluated particle swarm optimization for distributed regression in {WSNs}. Journal of Network and Computer Applications 2014;42:80 – 91.

[54] Kuo, R.J., Syu, Y.J., Chen, Z.Y., Tien, F.C.. Integration of particle swarm optimization and genetic algorithm for dynamic clustering. Informaton Sciences 2012;195:124 – 140.

[55] Eberhart, R.C., Shi, Y.. Comparing inertia weights and constriction factors in particle swarm optimization. In: Proceedings of the Evolutionary Computation; vol. 1. 2000, p. 84 – 88.

[56] Kuo, R., Lin, L.. Application of a hybrid of genetic algorithm and particle swarm optimization algorithm for order clustering. Decision Support Systems 2010;49(4):451 – 462.

[57] Chamkalani, A., Mae'soumi, A., Sameni, A.. An intelligent approach for optimal prediction of gas deviation factor using particle swarm optimization and genetic algorithm. Journal of Natural Gas Science and Engineering 2013;14:132 – 143.

[58] Moradi, M., Abedini, M.. A combination of genetic algorithm and particle swarm optimization for optimal {DG} location and sizing in distribution systems. International Journal of Electrical Power & Energy Systems 2012;34(1):66 – 74.

[59] Cho, H.C., Fadali, S.M., Lee, H.. Adaptive neural queue management for tcp networks. Computers and Electrical Engineering 2008;34:447 – 469.

[60] Kuo, R., Hung, S., Cheng, W.. Application of an optimization artificial immune network and particle swarm optimization-based fuzzy neural network to an rfid-based positioning system. Information Sciences 2014;262:78 – 98.

[61] Ruan, G., Tan, Y.. A three-layer back-propagation neural network for spam detection using artificial immune concentration. Soft Computing 2009;14(2):139 – 150.

[62] Kirubavathi Venkatesh, G., Anitha Nadarajan, R.. Http botnet detection using adaptive learning rate multilayer feed-forward neural network. In: International Conference on Information Security Theory and Practice: Security, Privacy and Trust in Computing Systems and Ambient Intelligent Ecosystems. Springer-Verlag; 2012, p. 38 – 48.

[63] Gomathy, A., Lakshmipathi, B.. Network intrusion detection using genetic algorithm and neural network. In: Advances in Computing and Information Technology Communications in Computer and Information Science; vol. 198. Springer-Verlag; 2011, p. 399 – 408.

[64] Barabas, M., Boanea, G., Dobrota, V.. Multipath routing management using neural networks-based traffic prediction. In: Proceedings of the 3rd International Conference on Emerging Network Intelligence. IARIA; 2011, p. 118 – 124.

[65] Karami, A., Guerrero-Zapata, M.. Mining and visualizing uncertain data objects and named data networking traffics by fuzzy self-organizing map. In: In Proceedings of the Second International Workshop on Artificial Intelligence and Cognition (AIC); vol. 1315. 2014, p. 156–163.

[66] Shivakumar, U., Ravi, V., Gangadharan, G.. Ranking cloud services using fuzzy multi-attribute decision making. In: IEEE International Conference on Fuzzy Systems (FUZZ). 2013, p. 1–8.

[67] Zadeh, L.A.. Fuzzy sets. Information Control 1965;8:338 – 353.

[68] Baig, Z., Khan, S.. Fuzzy logic-based decision making for detecting distributed node exhaustion attacks in wireless sensor networks. In: Second International Conference on Future Networks (ICFN '10). 2010, p. 185 – 189.

[69] Steiner, S.H.. Exponentially weighted moving average control charts with time varying control limits and fast initial response. Journal of Quality Technology 1999;31.

[70] Afanasyev, A., Moiseenko, I., Zhang, L.. ndnSIM: NDN simulator for NS-3. Technical Report NDN-0005; NDN; 2012. URL `named-data.net/techreports.html`.

[71] Dfn-verein: Dfn-noc. retrieved Jun. 2013. URL `dfn.de/dienstleistungen/dfninternet/noc`.

[72] Heckmann, O.. The competitive Internet service provider: network architecture, interconnection, traffic engineering and network design. Wiley series in communications networking & distributed systems; J. Wiley; 2006. URL `books.google.es/books?id=DyEfAQAAIAAJ`.

[73] Ding, X., Canu, S., Denoeux, T.. Neural network based models for forecasting. In: In proceedings of Applied Decision Technologies Conference (ADT'95). Uxbridge, UK; 1995, p. 243 – 252.

[74] Karami, A., Johansson, R.. Utilization of multi attribute decision making techniques to integrate automatic and manual ranking of options. Journal of Information Science and Engineering 2014;30(2):519–534.

[75] Funahashi, K.I.. On the approximate realization of continuous mappings by neural networks. Neural Networks 1989;2(3):183 – 192.

[76] Hornik, K., Stinchcombe, M., White, H.. Multilayer feedforward networks are universal approximators. Neural Networks 1989;2(5):359 – 366.

[77] Jin, Y.. Multi-Objective Machine Learning. Studies in Computational Intelligence; Springer; 2006. ISBN 9783540330196.