Digital Bifurcation Analysis of TCP Dynamics

Nikola Beneš, Luboš Brim, Samuel Pastva(✉), and David Šafránek

Faculty of Informatics, Masaryk University, Brno, Czech Republic
{xbenes3,brim,xpastva,safranek}@fi.muni.cz

Abstract. Digital bifurcation analysis is a new algorithmic method for exploring how the behaviour of a parameter-dependent computer system varies with a change in its parameters and, in particular, for identification of bifurcation points where such variation becomes dramatic. We have developed the method in an analogy with the traditional bifurcation theory and have it successfully applied to models taken from systems biology. In this case study paper, we demonstrate the appropriateness and usefulness of the digital bifurcation analysis as a push-button alternative to the classical approaches as traditionally used for analysing the stability of TCP/IP protocols. We consider two typical examples (congestion control and buffer sizes throughput influence) and show that the method provides the same results as obtained with classical non-automatic analytical and numerical methods.

1 Introduction

The objective of the bifurcation theory is to study qualitative changes to the properties of a parameter-dependent system as parameters are varied. The method is typically applied to continuous-time or discrete-time dynamical systems. Even a tiny change in parameters may cause a dynamical system to exhibit entirely different qualitative features. Such dramatic changes in the topology of the phase space of a dynamical system are known as *bifurcations*, and the values of the parameters for which a bifurcation occurs are called *bifurcation points*. For a complete global understanding of a complex dynamical system, it is essential to know the bifurcation points, as well as the parameter ranges in which there is no fundamental change. A simple example of a real-life bifurcation is the phase transition of water to ice at the temperature of 0 °C. At this critical temperature, a tiny change in the temperature results in a "sudden" systematic change in the substance. The two materials are governed by a different set of parameters and qualitative properties. For example, we can talk about cracking ice but not water.

Non-linear dynamical systems appearing in physics, biology or economy are not the only source of bifurcation phenomena. Even computer systems can

This work has been partially supported by the Czech Science Foundation grant No. 18-00178S.

suddenly alter the quality of their behaviour. A simple example might be a significant performance degradation of a computation caused by system swapping. Studying bifurcations in computer systems can provide an additional formal analysis ingredient leading to a better understanding of critical systems properties, like stability or robustness.

Inspired by the bifurcation theory for dynamical systems, we have developed an approach that allows analysing how the dynamics (runs, state transitions) of a discrete computer system changes when its parameters are changed [6,11]. We call the method *digital bifurcation analysis*. In the approach, the qualitative changes in the behaviour are represented as changes in the truth-value of temporal formulae defining specific behaviour (portrait) pattern of the system. The method for computing results of the bifurcation analysis (typically presented as *bifurcation diagrams*) uses our novel symbolic parallel parameter synthesis algorithm [3] which itself builds on the model-checking technology. As the approach employs a hybrid temporal logic for which the algorithm is computationally demanding we have also developed specialised algorithms dedicated to some specific formulae/patterns and thus working more efficiently.

Example of such patterns are attractors, which we see as a particular class of patterns representing the states of the system in which the system's execution persists in the long-time horizon, i.e., the so-called invariant subsets of the state-space towards which the system's runs are attracted. In computer systems, the most typical attractors can be observed in the form of terminal strongly connected components (tSCCs) [38]. We have developed an efficient parallel algorithm for detecting tSCCs in parametrised graphs in [1], and we use this algorithm in our two case studies. We have already successfully applied the digital bifurcation analysis to several models from systems biology [4,5].

In this case-study paper, we report on the application of digital bifurcation analysis to the Transmission Control Protocol (TCP) which currently facilitates most of the internet communication. One of the severe problems in practical applications of TCP is congestion, appearing when the required resources overrun the capacity of internet communication. Over the past years, many internet congestion control mechanisms have been developed to ensure the reliable and efficient exchange of information across the internet, such as Active Queue Management (AQM). Bifurcation analysis of TCP under various congestion control mechanisms have been studied by several authors [16,25,30,32,40,41]. All have used a continuous-time model (e.g., the fluid model) and applied traditional mathematical methods of bifurcation analysis, including simulations, to detect parameter values when the system passes through a critical point, the system loses its stability, and a so-called Hopf bifurcation occurs [22].

Our approach to bifurcation analysis does not require to remodel the given discrete system in terms of a continuous-time dynamical system. Digital bifurcation analysis works directly on discrete models represented as state transition systems. Furthermore, the method is, unlike mathematical methods, fully automatic and does not need mathematical skills to be utilised. Another advantage is that the method is scalable to state spaces with tens of variables and tens of possibly dependent parameters, overcoming thus significantly the limits of traditional mathematical methods. Last but not least the method is advantageous in performing global bifurcation analysis, which is harder to compute than the local analysis where bifurcation points are expected to be approximately known in advance.

It is important to stress that the purpose of this case-study paper is not to propose any new congestion control mechanisms or protocols. We aim to provide a demonstration of the appropriateness and usefulness of the digital bifurcation analysis as a push-button technique that makes a promising alternative to the classical approaches when analysing stability and robustness of TCP protocol specifications and implementations. To that end, we consider two different case studies targeting TCP. In both of them, we analyse how the structure and quality of attractors change when the parameters change. The first one deals with TCP that uses the Random Early Detection (RED) method [14] as an active queue management mechanism to control congestion. Although the RED mechanism alone is easy to understand, its interaction with TCP connections is rather complicated and is not well understood. In [33] the authors used a deterministic non-linear dynamical model of the TCP-RED protocol (together with detailed simulations) to demonstrate that the model exhibits a transition between a stable fixed point and an oscillatory or chaotic behaviour as parameters are varied. In our case study, we were able to achieve the same results fully automatically using our method. In the second example, we consider TCP itself combined with essential performance-oriented extensions. We analyse how the sizes of the send and receive socket buffers influence the throughput; in particular, we identify the combinations of sizes (bifurcation points) for which we observe a dramatic drop. The results we have achieved are in accordance with [28].

It is worth noting that bifurcation analysis provides a conceptually very different view of the protocol functionality than what is usually addressed by formal verification methods. The goal of verification is to prove the correctness of a system specification for all initial states and in the case of parametrised verification also regardless of the number of its components, or the parametrised domain of variables. On the other hand, the goal of bifurcation analysis of parametrised systems is to identify parameter values for which the system suddenly changes its behaviour regardless of its correctness.

Several examples of the TCP protocol verification are in [8,13,18,23,35–37]. As regards parametric verification, the Bounded Retransmission Protocol (BRP) for manually derived constraints has been checked by parametric model-checking in [19], the Stop-and-Wait Protocol (SWP) has been targeted in [15] for all possible values of the maximum sequence number and the maximum number of retransmissions parameters. We are not aware of any formal verification method that would address the bifurcations of the protocol behaviour.

Finally, we discuss the approaches related to bifurcation analysis. To the best of our knowledge, the only related approach to bifurcation analysis that also employs methods of formal verification has been presented in [20,21]. The authors address the identification of bifurcation points in non-trivial dynamics of a numerical cardiac-cell model represented using a hybrid automaton. The method is based on guided-search-based bounded-time reachability analysis used to estimate ranges of parameter values displaying two complementary patterns of systems behaviour. These ranges are computed for bounded-time reachability and over-approximated up to a particular δ-precision due to the underlying δ-decision algorithm.

2 Attractor Analysis Workflow

We first describe the standard scenario for digital bifurcation analysis focused on attractor analysis. The input is a parametrised system and a certain classification of stability-based attractor properties that we are interested in. The system is in the model design phase formalised as a discrete finite-state model and subsequently via the state-space generation procedure turned into a parametrised graph. How the initial model is obtained and what language the model is written in is domain-specific and is explained later when describing the case studies. The classification of the attractor properties specifies what shapes and forms of attractors we want to consider distinct enough to express a dramatic change in the system's behaviour. In the simplest case, which we call the *counting version* of our problem, we may be merely interested in the number of attractors and consider two parametrisations of a system non-equivalent if this number changes. More interesting cases may classify the attractors according to various stability-related properties, such as oscillations. The core parametric analysis algorithm then computes the parametric tSCC map. The resulting map is post-processed, producing e.g. the visualisation of bifurcation diagram, plots, tables, etc. The workflow of our method for the digital bifurcation analysis of attractors is summarised in Fig. 1.

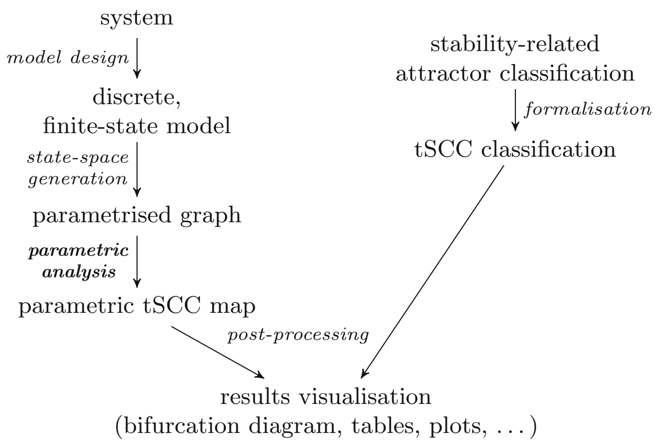

Fig. 1. Attractor analysis method workflow.

In general, our digital bifurcation analysis algorithm presupposes that the state space of the model has the form of a parametrised Kripke structure. In this case study, we are interested in attractor properties that are independent of the atomic proposition valuation. We, therefore, consider a simpler formalism here, namely that of parametrised graphs which are directed graphs with self-loops allowed and edges labelled by parameters taken from a given parameter set.

Definition 1. *A* graph *is a pair* (V, E) *where* V *is a finite set of* vertices *and* $E \subseteq V \times V$ *is a set of* edges*. A* parametrised graph *is a triple* $G = (V, E, \mathbb{P})$ *where* $\mathbb{P}$ *is a set of* parametrisations *and* $E : V \times V \to 2^{\mathbb{P}}$ *such that for each* $p \in \mathbb{P}$, $G_p = (V, E_p = \{(u, v) \mid p \in E(u, v)\})$ *is a graph. We call* G_p *the* projection of G on p*.*

To be able to investigate the properties of the attractors in the system, we need to use a notion that is analogous to an attractor in a parametrised graph. In dynamical systems theory, an attractor [27] is the smallest set of states (points in the phase space) invariant under the system dynamics. Parametrised graphs can be regarded as discrete abstractions of a dynamical system in which the dynamics are represented using paths in the graph. The respective abstraction of the notion of an attractor thus coincides with the notion of a terminal strongly connected component (tSCC) of a graph.

Definition 2. *Let* $G = (V, E)$ *be a graph. We say that a vertex* $t \in V$ is reachable *from a vertex* $s \in V$ *if* $(s, t) \in E^*$ *where* E^* *denotes the reflexive and transitive closure of* E*. A set of vertices* $C \subseteq V$ *is* strongly connected*, if* v *is reachable from* u *for any two vertices* $u, v \in C$*. A* strongly connected component *(SCC) is a* maximal *strongly connected set* $C \subseteq V$*, i.e. such that no* C' *with* $C \subsetneq C' \subseteq V$ *is strongly connected. A strongly connected component* C *is called* terminal *(tSCC) if* $(C \times (V \setminus C)) \cap E = \emptyset$*, i.e. there are no edges leaving* C*.*

We are now ready to state the algorithmic problem whose solution forms the basis of our method.

Terminal SCCs Enumeration Problem. Let $G = (V, E, \mathbb{P})$ be a parametrised graph. The goal is to enumerate, for every parametrisation $p \in \mathbb{P}$, all tSCCs in the graph G_p, the projection of G on p.

In this general version of our problem, the output is going to be a mapping that assigns to each $p \in \mathbb{P}$ the set of all tSCCs of G_p. We call this the *parametric tSCC map*. This map may be then further processed and visualised. We are mainly interested in the *bifurcation diagram* of the model. This diagram is a plot which partitions the parameter space into regions where the behaviour of the system is qualitatively invariant. In the case of a single parameter, this type of one-dimensional diagram is typically augmented by a second dimension which presents the location of the tSCCs with respect to a chosen system variable.

To be able to distinguish between quantitatively different behaviour of the system, we need to formalise the classification of stability-based attractor properties in terms of tSCCs. We thus get a classification function that separates tSCCs into classes. Two parametrisations of a system are then said to be qualitatively different if their respective graphs differ in the count of tSCCs belonging to each class. In the case of the counting version, we thus consider one class of tSCCs only. Here, parametrisations of a system are considered to be qualitatively different if their graphs contain a different number of tSCCs. In the more detailed cases, we can classify tSCCs according to size (small vs large), density (sparse vs dense), graph-specific properties (bipartite vs non-bipartite) etc.

For an example of how these classifications relate to the classical bifurcation analysis, we may see bipartite tSCCs as representing oscillatory patterns in attractors. The change from a small non-bipartite tSCC to a bipartite tSCC can be thus seen as an analogy of the Hopf bifurcation. In our two case studies, we distinguish between sinks (single-state tSCCs), bipartite (oscillatory) tSCC, and other tSCCs, which are further differentiated between small and large, based on a chosen domain-specific threshold.

The rest of this section gives a brief overview of the parallel algorithm for solving the tSCCs enumeration problem that we have developed in [1].

2.1 Core Algorithm

First, note that a simple *sequential* solution to the problem is to use any reasonable SCC decomposition algorithm (e.g. Tarjan's [39]) and enumerate the tSCCs in the residual graph. However, all known optimal sequential SCC decomposition algorithms use the depth-first search algorithm, which is suspected to be non-parallelisable [34]. There are known parallel SCC decomposition algorithms; for a survey, we refer to [2]. Our approach is based on the observation that we do not have to compute all of the SCCs to enumerate the terminal ones.

Furthermore, instead of scanning through all parametrisations and solving the problem for every one of them separately our approach deals with sets of parametrisations directly. This makes our algorithm suitable for use in connection with various kinds of symbolic set representations. The reason for using a parallel algorithm is the necessity to deal with the high computational demands of the method as discussed in [1].

The main idea of the Terminal Component Detection (TCD) algorithm lies in repeated reachability, which is known to be easily parallelisable. To explain the method, we start with a non-parametrised version of the algorithm. The following explication is illustrated in Fig. 2. Let us assume a given (non-parametrised) graph $G = (V, E)$. We choose an arbitrary vertex $v \in V$ (denoted by the double circle in the illustration) and compute all vertices reachable from v; let us call the resulting set of vertices F. We further compute the set of all vertices backwards-reachable from v inside F; we call the resulting set B. Finally, we compute all vertices backwards-reachable from any vertex of F; let us call this set B'.

Clearly, B is an SCC of the graph, and moreover, it is a terminal SCC iff $F \setminus B$ is empty. Furthermore, $B' \setminus F$ contains no tSCCs: all vertices in $B' \setminus F$ have a path to a vertex in F. We recursively run the algorithm in $F \setminus B$ and $V \setminus B'$ if non-empty. Observe that no tSCC may intersect both of these sets and these two subproblems can be thus dealt with independently (i.e. in parallel). Note that

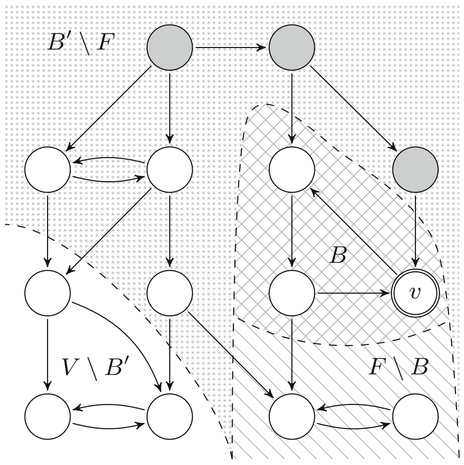

Fig. 2. Illustration of the non-parametrised version of our algorithm.

every time the algorithm is (recursively) started, its input is an induced subgraph of the original graph that satisfies the precondition that all its tSCCs are tSCCs of the original graph. These observations together imply the correctness of the algorithm.

The asymptotic complexity of the algorithm in its non-parametric version is of the order $\mathcal{O}(|V| \cdot (|V| + |E|))$ as in the worst case, every iteration may eliminate a single vertex of the graph. The actual performance of the algorithm strongly depends on the choice of the initial vertex v. If we consistently choose v that lies close to (or directly in) a tSCC of the graph, the complexity gets linear. Of course, such choice cannot be made in advance. The paper [1] discusses the impact of several heuristics that try to approximate this choice.

The algorithm can also be made more efficient using a *trimming* subprocedure in the manner of [26], i.e. removing all vertices without incoming edges. In Fig. 2, the removed vertices are marked in grey; furthermore, the $V \setminus B'$ part of the graph contains one vertex that would be removed in the next recursive run.

To extend the basic idea to parametrised graphs, we use a notion of parametrised sets of vertices. Formally, a parametrised set of vertices $\widehat{A}$ is a function $\widehat{A} : V \to 2^{\mathbb{P}}$. To deal with parametrised sets, we use a generalisation of the standard set operations. All the operations are performed element-wise, e.g. the union of parametrised sets $\widehat{A} \cup \widehat{B}$ is defined as the parametrised set $\widehat{C}$ such that $\widehat{C}(v) = \widehat{A}(v) \cup \widehat{B}(v)$ for all v. The parametrised set of all vertices and all parametrisations is given by $\widehat{V}$ such that $\widehat{V}(v) = \mathbb{P}$ for all $v \in V$.

The notions of the forward and backward reachable sets can be easily extended to the parametrised setting. They can be computed by a fixed-point algorithm which iterates the parametrised successor (or predecessor) operator. Given a parametrised set of vertices $\widehat{X}$, the successor operator computes the parametrised set $\widehat{Y}$ such that $\widehat{Y}(v) = \widehat{X}(v) \cup \bigcup_{u \in V}(\widehat{X}(u) \cap E(u, v))$ and similarly for the predecessor operator.

The parametrised algorithm then proceeds as described in the previous, extended with the parametrised sets. One further key difference is that instead of choosing one starting vertex, we need to choose a set of starting vertices with disjoint parametrisation sets that together cover all parametrisations that are present in the currently explored parametrised subgraph. The reason for this, as well as a discussion on heuristics that allow choosing such sets efficiently, can be again found in [1].

In the worst case, when parametrisations are represented explicitly, the asymptotic complexity of the algorithm is of the order $\mathcal{O}(|\mathbb{P}| \cdot |V| \cdot (|V| + |E|))$. The actual performance of the algorithm depends on various choices and heuristics. It can also be strongly influenced by the usage of a symbolic encoding of the parametrised sets. In this paper, the sets of parametrisations are represented symbolically using an interval encoding, similar to the one used in [10]. Other options for a symbolic representation of parameters include SMT formulae [3].

3 Case Studies

In this section, we present two case studies focusing on discovering bifurcations in the behaviour of the TCP protocol. Each of them addresses a different essential aspect of the protocol, namely congestion control and packet flow stability. We demonstrate how the digital bifurcation analysis can aid in the design, analysis and control of these discrete reactive systems.

In the first case study, we consider a relatively common setting in the standard bifurcation theory: A discrete map governing the behaviour of the RED congestion control mechanism. This mechanism prevents congestion on network nodes such as routers and is subject to changes in its behaviour due to different internal and external parameters. We show how different parameters influence the stability of the mechanism and how a hypothetical system administrator or an automated controller can use this information to avoid faulty behaviour.

The second case study presents an entirely discrete model of the basic TCP focusing on the stability of packet flow. We study the influence of the sender and receiver buffer sizes on the behaviour of the protocol and its ability to transfer packets in a timely manner. We assume the role of a hypothetical protocol designer and consider a set of extensions and modifications to the protocol proposed by various networking experts. We observe that such extensions and their interplay can introduce bifurcations leading to serious degradation of the protocol performance.

The case studies are implemented with the help of the tool Pithya [7] which provides the necessary parametrised graph analysis algorithms. The source code of this implementation is available at https://github.com/sybila/tcp-bifurcation. All experiments were performed on a typical 4-core 3 GHz desktop computer with 16 GB of RAM.

3.1 Instabilities in TCP-RED

This case study addresses the congestion control in TCP. The congestion control mechanism prevents the protocol from overloading the network with too many packets. The problem has two important aspects. The first aspect is the congestion control on the sender side that has to ensure maximal throughput for a single flow of packets. The second aspect is the congestion control on other network nodes, such as routers, where several connections meet.

One of the common approaches to implementing the congestion control on routers is the Random Early Drop (RED) method proposed in [14]. This technique explicitly drops packets as the router queue starts to fill up. Consequently, senders are indirectly notified (by observing the packet loss) that the link is approaching a congested state before the situation becomes critical.

Model Description. To study the RED mechanism, we use a discrete time model proposed in [33]. In Fig. 3, we present the model equations and a basic description of all model variables and constants. Detailed aspects of the model design are given in the original paper.

$$p_t(\overline{q}_t) = \begin{cases} 0 & \overline{q}_t \in [0, q_l] \\ \frac{\overline{q}_t - q_l}{q_u - q_l} p_{max} & \overline{q}_t \in (q_l, q_u) \\ 1 & \overline{q}_t \in [q_u, B] \end{cases} \quad (1) \qquad q_t(p_t) = \begin{cases} B & p_t \in [0, p_l] \\ \frac{n \cdot k}{\sqrt{p_t}} - \frac{c \cdot d}{m} & p_t \in (p_l, p_u) \\ 0 & p_t \in [p_u, 1] \end{cases} \quad (2)$$

$$\overline{q}_{t+1}(\overline{q}_t) = (1 - w) \cdot \overline{q}_t + w \cdot q_t(p_t(\overline{q}_t)) \quad (3)$$

$$\begin{aligned} \text{maximum buffer size } B &= 3750 \\ \text{lower queue threshold } q_l &= 250 \\ \text{upper queue threshold } q_u &= 750 \\ \text{packet size } m &= 4\text{kb} \\ \text{maximum drop rate } p_{max} &= 0.1 \\ \text{number of TCP connections } n &= 250 \\ \text{propagation delay } d &= 0.1\text{s} \\ \text{link capacity } c &= 75\text{Mb/s} \end{aligned}$$

$$\begin{aligned} \text{drop rate } p_t &\in [0, 1] \\ \text{queue size } q_t &\in [0, B] \\ \text{average queue size } \overline{q}_t &\in [0, B] \\ \text{lower drop threshold } p_l &= \left(\frac{n \cdot m \cdot k}{dc + Bm}\right)^2 \\ \text{upper drop threshold } p_u &= \left(\frac{n \cdot m \cdot k}{dc}\right)^2 \\ \text{rate constant } k &= \sqrt{3/2} \\ \text{averaging weight } w &= 0.15 \end{aligned}$$

Fig. 3. A discrete time model of the RED congestion control behaviour. The individual constants are stated with basic explanations and default values. The parameters are selected from the given set of constants and their bounds are specified later with the corresponding experiments.

The model assumes n connections flowing through a single RED-capable router. All connections share basic properties, namely the packet size and the propagation delay. In such a case, the situation can be simplified by considering only a single combined flow, as the router cannot differentiate between the individual flows anyway. The router then maintains the current drop rate p_t (Eq. 1) and the queue size q_t (Eq. 2) based on the current exponentially weighted average queue size $\overline{q}_t$ (Eq. 3).

A typical scenario is that a network administrator takes control over parameters such as the averaging weight w or the queue thresholds q_l and q_u. Furthermore, it is also important to consider the influence of the connection count n and the propagation delay d, as these numbers will change depending on the current network load.

Parametrised Graph. To analyse the model, we require a finite parametrised graph $G = (V, E, \mathbb{P})$. Here, $\mathbb{P}$ is the parameter space given by the chosen model parameters (we specify the chosen parameters for each experiment later). In Eq. 3, we write $\overline{q}_{t+1}(\overline{q}, \lambda)$ for $\lambda \in \mathbb{P}$ to specify the parametrised version of the model.

We assume $s + 1$ thresholds $t_0 < t_1 < \ldots < t_s$ such that $t_0 = 0$ and $t_s = B$. These thresholds partition the state space of the variable $\overline{q}$ into s intervals $[t_0, t_1], \ldots, [t_{s-1}, t_s]$, denoted as $I_1, \ldots, I_s$. These intervals then represent vertices of our parametrised graph $V = \{I_i \mid i \in [1, s]\}$.

Next, we construct the parametrised edges between our intervals so that they over-approximate the behaviour of the original discrete map. Let us consider two intervals I_i and I_j and the edge from I_i to I_j. Clearly, the set of parametrisations $E(I_i, I_j)$ has to include all parametrisations λ such that for some $\overline{q}_t \in I_i$ it holds that $\overline{q}_{t+1}(\overline{q}_t, \lambda) \in I_j$. We compute these sets using interval arithmetic, ensuring that all such parametrisations are included.

Finally, since our graph over-approximates the original discrete map, each tSCC over-approximates some attractor(s) of the original system. Furthermore, the precision of this over-approximation can be refined by introducing additional thresholds or substituting interval arithmetic for a more sophisticated approximation method, e.g., Taylor models [24].

Analysis Results. The analysis procedure consists of two scenarios:

Scenario 1: Consider a system designer who studies the effects of parameters to assess correct settings ensuring the stable behaviour of the protocol. In Fig. 4(a) and (b), the locations and types of attractors are shown for parameters w and n, respectively. It can be seen that increasing the parameter w has a destabilising effect – the small (stable) tSCC (component size $\leq 0.01 \cdot B$) turns into a bipartite tSCC (representing oscillation) and finally into a large non-bipartite tSCC. On the other hand, the effect of the connection count n is complementary: a higher number of connections stabilise the behaviour (Fig. 4b). Additionally, the protocol behaves as expected in the stable region – w does not influence the location of the steady state whereas a higher number of connections require higher queue sizes to accommodate the increased data flow. Using this

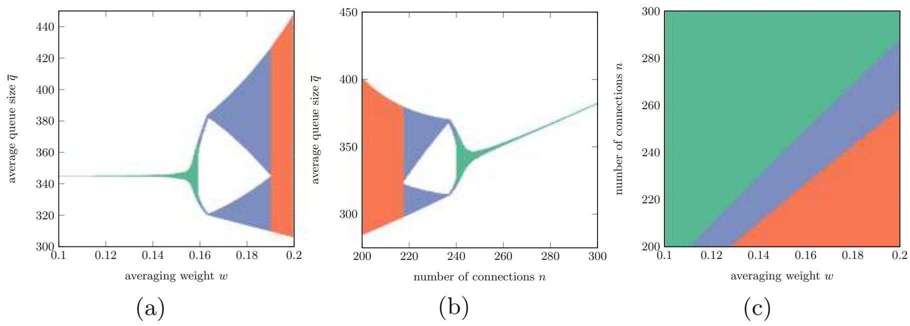

Fig. 4. Bifurcation diagrams showing the location and character of the tSCC depending on model parameters in the RED model. The green region indicates a small component ($\leq 0.01 \cdot B$), the blue region shows oscillatory behaviour (bipartite graph), and the red region corresponds to a large non-bipartite tSCC. (a) $w \in [0.1, 0.2]$ and $n = 250$; (b) $n \in [200, 300]$ and $w = 0.15$; (c) $w \in [0.1, 0.2]$ and $n \in [200, 300]$. (Color figure online)

kind of analysis, a general overview of the systems behaviour w.r.t. the given parameters can be directly obtained in a matter of minutes.

Scenario 2: Assume an administrator (or an automated controller) is supposed to adjust the parameter w to preserve the correct functionality of the system subject to a varying number of connections n. In Fig. 4c, it is shown how the character of the attractor changes with the controllable parameter w and the external condition n. This allows the administrator to select optimal values for the given situation. Note that while this specific type of diagram does not show the concrete location of components, it is still contained in the method results and can be used to support the decision further. While this type of analysis is certainly more computationally challenging, it can still be performed in under one hour.

3.2 Packet Flow Stability

The TCP specification as defined in RFC 793 [31] provides a fundamental description of the TCP protocol such as the packet format or the state machine for event processing. However, many implementation and performance aspects were not addressed in the original specification. Therefore in the subsequent years, several extensions and improvements of the protocol functionality have been introduced [9,12,29].

Nowadays, many well-tested, production ready implementations of TCP exist. However, as demonstrated in [28], non-standard network configurations and combinations of various modifications can cause problems even in well-established implementations. Furthermore, new implementations are still being developed where such fundamental problems can easily re-appear [17].

In this case study, we assume the role of a hypothetical protocol engineer. We introduce a basic parametrised model of TCP according to RFC 793 [31] extended with two performance-oriented modifications, namely *delayed acknowledgement* and *Nagle's algorithm*. We observe that these modifications, while useful in many instances, can introduce unexpected bifurcations in the behaviour of the protocol. Additionally, we compare our results with [28].

Model Description. We consider a model of TCP based on RFC 793 [31] extended with Nagle's algorithm according to RFC 896 [29] and delayed acknowledgement according to RFC 813 [12] and RFC 1122 [9]. We assume a single sender which sends an uni-directional infinite stream of data to a single receiver connected by a reliable link with unlimited capacity. As parameters, we assume a fixed maximal buffer size S for the sender and R for the receiver. Finally, the size of each packet is limited by the Maximum Segment Size (MSS) set by the network administrator.

Since we are not interested in the exact values of the transmitted data bytes, we can model the state of the protocol using the number of bytes in each protocol phase. This abstraction leads to the following five state variables:

- `W` – the number of bytes in the send buffer waiting to be sent;
- `D` – the list of data packet sizes in transit;
- `U` – the number of bytes in the receive buffer waiting to be acknowledged;
- `A` – the list of acknowledgement packets in transit;
- `ACK` – the out-of-order acknowledgement flag.

Furthermore, we use `outstanding` to denote the number of unacknowledged bytes (`U` plus the sum of all elements in `D` and `A`). Since the protocol is not limited by the link capacity, we assume the available `window` is always equal to $min(S, R)$ minus `outstanding` bytes. Notice that all the bytes considered by the model variables must be stored in the send buffer (the sender must keep the data until acknowledgement arrives), whereas only the bytes waiting to be acknowledged are stored in the receive buffer.

The dynamics of the model is governed by a set of discrete asynchronous events. Each event can be only executed when its preconditions are met. As our parametrised graph, we consider the graph of the protocol states reachable from the initial configuration where all channels are empty, and all variables are zero. The model consists of the following discrete events:

Copy data from the application: Before sending, the data needs to be copied from the application to the kernel memory where the networking layer operates. This occurs in 1024-byte chunks such that at least for every four chunks, the copying is interrupted to send available data right away [28] if possible:

$$\begin{gathered}\mathtt{W} = \mathtt{W} + k \cdot 1024;\ \text{ where } k \in [1..4] \text{ is maximal}\\ \text{such that } (k \cdot 1024 + \mathtt{W} + \mathtt{outstanding} \leq S)\end{gathered}$$

Send full packet: When MSS unsent bytes are available in the send buffer and the `window` capacity is sufficient, a full packet can be constructed and sent:

$$\mathtt{W} = \mathtt{W} - \text{MSS};\ \mathtt{D} = \mathit{append}(\mathtt{D}, \text{MSS});\ \text{ when } (\mathtt{window} \geq \text{MSS} \wedge \mathtt{W} \geq \text{MSS})$$

Send partial packet: When less than MSS unsent bytes are available, or the `window` is not large enough, the protocol can decide to send a partial packet. This decision is governed by Nagle's algorithm which dictates that a partial packet can be sent only when there are no outstanding bytes. This criterion prevents the sender from sending unnecessary small packets in an unbuffered stream of data:

$$\begin{gathered}\mathtt{W} = \mathtt{W} - \mathit{packet};\ \mathtt{D} = \mathit{append}(\mathtt{D}, \mathit{packet});\ \text{where}\\ (\mathit{packet} = \mathit{min}(\mathtt{window}, \mathrm{MSS}, \mathtt{W}) \wedge \mathtt{outstanding} = 0)\end{gathered}$$

Receive and acknowledge packet: The receiver can process and acknowledge any data packet (we assume the data is immediately handed over to the application). However, to avoid a large number of small acknowledgement packets, the packet acknowledgement is often delayed until a sufficient amount of data is received (RFC 813). In our case, we use the threshold specified in [28] – 35% of R. In RFC 1122, this rule is further augmented to send an acknowledgement packet whenever two full segments are received:

$$\begin{gathered}\mathtt{A} = \mathit{append}(\mathtt{A}, \mathtt{U} + \mathit{head}(\mathtt{D}));\ \mathtt{D} = \mathit{tail}(\mathtt{D});\ \mathtt{U} = 0;\ \text{when}\\ (|\mathtt{D}| > 0 \wedge \mathtt{U} + \mathit{head}(\mathtt{D}) \geq \mathit{min}(0.35 \cdot R, 2 \cdot \mathrm{MSS}))\end{gathered}$$

Receive without acknowledgement: When the rules of delayed acknowledgement are not met, the data bytes are transferred to the receive buffer instead:

$$\begin{gathered}\mathtt{U} = \mathtt{U} + \mathit{head}(\mathtt{D});\ \mathtt{D} = \mathit{tail}(\mathtt{D});\ \text{when}\\ (|\mathtt{D}| > 0 \wedge \mathtt{U} + \mathit{head}(\mathtt{D}) < \mathit{min}(0.35 \cdot R, 2 \cdot \mathrm{MSS}))\end{gathered}$$

Out-of-order acknowledgement: According to RFC 813, when data is received without immediate acknowledgement, a 200 ms timer should be started to acknowledge the data if no acknowledgement packet is generated in the meantime. However, as discussed in [28], regularly rescheduling such a timer can be an expensive operation. Therefore a cyclic timer acknowledging all received data every 200 ms is often used instead. In our model, we include this design decision by allowing one non-deterministic out-of-order acknowledgement packet to occur:

$$\mathtt{A} = \mathit{append}(\mathtt{A}, \mathtt{U});\ \mathtt{U} = 0;\ \mathtt{ACK} = 1 \text{ when } (\mathtt{U} > 0 \wedge \mathtt{ACK} = 0)$$

Process acknowledgement: The data cannot be removed from the send buffer until they are acknowledged. Thus whenever there is an acknowledgement packet in transit, the packet can be processed by the receiver:

$$\mathtt{A} = \mathit{tail}(\mathtt{A});\ \text{when } |\mathtt{A}| > 0$$

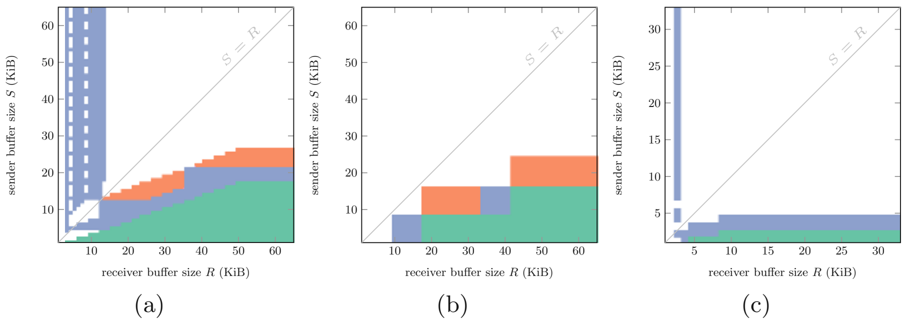

Fig. 5. The bifurcation diagrams showing the character of tSCCs depending on the model parameters in the TCP model. The white space indicates a single large tSCC; the other colours indicate the regions displaying various types of single state tSCCs. (a) MSS = 9204, 1 KiB increments of S and R; (b) MSS = 9204, 8 KiB increments of S and R; (c) MSS = 1460, 1 KiB increments of S and R. (Color figure online)

Analysis Results. In our analysis, we assume the buffer sizes S and R ranging from 1 KiB to 64 KiB in 1 KiB increments. First, we consider MSS to be 9204, as in [28]. This MSS configuration corresponds to a specific high-performance network and is not used in typical Ethernet configurations.

The complete results of our analysis are presented in Fig. 5a. In contrary to the previous case study, we consider the presence of a single large terminal tSCC as the desired behaviour (depicted in white). In this case, the situation indicates that the protocol is functioning properly. On the other hand, the presence of a small, single state tSCC means that the protocol cannot continue transmitting and is waiting for a time-out to resolve the problematic situation.

Additionally, based on enabling and disabling various extensions of the protocol model, we can distinguish between different bifurcation causes:

- *Delayed acknowledgement (DA):* With the delayed acknowledgement employed exclusively, the parametrisations satisfying $S < 0.35 \cdot R \wedge S < 2 \cdot \mathrm{MSS}$ can never trigger the automatic acknowledgement and thus rely on the acknowledgement time-out instead. The corresponding regions are depicted in green in Fig. 5.
- *Combination of DA and Nagle's algorithm:* A single state tSCC emerges whenever the amount of data necessary to trigger the next automatic acknowledgement cannot be sent due to Nagle's condition. The corresponding regions are depicted in blue in Fig. 5.
- *Combination of DA, Nagle's algorithm, and cyclic timer:* The regions depicted in red in Fig. 5 correspond to single state tSCCs appearing only when all the three extensions are enabled. The reason is that while delayed acknowledgement and Nagle's algorithm can coexist well under these parametrisations, the cyclic timer can cause transmissions of small packets which is not possible in the cases above.

In the case of $S < R$, the achieved results are in line with the findings of [28]. However, in the $R > S$ area, we observe a bifurcation caused by the interplay of delayed acknowledgement and Nagle's algorithm which has not been considered in the original paper. This bifurcation is caused by small packets sent right after an acknowledgement is received. The small packet is transmitted after the acknowledgement clears the outstanding bytes (so Nagle's condition holds), but before more data is copied into the send buffer (before the acknowledgement was received, the send buffer was full).

In [28], the situation might have been avoided by some undisclosed implementation or timing aspects. However, another possible explanation is that this behaviour has been overlooked because such issues never occurred during the experiments. In Fig. 5b, we present our reconstruction of the same results, but in 8 KiB increments. It corresponds exactly to the experimental evaluation presented in [28]. The described behaviour is absent in this case, since the 8 KiB increments avoid the problematic region entirely.

Finally, in Fig. 5c, we present the same analysis for the maximal buffer size of 32 KiB and MSS of 1460 bytes, which is the typical setting on an Ethernet network. In this case, the red region is completely absent, and while other bifurcations are still present, the problematic regions are much smaller due to the smaller MSS. This puts into perspective the drastic behavioural changes present for larger MSS values and shows how bifurcations can emerge in unexpected situations.

4 Discussion and Conclusion

In this paper, we have presented two case studies demonstrating a promising application of the digital bifurcation analysis in the domain of network protocols. To that end, we have utilised the methodology developed in our previous work.

The key aspects of the method as applied in this paper are the following. First, it gives rigorous results concerning the given models of the studied protocol. Second, it can be performed fully automatically. In general, the only tasks that have to be done manually are to acquire a suitable model and to post-process the results (incl. visualisation and interpretation). The crucial step to be done within the latter task is to classify the studied protocol properties in terms of attractors. However, this can be easily automated since the interest of a network administrator (or a designer) is primarily focused on parameter values for which the stable behaviour (a single simple attractor) disappears.

Both case studies show that the digital bifurcation analysis provides a methodologically different view on the protocol analysis than formal verification or testing. This is allowed by providing a global view of the protocol behaviour with respect to parameters. Due to the global approach, in the second case study, we have revealed regions in bifurcation diagrams that were omitted in previous studies.

The push-button characteristics of the digital bifurcation analysis allow making the results easily reproducible. All steps necessary to reconstruct both case studies are publicly available[1].

For future work, our primary intention is to target similar, but not yet fully explored, problems in network protocols using digital bifurcation analysis that will allow further fine-tuning (and generalisation) of the presented workflow.

References

1. Barnat, J., et al.: Detecting attractors in biological models with uncertain parameters. In: Feret, J., Koeppl, H. (eds.) CMSB 2017. LNCS, vol. 10545, pp. 40–56. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67471-1_3
2. Barnat, J., Chaloupka, J., Van De Pol, J.: Distributed algorithms for SCC decomposition. J. Logic Comput. **21**(1), 23–44 (2011)
3. Beneš, N., Brim, L., Demko, M., Pastva, S., Šafránek, D.: Parallel SMT-based parameter synthesis with application to piecewise multi-affine systems. In: Artho, C., Legay, A., Peled, D. (eds.) ATVA 2016. LNCS, vol. 9938, pp. 192–208. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46520-3_13
4. Beneš, N., et al.: Fully automated attractor analysis of cyanobacteria models. In: 2018 22nd International Conference on System Theory, Control and Computing, ICSTCC, pp. 354–359, October 2018
5. Beneš, N., Brim, L., Demko, M., Hajnal, M., Pastva, S., Šafránek, D.: Discrete bifurcation analysis with Pithya. In: Feret, J., Koeppl, H. (eds.) CMSB 2017. LNBI, vol. 10545, pp. 319–320. Springer, Heidelberg (2017)
6. Beneš, N., Brim, L., Demko, M., Pastva, S., Šafránek, D.: A model checking approach to discrete bifurcation analysis. In: Fitzgerald, J., Heitmeyer, C., Gnesi, S., Philippou, A. (eds.) FM 2016. LNCS, vol. 9995, pp. 85–101. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-48989-6_6
7. Beneš, N., Brim, L., Demko, M., Pastva, S., Šafránek, D.: Pithya: a parallel tool for parameter synthesis of piecewise multi-affine dynamical systems. In: Majumdar, R., Kunčak, V. (eds.) CAV 2017. LNCS, vol. 10426, pp. 591–598. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63387-9_29
8. Bishop, S., Fairbairn, M., Norrish, M., Sewell, P., Smith, M., Wansbrough, K.: Rigorous specification and conformance testing techniques for network protocols, as applied to TCP, UDP, and sockets. SIGCOMM Comput. Commun. Rev. **35**(4), 265–276 (2005)
9. Braden, R.: Requirements for Internet Hosts - Communication Layers. RFC 1122, RFC Editor, October 1989. https://www.rfc-editor.org/rfc/rfc1122.txt
10. Brim, L., Češka, M., Demko, M., Pastva, S., Šafránek, D.: Parameter synthesis by parallel coloured CTL model checking. In: Roux, O., Bourdon, J. (eds.) CMSB 2015. LNCS, vol. 9308, pp. 251–263. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23401-4_21
11. Brim, L., Demko, M., Pastva, S., Šafránek, D.: High-performance discrete bifurcation analysis for piecewise-affine dynamical systems. In: Abate, A., Šafránek, D. (eds.) HSB 2015. LNCS, vol. 9271, pp. 58–74. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-26916-0_4

[1] https://github.com/sybila/tcp-bifurcation.

12. Clark, D.D.: Window and Acknowledgement Strategy in TCP. RFC 813, RFC Editor, July 1982. https://www.rfc-editor.org/rfc/rfc813.txt
13. Fiterău-Broştean, P., Janssen, R., Vaandrager, F.: Combining model learning and model checking to analyze TCP implementations. In: Chaudhuri, S., Farzan, A. (eds.) CAV 2016. LNCS, vol. 9780, pp. 454–471. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-41540-6_25
14. Floyd, S., Jacobson, V.: Random early detection gateways for congestion avoidance. IEEE/ACM Trans. Netw. **1**(4), 397–413 (1993)
15. Gallasch, G.E., Billington, J.: A parametric state space for the analysis of the infinite class of stop-and-wait protocols. In: Valmari, A. (ed.) SPIN 2006. LNCS, vol. 3925, pp. 201–218. Springer, Heidelberg (2006). https://doi.org/10.1007/11691617_12
16. Ghosh, D., Jagannathan, K., Raina, G.: Local stability and Hopf bifurcation analysis for Compound TCP. IEEE Trans. Control Netw. Syst. **5**(4), 1668–1681 (2018). (Early Access)
17. GitHub: TCP flow deadlock: receive window closes and never opens again (2017). https://github.com/mirage/mirage-tcpip/issues/340
18. Han, B., Billington, J.: Termination properties of TCP's connection management procedures. In: Ciardo, G., Darondeau, P. (eds.) ICATPN 2005. LNCS, vol. 3536, pp. 228–249. Springer, Heidelberg (2005). https://doi.org/10.1007/11494744_14
19. Hune, T., Romijn, J., Stoelinga, M., Vaandrager, F.: Linear parametric model checking of timed automata. In: Margaria, T., Yi, W. (eds.) TACAS 2001. LNCS, vol. 2031, pp. 189–203. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45319-9_14
20. Islam, M.A., et al.: Bifurcation analysis of cardiac alternans using δ-decidability. In: Bartocci, E., Lio, P., Paoletti, N. (eds.) CMSB 2016. LNCS, vol. 9859, pp. 132–146. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45177-0_9
21. Islam, M.A., Cleaveland, R., Fenton, F.H., Grosu, R., Jones, P.L., Smolka, S.A.: Probabilistic reachability for multi-parameter bifurcation analysis of cardiac alternans. Theoret. Comput. Sci. (2018, in press)
22. Kuznetsov, Y.A.: Elements of Applied Bifurcation Theory, 2nd edn. Springer, Heidelberg (1998)
23. Lockefeer, L., Williams, D.M., Fokkink, W.: Formal specification and verification of TCP extended with the Window Scale Option. Sci. Comput. Program. **118**, 3–23 (2016)
24. Makino, K., Berz, M.: Verified computations using Taylor models and their applications. In: Abate, A., Boldo, S. (eds.) NSV 2017. LNCS, vol. 10381, pp. 3–13. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63501-9_1
25. Manjunath, S., Raina, G.: FAST TCP: some fluid models, stability and Hopf bifurcation. Perform. Eval. **110**, 48–66 (2017)
26. McLendon III, W., Hendrickson, B., Plimpton, S.J., Rauchwerger, L.: Finding strongly connected components in distributed graphs. J. Parallel Distrib. Comput. **65**(8), 901–910 (2005)
27. Milnor, J.: On the concept of attractor. Commun. Math. Phys. **99**(2), 177–195 (1985)
28. Moldeklev, K., Gunningberg, P.: Deadlock situations in TCP over ATM. In: Neufield, G., Ito, M. (eds.) Protocols for High Speed Networks IV. IAICT, pp. 243–259. Springer, Boston, MA (1995). https://doi.org/10.1007/978-0-387-34885-8_15
29. Nagle, J.: Congestion Control in IP/TCP Internetworks. RFC 896, RFC Editor, January 1984. https://www.rfc-editor.org/rfc/rfc896.txt

30. Nga, J., Iu, H., Ling, B., Lam, H.: Analysis and control of bifurcation and chaos in average queue length in TCP/RED model. Int. J. Bifurc. Chaos **18**(8), 2449–2459 (2008)
31. Postel, J.: Transmission Control Protocol. RFC 793, RFC Editor, September 1981. https://www.rfc-editor.org/rfc/rfc793.txt
32. Raman, S., Mohan, A., Raina, G.: TCP Reno and queue management: local stability and Hopf bifurcation analysis. In: CDC 2013, pp. 3299–3305. IEEE (2013)
33. Ranjan, P., Abed, E.H., La, R.J.: Nonlinear instabilities in TCP-RED. IEEE/ACM Trans. Netw. **12**(6), 1079–1092 (2004)
34. Reif, J.H.: Depth-first search is inherently sequential. Inf. Process. Lett. **20**(5), 229–234 (1985)
35. Schieferdecker, I.: Abruptly terminated connections in TCP - a verification example. In: Brezočnik, Z., Kapus, T. (eds.) Proceedings of the COST 247 International Workshop on Applied Formal Methods in System Design, pp. 136–145. University of Maribor (1996)
36. Smith, M.A., Ramakrishnan, K.K.: Formal specification and verification of safety and performance of TCP selective acknowledgment. IEEE/ACM Trans. Netw. **10**(2), 193–207 (2002)
37. Smith, M.A.S.: Formal verification of communication protocols. In: Gotzhein, R., Bredereke, J. (eds.) Formal Description Techniques IX. IFIPAICT, pp. 129–144. Springer, Boston (1996). https://doi.org/10.1007/978-0-387-35079-0_8
38. Sullivan, D., Williams, R.: On the homology of attractors. Topology **15**(3), 259–262 (1976)
39. Tarjan, R.E.: Depth-first search and linear graph algorithms. SIAM J. Comput. **1**(2), 146–160 (1972)
40. Xu, C., Tang, X., Liao, M.: Local Hopf bifurcation and global existence of periodic solutions in TCP system. Appl. Math. Mech. **31**(6), 775–786 (2010)
41. Xu, C., Li, P.: Dynamical analysis in exponential RED algorithm with communication delay. Adv. Differ. Equ. **2016**(1), 40 (2016)