CogProt: A Framework for Cognitive Configuration and Optimization of Communication Protocols

Dzmitry Kliazovich[1], Neumar Malheiros[2], Nelson L.S. da Fonseca[2], Fabrizio Granelli[1], and Edmundo Madeira[2]

[1] DISI – University of Trento
Via Sommarive 14, I-38050 Trento, Italy
{kliazovich,granelli}@disi.unitn.it

[2] Institute of Computing – University of Campinas
Av. A. Einstein, 1251, 13083-970 Campinas, Brazil
{ncm,nfonseca,edmundo}@ic.unicamp.br

Abstract. Advancements in network technologies dramatically increased management complexity. Cognitive networking was introduced to deal with this problem, by providing algorithms for autonomous network management and protocol reconfiguration. In this paper, we propose a framework for cognitive configuration and optimization of communication protocols called CogProt. CogProt is a distributed framework that allows dynamic reconfiguration of operational protocol stack parameters for optimizing protocol performance under changing network conditions. As a proof of concept, the framework is illustrated for the cognitive configuration of TCP congestion window evolution. In this setup, the TCP window increase factor is adjusted in runtime based on the TCP goodput experienced in the immediate past. Simulation results demonstrate that the proposed cognitive framework is able to improve average TCP performance under changing network conditions.[1]

Keywords: cognitive networks, self-configuration, cognitive TCP.

1 Introduction

Degradation of performance due to time-varying network conditions is a challenging issue that needs to be properly addressed by current network research. The dynamic adjustment of the protocol stack parameters in a cognitive fashion is a promising approach to deal with that issue. Such idea was introduced by Mitola [6], along with the concept of cognitive radio to provide efficient spectrum sharing by avoiding interference among communication systems.

Actually, cognitive radio is limited to the ability to tune the parameters of physical and link layers while following local optimization goals. On the other

[1] This work is supported by the Italian Ministry in the framework of WORLD (Wireless multiplatfOrm mimo active access netwoRks for QoS-demanding muLtimedia Delivery) project and by FAPESP under grant number 2006/50512-4.

hand, the broader concept of cognitive networks [8,3] considers system-wide goals and cross-layer design[7,9]. Cognitive network is a recently emerged networking paradigm that combines cognitive algorithms, cooperative networking, and cross-layer design in order to provide real-time optimization of complex communication systems.

The main contribution of this paper is a framework for cognitive configuration and optimization of communication protocols called CogProt. CogProt involves main concepts of the cognitive network approach, such as global vision and optimization of end-to-end goals. It is suitable for joint optimization of multiple protocol layers for different network nodes in a distributed way. CogProt is not only concerned with initial setup of protocol parameters, but also with their runtime reconfiguration and optimization.

The CogProt framework is not limited to finding optimal values for each protocol parameter, but it aims the optimization of network performance by periodically reconfiguring each protocol parameter based on recent (even immediate) experience. Each network node is allowed to decide on the best protocol configuration to fit current network conditions. This way, the CogProt framework is not only able to avoid performance degradation due to time-varying conditions, but also to maximize overall performance for different network scenarios.

In this work, CogProt is applied to the TCP/IP stack as an illustration of its use. Despite its success, the TCP/IP Internet protocol suite still presents fundamental design challenges since its strict design is not able to guarantee the required performance[4,1]. The TCP/IP protocol suite, historically designed for wired networks, neither includes adaptation techniques for heterogeneous and dynamic network environments nor allows communication between layers, which leads to severe limitations in terms of performance. As a proof of concept, we have implemented the CogProt framework into the Network Simulator (ns-2) in order to provide cognitive configuration and optimization of TCP congestion window evolution. Simulation results show performance improvement on TCP average throughput, which demonstrates the feasibility and efficiency of the proposed approach.

Section 2 presents a motivation example, while Section 3 describes the proposed framework. Section 4 discusses on tunable parameters at different protocol stack layers. Section 5 presents performance evaluation of the proposed approach. We describe the simulation scenario and present results considering dynamic adaptation of TCP congestion window increase factor. Finally, Section 6 presents some conclusions and future directions.

2 Motivating Example

The configuration of communication protocols defines the overall performance of applications and data transfer services. However, it is not always possible to anticipate the best setup due to the unpredictable state of the network. For example, the choice for the default initial value of TCP congestion window (w) changed over the time. The original TCP specification defined the initial w value equal to 1 segment. Later, due to a bug in NetBSD implementation, the initial w value was increased to 2 segments. Then, the specification was adapted to allow the initial w value as either 1 or 2 segments. The most recent specification from 2002 [2] permits upper bound for initial w of roughly 4K, which is equivalent to 3 TCP segments. Actually, the authors in [2] proposed setting the initial w value as large as 4 segments in order to improve TCP startup performance.

Advantages of large initial window values include better performance (with TCP receiver implementing Delayed-ACK option) and low protocol delay for tiny-flow applications such as SMTP- and HTTP-based. In addition, large w values enable fast retransmit and fast recovery algorithms right from the beginning of the TCP flow. However, disadvantages of large w values are present in highly congested environments. Although even in such environments, individual flow throughput reduction is rarely observed. The main danger comes from the possibility of congestion collapse of entire network.

In summary, large w values bring performance increase and are desirable in high bandwidth-delay network with low or moderate congestion levels, and should be avoided otherwise. However, there is no way for a network node to know in advance the available network bandwidth and the level of congestion at the end-to-end path between the sender and receiver. Therefore, it is not possible to define the best default value for the initial window. A cognitive approach would dynamically adjust the initial window value for TCP flows according to current network conditions. In Section 4, this problem will be revisited to underline the benefits of the cognitive approach.

3 CogProt Framework

The proposed cognitive framework, CogProt, aims at supporting dynamic configuration and optimization of communication protocols. It provides a way for network elements to adapt their configuration and protocol stack parameters in order to fit constantly changing network conditions. The process of search for optimal setup of protocol parameters is performed by using cognitive algorithms and by sharing information among network nodes. The proposed approach considers the network node architecture presented in Fig. 1. It introduces a cognitive plane in parallel to the protocol layers. This plane is capable of monitoring protocol stack parameters as well as controlling them by issuing configuration commands.

The cognitive optimization process involves a quality feedback loop. For the duration of the optimization process, the cognitive plane monitors the performance of current protocol parameters setup according to well defined target quality metrics. For example, the quality metric for a physical layer parameter could be the measured data rate, while at the application layer the quality metric could be the end-to-end delay for real-time multimedia applications.

The feedback loop iteration is completed with the enforcement of reconfiguration actions from the cognitive plane. These reconfiguration actions are the result of decision-making procedures performed by the cognitive framework. According to the scope they operate, such decision-making procedures can be classified

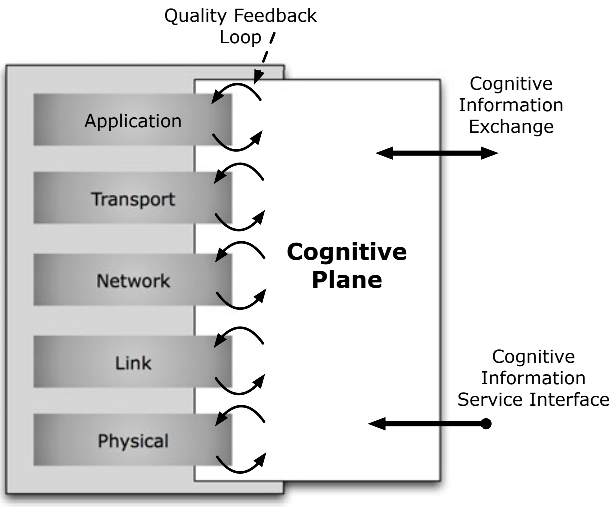

Fig. 1. Cognitive node architecture

into: i) local, which form their decisions based only on locally available information, ii) distributed, receiving additional information from neighboring network nodes, or iii) service-based, requiring reconfiguration policies from a Cognitive Information Service (CIS) fostering cognitive signaling in the local network.

The main task of the cognitive plane is the adaptation of different protocol stack parameters in order to converge to an optimal operational point given the network state. The proposed framework adopts conservative cognitive adaptation to minimize the changes in the protocol stack operation and yet promote convergence to the desired operational point. In the proposed approach, each protocol parameter P is expressed in terms of its default value P_{def} and its operation range $[P_{min}, P_{max}]$. The operation of the protocol is initiated with parameter P set to its default value. Then, the cognitive mechanism begins searching for optimal P values. Such adaptation process is divided in three phases: data analysis, decision-making, and action, as illustrated in Fig. 2.

At the end of an interval I, the cognitive mechanism measures and stores the obtained performance from the current value of P in accordance with the defined quality metric. This *data analysis* phase allows the algorithm to build a knowledge base of performance information on the available values for P along protocol operation. Then, in the *decision-making* phase, the mechanism selects the value of P that provides the best performance according to the performance information base. That value is assigned to the mean of a random number generator that follows a normal distribution. Finally, in the *action* phase, a new value for P is chosen in the range $[P_{min}, P_{max}]$ from the random number generator.

The initial mean for the number generator is P_{def}. This loop continuously adjusts the mean of the normal distribution to the value of P that provides the best performance under current network conditions. Thus, the mean converges

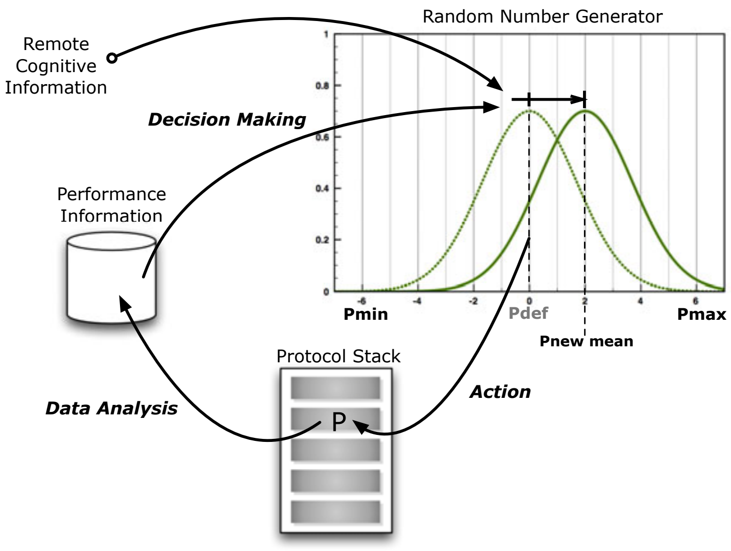

Fig. 2. The quality feedback loop of the cognitive adaptation mechanism

to the optimal value for P. As a result, that optimal value is chosen with a higher frequency since it is the mean of the normal distribution. Meanwhile, the mechanism will choose values nearby the mean, which allow it to react to changes on the network state. The standard deviation assigned to the normal distribution affects the aggressiveness of the mechanism in trying new values of P at each interval I.

CogProt builds a history of operation with different settings and can reconfigure the parameter or adjust future setups based on its past experience. In order to illustrate the operation of the cognitive framework, let us discuss again the problem of setting the initial value of the TCP window size (w) as mentioned in Section 2. The state of end-to-end paths cannot be known "a priori". Therefore, there is a need to properly adjust the initial window to avoid performance degradation. The proposed cognitive mechanism solves that problem by using the described quality feedback loop to constantly trying different values for w in a "smart" way, while monitoring the corresponding protocol performance metrics. CogProt sets most of the flows initiated by the node to use the default value for w, since the default value is the initial mean of the normal distribution. However, from time to time it will use values nearby the mean and update the performance information base. Then, CogProt adjusts the mean of the normal distribution to follow optimal performance.

According to proposed approach, the default setup (P_{def}) of the initial window size for TCP flows may be chosen equal to 4K bytes, P_{min} may correspond to 1

MSS (Maximum Segment Size of TCP), while P_{max} may be chosen to be equal to 10 MSSs. This way, by using the proposed cognitive adaptation algorithm, most of the outgoing flows will start with congestion window of 4K, which should deliver optimal performance in normal network conditions. In case of unfavorable conditions, like network congestion, a lower value for w will provide the best performance. On the other hand, in case of unloaded high bandwidth-delay links, a higher value for w will provide the best performance. In both cases, as soon as the algorithm finds the "new" optimal value in the performance table, the mean of the normal distribution is adjusted accordingly. In this way, the mean is dynamically adjusted in order to optimize the protocol performance.

4 Cognitive Tuning of TCP/IP Parameters

The TCP/IP protocol reference model is the de-facto standard for communication on the Internet. It contains a large variety of protocols whose parameters need to be properly configured. Such need has motivated the definition of some protocol variants, for example TCP variants for specific network scenarios. Figure 3 presents a snapshot of the most widely used protocols in the TCP/IP reference model outlining their main configuration parameters and corresponding performance metrics.

The Application layer provides the environment for running user applications. Configurable parameters and quality metrics at this layer depend on the nature of applications. For File Transfer (FTP) applications, a configurable parameter

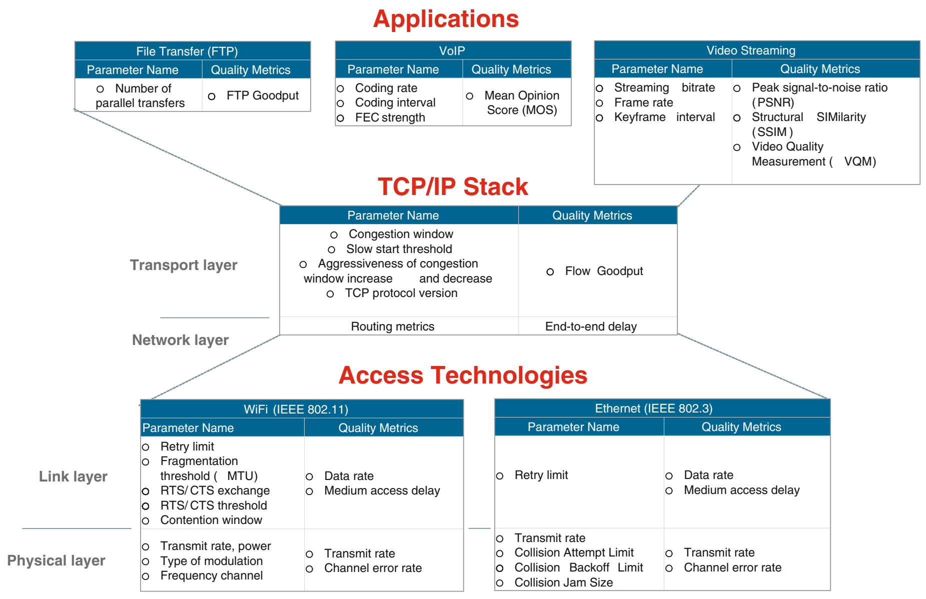

Fig. 3. Tunable TCP/IP Protocol Stack Parameters

could be the number of parallel connections and the main quality metric is the file transfer goodput. For Voice over IP (VoIP) applications, controlling coding rate, coding interval, and Forward Error Correction (FEC) strength helps to increase the voice quality commonly expressed using the Mean Opinion Score (MOS) metric [5].

For video streaming applications, streaming bitrate, framerate, and keyframe interval determine the quality of video flow perception. High bitrates allow video transmissions with high resolutions, high frame rates improve perception of video samples involving high motions, while shorter keyframe intervals improve decoding capabilities in the presence of frame losses or transmission errors.

The Transport layer is generally represented by Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) protocols. While UDP is mostly passive with the main task of providing differentiation of IP datagrams between different port numbers, TCP implements complex mechanisms including reliable transmissions and flow control. The TCP congestion window evolution is the key factor affecting the flow throughput. Controlling the congestion window increase factor (α) and decrease factor (β) parameters allows improving the flow performance by adjusting the tradeoff between network utilization, protocol fairness, and the level of network congestion.

The Network layer is responsible for routing packets across interconnected networks. Hence, the main performance metric can correspond to the quality of the selected route, for instance in terms of the number of hops or end-to-end delay.

The Link layer serves network layer requests and controls physical layer for different access technologies. Most of the controllable parameters in this layer are determined by the communication technology in use. In Career Sense Multiple Access (CSMA) protocols, such as WiFi IEEE 802.11 or Ethernet IEEE 802.3, the main tunable parameters correspond to the size and evolution of the contention window, as well as the retry limit, which corresponds to the maximum number of retransmission attempts taken at the link layer before a frame is discarded.

The Physical layer parameters are defined by the used network access technology. Wireless access technologies can provide control over the power level of the transmitted signal, choice of the type of modulation, and frequency channel. Fixed network technologies like Ethernet are usually described by collision detection capabilities. Physical layer performance can be defined in terms of the data rate achievable at the physical layer, as well as Bit Error Rate (BER) achieved for the transmitted bit stream.

Indeed, a lot of parameters in the TCP/IP stack need proper tuning. Some of them have a great influence on applications performance. For instance, the parameters governing the TCP transmission window and the initial value of the window have great influence on the performance, especially in networks with high bandwidth delay product.

5 Performance Evaluation

In order to present performance benefits of the proposed approach, we extended the Network Simulator (ns-2) with the required CogProt functionalities to provide dynamic reconfiguration of the TCP congestion window. In particular, the parameter of interest in this evaluation is the TCP congestion window increase factor (α), which controls the evolution of TCP window and, thus, its throughput performance.

5.1 Scenario Description

The simulated topology is illustrated in Fig. 4. It consists of a single source node S and two destination nodes D_1 and D_2 connected using two routers R_1 and R_2. These routers form a bottleneck link of 2 Mbps with the propagation delay of 1 ms. The CogProt framework is implemented only at the source S, while destination nodes and network routers left unmodified.

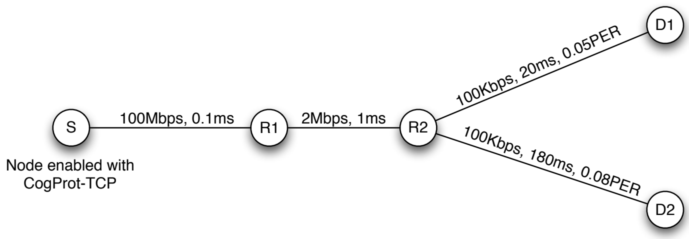

Fig. 4. Simulated topology

Such simulation topology allows establishment of TCP connections that require different window increase strategies depending on the error rate and delay of the last mile link connecting the corresponding destination node. The TCP connection $S - D_1$ has smaller Packet Error Rate (PER) of 5% and delay of 20 ms, while the connection $S - D_2$ experiences higher PER of 8% and larger delay of 180 ms.

Different values of link PERs and propagation delays will require from the flows different window increase strategies in order to obtain optimal performance. High values of α are expected to bring better throughput for high PERs. However, in case of no errors, high values of α will lead to multiple congestion-related losses and throughput degradation due to multiple retransmissions.

In our simulations, we compared the performance of standard TCP NewReno protocol using fixed values for the parameter α with the dynamic reconfiguration of α performed by the CogProt approach.

Average TCP flow throughput is chosen as the main quality metric for the feedback loop. In our simulations, CogProt was allowed to vary the value of the parameter α in the interval $[1, 5]$. A new value for α is selected at the beginning of an interval I equal to 1 second with the standard deviation for the normal distribution (of the random generator) configured at 0.5. The average performance is computed by using an exponentially weighted moving average as follows:

$$T_a = T_a * (1 - s) + T_m * (s) \tag{1}$$

where s is the weight assigned to the immediate throughput (T_m) measured for the current value of α and T_a is the average throughput for the current value of α. The weight of 0.5 was found to provide good tradeoff between the significance of past and present values affecting the speed of algorithms convergence.

5.2 Simulation Results

The CogProt performance was evaluated in three different experiments involving the following TCP flows: $S - D_1$, $S - D_2$, and both $S - D_1$ and $S - D_2$. In the latter case, with two flows, the flow $S - D_1$ was active the first half of the simulation time, while the flow $S - D_2$ during the second half. For all scenarios, the simulation time was 1000 seconds, and the results were the average from 10 runs with a 95% confidence interval.

Fig. 5 shows the average TCP flow throughput for the three experiments with fixed values of α compared with the proposed cognitive mechanism CogProt. For the flow $S - D_1$, experiencing shorter RTT and low error rate, lower values of parameter α lead to optimal performance, while in the other case with flow $S - D_2$ experiencing high error rate and larger RTT, high values of α become desirable. This confirms our assumption that, for any fixed value of α assigned, the flow is going to suffer from performance degradation under specific conditions. However, when CogProt is active (right set of bars in Fig. 5), it is able to keep performance close to optimal for the scenarios with individual flows and outperforms any fixed values of α in the combined scenario. This confirms the benefit of CogProt dynamic adaptation capabilities enabling it to avoid performance degradation in varying network conditions.

Fig. 6 presents the average throughput over time for the experiment with two TCP flows: $S - D_1$ active in the first half of the simulation, and $S - D_2$ active in the second half. The parameter α fixed at 1 performs well for the flow $S - D_1$ and not for $S - D_2$. On the other hand, parameter α fixed at 5 performs well for $S - D_2$ but not for $S - D_1$. This way, performance degradation cannot be avoided by using any fixed value of α. With CogProt the measured TCP throughput corresponds to the best average performance.

In this scenario with both flows, during the first half of the simulation CogProt selected the value of 1 for the parameter α with higher frequency than any other value, while in the second half the value selected with higher frequency is 5. Fig. 7 shows the frequency of selection of different values for α by the proposed algorithm. We can see that the frequency of the value 1 was higher for the flow $S - D_1$ than $S - D_2$, as well as the frequency of the value 5 was higher for the flow $S - D_2$ rather than $S - D_1$.

Results demonstrate the ability of the proposed mechanism to adapt the parameter of interest with a fine granularity. Even considering very short intervals

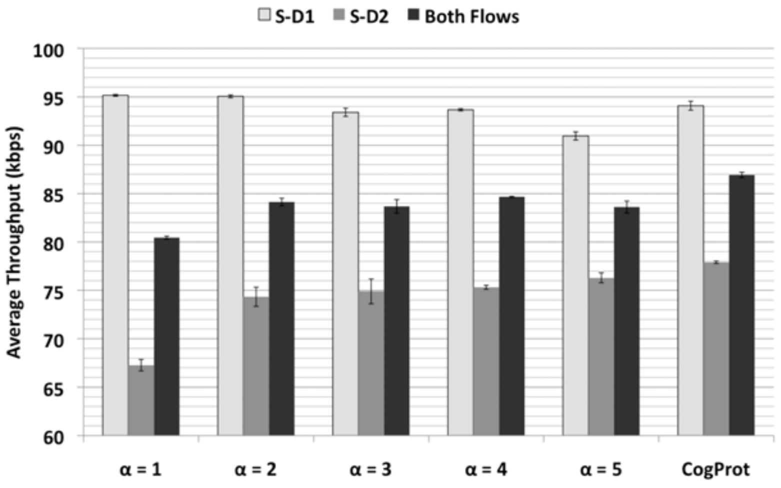

Fig. 5. Average TCP Flow Throughput

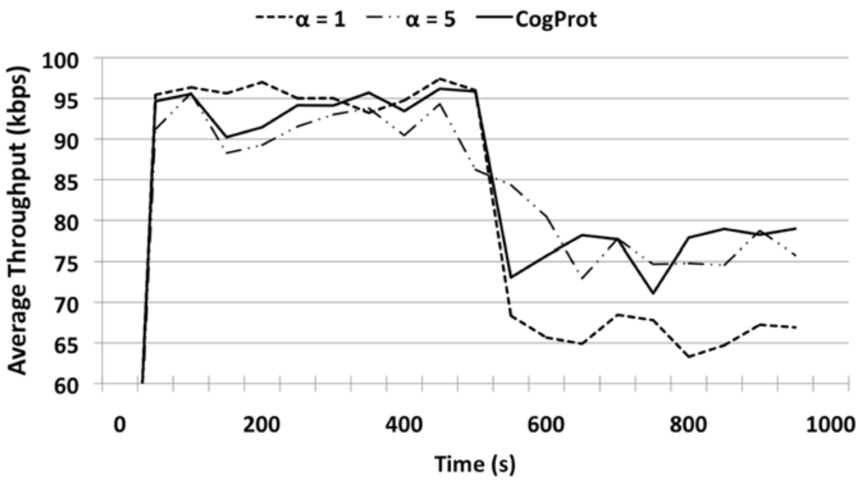

Fig. 6. Average throughput over time in Scenario 3

of reconfiguration (in the order of milliseconds), the mechanism is capable to react to changing network conditions. For example, after a window drop by the congestion mechanism, CogProt can keep high values for α allowing the congestion window to increase fast.

Improving the performance of the TCP protocol is a challenging issue. Results show that CogProt is able to improve average TCP performance by exploring the difference on performance for different values of α and reconfiguring TCP

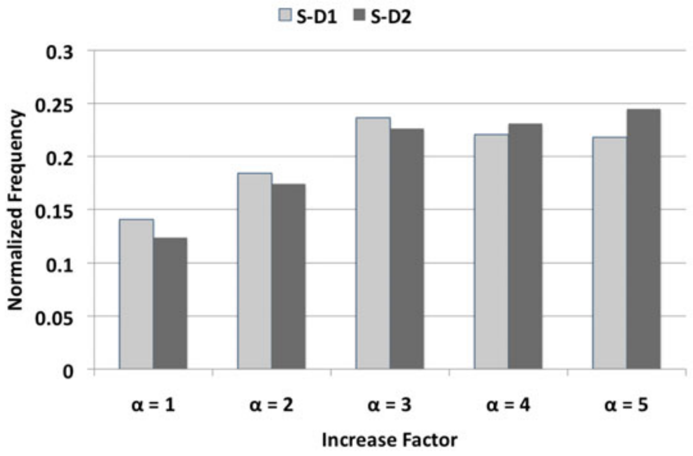

Fig. 7. Frequency of each value of α as chosen by CogProt

behavior accordingly. With any fixed value of α, TCP flows were likely to experience performance degradation for a variety of scenarios. The proposed cognitive mechanism avoids such problem. Moreover, if there are no global optimal values for a protocol parameter, CogProt can provide the best average performance by adapting the protocol to current conditions

6 Conclusion and Future Work

The proposed cognitive framework allows dynamic reconfiguration of main protocol stack parameters at different layers for achieving performance goals driven by target quality metrics. CogProt does not add a high degree of complexity and therefore it can be implemented even in network elements with limited resources. In addition, the proposed approach does not raise compatibility issues. Cognitive nodes can interoperate with ordinary nodes since the cognitive mechanism does not change protocol messages and operation.

The proposed approach was illustrated in the cognitive setting of TCP/IP reference model which brings cognitive reconfiguration into network management and protocol configuration. Performance evaluation studied the application of the proposed approach to cognitive configuration of TCP window evolution. The window increase factor is adjusted during runtime based on the TCP throughput experience achieved in the immediate past. This is a sender side only modification which constrains proper configuration of window increase factor during connection life time. It does not require any changes at the TCP receiver or any other network nodes and it is transparent to other TCP modules.

Future work will be focused on application of the proposed approach to other TCP/IP protocols as well as towards cognitive optimization algorithms distributed among network nodes. In addition, we will work on further development of architecture for the Cognitive Information Service, as well as on signaling mechanisms for cognitive information exchange between network nodes and the service and among the nodes themselves.

References

1. Akyildiz, I.F., Wang, X.: Cross-layer design in wireless mesh networks. IEEE Transactions on Vehicular Technology 57(2), 1061–1076 (2008)
2. Allman, M., Floyd, S., Partridge, C.: Increasing TCP's Initial Window. IETF RFC 3390 (Proposed Standard) (October 2002)
3. Demestichas, P., Stavroulaki, V., Boscovic, D., Lee, A., Strassner, J.: m@angel: autonomic management platform for seamless cognitive connectivity to the mobile internet. IEEE Communications Magazine 44(6), 118–127 (2006)
4. Foukalas, F., Gazis, V., Alonistioti, N.: Cross-layer design proposals for wireless mobile networks: a survey and taxonomy. IEEE Communications Surveys & Tutorials 10(1), 70–85 (2008)
5. ITU-T. Methods for subjective determination of transmission quality. ITU-T Recommendation P.800, International Telecommunication Union (ITU) (1996)
6. Mitola, J.: Cognitive radio for flexible mobile multimedia communications. Mobile Networks and Applications 6(5), 435–441 (2001)
7. Srivastava, V., Motani, M.: Cross-layer design: a survey and the road ahead. IEEE Communications Magazine 43(12), 112–119 (2005)
8. Thomas, R.W., Friend, D.H., DaSilva, L.A., Mackenzie, A.B.: Cognitive networks: adaptation and learning to achieve end-to-end performance objectives. IEEE Communications Magazine 44(12), 51–57 (2006)
9. Winter, R., Schiller, J.H., Nikaein, N., Bonnet, C.: Crosstalk: cross-layer decision support based on global knowledge. IEEE Communications Magazine 44(1), 93–99 (2006)