Simulation-based Retrieval of Adaptation Knowledge

Alexander Frömmgen Patrick Wagner Alejandro Buchmann
DVS, TU Darmstadt, Germany
{froemmgen, wagner, buchmann}@dvs.tu-darmstadt.de

ABSTRACT

Today's networking applications have to operate under changing environmental conditions. Web browsers on mobile devices, for example, are facing rapidly changing network conditions. Even though browsers and network stacks provide fine grained configuration options, it is challenging to find configurations which perform well in these changing environments.

In this paper, we propose to adapt the configuration to changing network conditions at runtime. Based on monitored network properties, a k-nearest neighbour classifier predicts suitable configurations. For a first evaluation, we trained the classifier for a web browsing scenario with Firefox in Mininet simulations. The predicted configurations outperform all static configurations and reduce the average page load time by 1.5 seconds, realizing 68% of the possible improvement of an optimal prediction.

1. INTRODUCTION

Users expect network applications to work under different network conditions. Especially applications on mobile devices with multiple access technologies are facing rapidly changing network conditions regarding bandwidth, latency, and packet drops. Optimizing the application for all possible environments at the same time is challenging. Applications such as modern web browsers provide a multitude of configuration parameters [14], e.g. the maximum number of parallel TCP connections and the usage of HTTP pipelining [6]. However, as these configurations are static, today's applications are statically configured in a highly dynamic environment.

In this paper, we propose to leverage the already existing configuration parameters and adapt the application at runtime to always execute a suitable configuration. Therefore, we predict the optimal configuration out of a huge configuration space based on the currently monitored behaviour of the environment (Figure 1 top). To manage the inherently complex dependencies, we propose to use a classifier for the prediction and automatically train the classifier offline for a wide range of environments. For a first evaluation of our approach, we executed a typical web browsing example with the Firefox browser in 175,500 Mininet [11] simulations with different network conditions to train a k-nearest neighbour classifier (Figure 1 bottom).

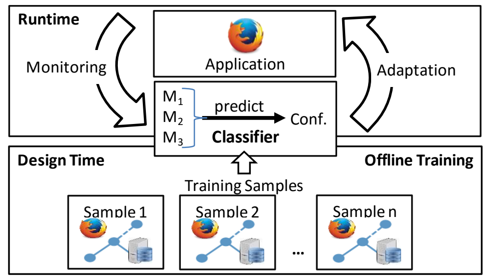


Figure 1: The classifier predicts the optimal application configuration at runtime based on offline retrieved training samples from Mininet simulations.

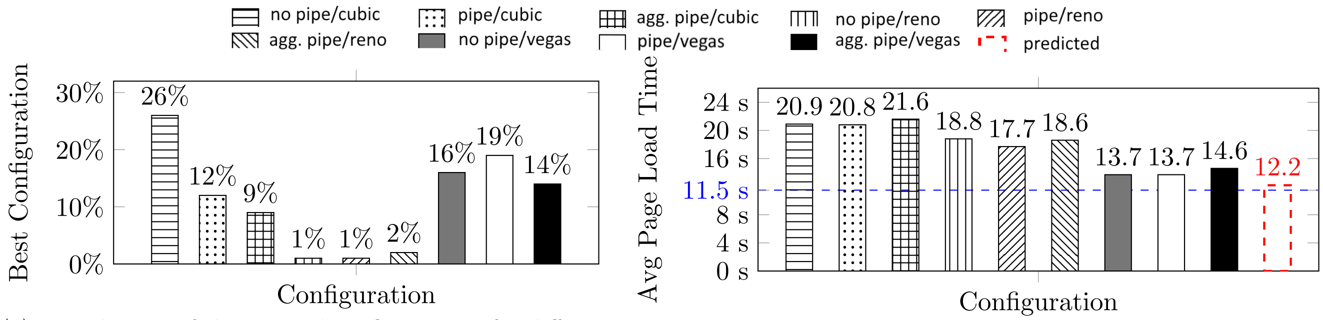
Using simulations to learn desired behaviour offline was successfully applied by Winstein et al. [20], who automatically generated congestion controls which outperform analytically derived algorithms. Our presented approach follows the *MAPE-K* cycle of autonomous computing [15] for managing adaptive behaviour. Compared with existing approaches, such as the *Fossa* [7, 8] framework, our approach uses configuration parameters of real world applications and is not restricted to a rule based representation of the adaptation logic.

2. PREDICTION ENGINE

The prediction engine maps a set of monitored attributes $\langle mon\_attr_1, \ldots, mon\_attr_n \rangle$ to the target configuration. The monitoring attributes should be chosen carefully, as they are essential for a high classification performance. For applications which use TCP, connection details such as the round trip time and the experienced packet drops are obvious candidates. They are easily accessible from the socket state [16].

CoNEXT Student Workshop'15, December 01 2015, Heidelberg, Germany

DOI: 10.1145/2842665.2843564



(a) Distribution of the optimal configurations for different network conditions in the Mininet simulations.

(b) Average page load time of the configurations for the tested web pages. The average over all optimal configurations is 11.5s.

Figure 2: Evaluation of the prediction engine with 175,500 simulations.

Regarding the choice of the classifier, first evaluations showed that a k-nearest neighbour classifier outperforms both tree based classifiers and support vector machines. The k-nearest neighbour classifier uses a distance function to choose the target configuration based on the majority of the k nearest trained samples [18].

3. TRAIN THE ENGINE

The classifier uses training samples with a set of attributes and their target label for training. In the web browser example, the classifier minimizes the time from starting the HTTP request until the document and all dependent resources are loaded. Thus, the label is the configuration which leads to the lowest page load time regarding the *Navigation Timing* interface [9].

As the classifier requires a huge amount of training data, we automatically generate these in Mininet simulations with different network conditions. Mininet allows to run unmodified binaries and supports all relevant network parameters for bandwidth, delay, and loss. We developed a mass simulation environment to run simulations on a multitude of server instances in parallel. Additionally, we extended Mininet's Python API[1] to natively support both Apache 2.4.7 and Firefox 39.0. Our extension supports the manipulation of configuration parameters, e.g. *setFirefoxParameter(h1, 'network.http.pipelining', 'true')*, and to control a headless running Firefox from the Mininet API.

4. EVALUATION

For the evaluation, we implemented the presented frameworks and optimized the page load time of the front pages of three popular web pages (Google, Amazon, and Wikipedia). As configuration parameters, we used three congestion controls (Reno, Cubic, and Vegas [10, 2]) and the HTTP pipelining modes of Firefox: *off*, *on*, and *aggressive*. HTTP pipelining allows the client to send multiple requests over a persistent connection without waiting for a response and thereby reduces the number of required round trip times for multiple requests [17]. We tested 650 network conditions for access networks as reported in the literature: bandwidth between 0.384 and 100 Mbit/s, latency between 0 and 300ms, and packet drop rates between 0 and 5% [1, 3, 4, 5, 12, 13, 19]. Each simulation was repeated 10 times, leading to $3 \cdot 3 \cdot 3 \cdot 650 \cdot 10 = 175,500$ simulations.

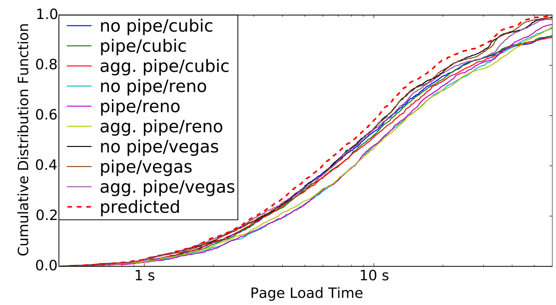


Figure 3: Cumulative page load time distribution.

The evaluation shows that *no pipe/cubic* is the optimal configuration for 26% of all tested network conditions (Figure 2a). *No pipe/vegas* and *pipe/vegas*, however, have the lowest average page load time 13.7s of all configurations (Figure 2b). Thus, they should be chosen for a static solution. The trained prediction engine has an average page load time of 12.2s. The optimal solution, which always executes the optimal configuration, would lead to 11.5s. The prediction engine easily outperforms static solutions, and reduces the average page load time by 1.5s of the possible 2.2s, realizing 68% of the possible improvement of an optimal prediction. The CDF shows that the predicted solution improves especially medium and high page load times (Figure 3).

5. CONCLUSION AND OUTLOOK

This paper proposed a simulation-based retrieval of adaptation knowledge for networking applications. We showed that runtime adaptations between Firefox configurations and congestion control algorithms reduce the average page load time. For future work, we plan to apply our framework on more applications and to evaluate the classifiers in real world scenarios.

Acknowledgments

This work has been funded by the German Research Foundation (DFG) as part of projects A02 within the Collaborative Research Center (CRC) 1053 – MAKI.

[1]Available at http://dvs.tu-darmstadt.de/simu

6. REFERENCES

[1] A. Balachandran, G. M. Voelker, P. Bahl, and P. V. Rangan. Characterizing user behavior and network performance in a public wireless lan. In *Proceedings of the 2002 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '02, pages 195–205, New York, NY, USA, 2002. ACM.

[2] L. S. Brakmo and L. L. Peterson. Tcp vegas: End to end congestion avoidance on a global internet. *IEEE J.Sel. A. Commun.*, 13(8):1465–1480, Sept. 2006.

[3] Y.-C. Chen, Y.-s. Lim, R. J. Gibbens et al. A Measurement-based Study of MultiPath TCP Performance over Wireless Networks. In *IMC*, 2013.

[4] S. Deng, R. Netravali, A. Sivaraman, and H. Balakrishnan. WiFi, LTE, or Both?: Measuring Multi-Homed Wireless Internet Performance. In *Proceedings of the 2014 Conference on Internet Measurement Conference*, IMC '14, pages 181–194, New York, NY, USA, 2014. ACM.

[5] M. Dischinger, A. Haeberlen, K. P. Gummadi, and S. Saroiu. Characterizing residential broadband networks. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, IMC '07, pages 43–56, New York, NY, USA, 2007. ACM.

[6] R. Fielding, J. Gettys, J. Mogul et al. Hypertext transfer protocol – http/1.1, 1999.

[7] A. Froemmgen, R. Rehner, M. Lehn, and A. Buchmann. Fossa: Learning ECA Rules for Adaptive Distributed Systems. In *Autonomic Computing (ICAC), 2015 IEEE International Conference on*, pages 207–210. IEEE, 2015.

[8] A. Froemmgen, R. Rehner, M. Lehn, and A. Buchmann. Fossa: Using Genetic Programming to Learn ECA Rules for Adaptive Networking Applications. In *Local Computer Networks (LCN), 2015 IEEE International Conference on*. IEEE, 2015.

[9] W. P. W. Group. Navigation timing - w3c recommendation 17 december 2012. http://www.w3.org/TR/2012/REC-navigation-timing-20121217/, 2012.

[10] S. Ha, I. Rhee, and L. Xu. Cubic: A new tcp-friendly high-speed tcp variant. *SIGOPS Oper. Syst. Rev.*, 42(5):64–74, July 2008.

[11] N. Handigol, B. Heller, V. Jeyakumar et al. Reproducible network experiments using container-based emulation. In *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '12, pages 253–264, New York, NY, USA, 2012. ACM.

[12] J. Huang, F. Qian, Y. Guo et al. An in-depth study of lte: Effect of network protocol and application behavior on performance. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, pages 363–374, New York, NY, USA, 2013. ACM.

[13] T. Jehaes, D. De Vleeschauwer, T. Coppens et al. Access network delay in networked games. In *Proceedings of the 2Nd Workshop on Network and System Support for Games*, NetGames '03, pages 63–71, New York, NY, USA, 2003. ACM.

[14] D. Jin, X. Qu, M. B. Cohen, and B. Robinson. Configurations everywhere: Implications for testing and debugging in practice. In *Companion Proceedings of the 36th International Conference on Software Engineering*, ICSE Companion 2014, pages 215–224, New York, NY, USA, 2014. ACM.

[15] J. Kephart and D. Chess. The vision of autonomic computing. pages 41–50, Jan 2003.

[16] U. Manpage. ss - another utility to investigate sockets. http://manpages.ubuntu.com/manpages/trusty/en/man8/ss.8.html, 2015. [Online; last accessed 09.07.2015].

[17] H. F. Nielsen, J. Gettys, A. Baird-Smith et al. Network performance effects of http/1.1, css1, and png. In *Proceedings of the ACM SIGCOMM '97 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '97, pages 155–166, New York, NY, USA, 1997. ACM.

[18] C. Sammut and G. I. Webb. *Encyclopedia of Machine Learning*. Springer Publishing Company, Incorporated, 1st edition, 2011.

[19] J. Sommers and P. Barford. Cell vs. wifi: On the performance of metro area mobile connections. In *Proceedings of the 2012 ACM Conference on Internet Measurement Conference*, IMC '12, pages 301–314, New York, NY, USA, 2012. ACM.

[20] K. Winstein and H. Balakrishnan. Tcp ex machina: Computer-generated congestion control. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, pages 123–134, New York, NY, USA, 2013. ACM.