BFT Selection

Ali Shoker and Jean-Paul Bahsoun

University of Toulouse III, IRIT Lab.
Toulouse, France
`firstName.lastName@irit.fr`

Abstract. One-size-fits-all protocols are hard to achieve in Byzantine fault tolerance (BFT). As an alternative, BFT users, e.g., enterprises, need an easy and efficient method to choose the most convenient protocol that matches their preferences best. The various BFT protocols that have been proposed so far differ significantly in their characteristics and performance which makes choosing the 'preferred' protocol hard. In addition, if the state of the deployed system is too fluctuating, then perhaps using multiple protocols at once is needed; this requires a dynamic selection mechanism to move from one protocol to another. In this paper, we present the first BFT selection model and algorithm that can be used to choose the most convenient protocol according to user preferences. The selection algorithm applies some mathematical formulas to make the selection process easy and automatic. The algorithm operates in three modes: Static, Dynamic, and Heuristic. The Static mode addresses the cases where a single protocol is needed; the Dynamic mode assumes that the system conditions are quite fluctuating and thus requires runtime decisions, and the Heuristic mode is similar to the Dynamic mode but it uses additional heuristics to improve user choices. We give some examples to describe how selection occurs. We show that our approach is automated, easy, and yields reasonable results that match reality. To the best of our knowledge, this is the first work that addresses selection in BFT.

1 Introduction

Byzantine fault tolerance [1–3] (BFT) is a replication-based approach used to maintain the resiliency of services, often state-machines, against *Byzantine* (i.e., arbitrary) faults. A BFT protocol is used to manipulate the communication among system replicas under partial synchrony [4, 2], mainly, using a majority consensus. A typical BFT protocol requires at least $3f+1$ replicas to ensure consistency among system replicas, where up to f replicas can be Byzantine [1, 3, 5] [1].

PBFT [3, 2] is considered the bedrock of practical BFT protocols. It maintains fault tolerance in the presence of Byzantine failures. Unfortunately, the

[1] Some protocols like Q/U [6] require at least $5f+1$; others use trusted components with certain synchrony assumptions to reduce the minimum number of replicas to $2f+1$, e.g., in [7–9].

performance of PBFT is not high. Consequently, the major concern of research community was to boost up the performance of PBFT even if robustness is a bit compromised; thus, they used speculation [10, 6, 11, 12] for this sake. The argument is that stable services are often reliable, but they mostly suffer from transient failures or attacks. Indeed, the perseverance of researchers to improve PBFT yielded many BFT protocols with better reliability and performance (e.g., [3, 10, 6, 11–17], etc); however, this needed some sacrifice in other properties. An upcoming protocol used to resolve some issues of its predecessors and, unfortunately, introduced other new ones. For instance, Q/U [6] and Quorum [12] exhibit the lowest latency and are fault scalable; however, this does not hold true under contention. Zyzzyva [10] and Chain [12] achieve a high throughput, but rather they suffer from expensive recovery. Ring [13] scales to a high number of clients under contention but, in general, it exhibits a low performance. Aardvark [17], Prime [18], and Spinning [16] provide more robustness and tolerance to DoS attacks, but they either remain vulnerable to some attacks, or require more complex infrastructure and resources, etc.

On the other hand, these protocols differ in many characteristics. Table 1 depicts a sample of some properties of existing BFT protocols. In this table, we consider seven different properties (i.e., the rows). Though the choice of these properties is not axial, it gives an intuition about the differences in the characteristics of these protocols. In this table, a property of value 'Yes' (resp., 'No') means that the protocol has (resp., lacks) the corresponding property. The table demonstrates that none of the compared protocols comprises all properties. Intuitively, considering further properties and more protocols yields greater discrepancy among them, and it becomes harder to recommend a single protocol that fits the demands of a BFT user [2].

Therefore, the various BFT protocols and their discrepancy, in terms of both characteristics and performance, leverage important facts. First, designing one-size-fits-all protocols in BFT is very hard, and thus recommending one of them, given BFT users preferences, can be an alternative. Second, it is hard for BFT users, e.g., enterprises, to choose the protocol that suites their demands best

[2] A 'BFT user' in our context is an enterprise that demands a BFT protocol to leverage the resiliency of its deployed services.

	PBFT	Zyzzyva	Q/U	HQ	Quorum	Ring	OBFT	Chain
Non-speculative	**Yes**	No	No	No	No	No	No	No
Tolerates Byzantine clients	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	No	**Yes**
Tolerates contention	**Yes**	**Yes**	No	**Yes**	No	**Yes**	**Yes**	**Yes**
No IP multicast	**Yes**	**Yes**	No	**Yes**	No	**Yes**	**Yes**	**Yes**
Recovery phase	**Yes**	**Yes**	**Yes**	**Yes**	No	**Yes**	**Yes**	No
Obfuscated	No	No	**Yes**	No	**Yes**	No	**Yes**	No
Requires $\leq 3f+1$ replicas	**Yes**	**Yes**	No	**Yes**	**Yes**	**Yes**	No	**Yes**

Table 1: Discrepancy in the properties of a sample state of the art BFT protocols.

(with such number of protocols and their discrepancy), especially, since it is unpractical to expect from a BFT user to acquire knowledge about all (or many) existing protocols; therefore, the selection process should be automated. Finally, a system with fluctuating states (i.e., conditions) requires to move dynamically from one protocol to another as the underlying conditions change. Guerraoui et al. introduced *abortable BFT* in [19, 12] that allows switching between protocols in a static predefined order; however, no switching model or dynamic switching policy was proposed until now.

In this paper, we introduce the first BFT selection model and algorithm that automate and simplify the election process of the 'preferred' BFT protocol among a set of candidate ones. The 'preferred protocol' is the one that matches user preferences the most. An evaluation process is in charge of matching the user preferences against the profiles of the nominated BFT protocols considering both: reliability and performance. The elected protocol is the one that achieves the highest evaluation score. The mechanism is automated via mathematical matrices, and produces selections that are reasonable and close to reality. The selection mechanism operates in three modes: Static, Dynamic, and Heuristic.

The Static mode allows BFT users to choose a single BFT protocol only once. This mode is basically designed for systems that do not have too fluctuating states. A possible application for this mode is to use in clouds, where BFT can be sold as a service (and signed in the SLA contract) along with other services or cloud resources. Some systems, however, may experience fluttering conditions like variable contention or message payloads, etc. In this case, the Static mode will not be a good solution since a chosen protocol might not fit the new conditions. The Dynamic mode solves this issue. It combines a collection of BFT protocols and switches between them, thus, adapting to the changes of the underlying system state. Consequently, the 'preferred' protocol is always polled for each system state. This yields optimal reliability and performance among competing protocols. The Heuristic mode is similar to the Dynamic mode; however, using predefined heuristics, this mode alters/optimizes BFT user (e.g., an enterprise) preferences to improve the selection process. This is useful since BFT users do not have to know about all BFT protocols and their behavior as the underlying conditions change.

We explore in this paper the selection model and algorithm. We describe how the selection mechanism works accompanied with examples. The explanation shows that the model is automated, simple to use, and gives results that are close to reality. We focus on discussing the Static mode, and we describe the Dynamic and Heuristic modes in brief. The latter modes will be demonstrated in more details in another publication due to lack of space (since prediction mechanisms are required in these modes to assess the performance of protocols at runtime).

The rest of the paper is organized as follows. We introduce the selection model and the selection algorithm, mainly the Static mode, in Sections 2 and 3, respectively. The Dynamic and Heuristic modes are described briefly in Sections 4 and 5. Finally, we conclude our paper in Section 6.

2 BFT Selection Model

2.1 Notations and Terms

Before diving into the presentation of the selection model, we introduce some terms and notations that will be used in this model. First, we call any property that specifies the quality of a BFT protocol by Key Quality Indicator (KQI). A KQI is composed of two types of indicators: Key Characteristic indicators (KCI), and Key Performance Indicators (KPI). KCIs are those properties (with boolean values) of a protocol that indicate its behavior, properties, requirements, e.g., number of replicas needed, tolerates malicious clients, obfuscated, etc. The value of this type of indicators can strictly decide whether an evaluated protocol could be selected or not. The KPIs are the properties that evaluate the performance of the protocol like throughput, latency, scalability, etc. These values are usually real numbers. KPIs are usually used to recommend a protocol over the others but, in general, it could not rule out a protocol. In addition, we define the system state by $S = \{s_i = (f_1, f_2, ..., f_j, ..., f_m)\}$ where f_j represents the j^{th} impact factor of the system state and m is the number of considered impact factors by the system. Number of clients, request size, response size are examples of 3 impact factors. The definition and the discussion of concrete measurements of KCIs, KPIs, and impact factors is out of the scope of this paper.

2.2 Selection Model

Consider a service provider (e.g., a cloud vendor) that offers n different BFT protocols along with its provided services (e.g., signed in SLA contract). We define the set of protocols $\psi = \{p_i;\ where\ 1 \leq i \leq n\}$; where p_i is one of the BFT protocols. On the other hand, consider a selection model represented by:

$$\Sigma = \{\text{PROTOCOL,USER,MODE}\} \tag{1}$$

where PROTOCOL represents the profile of a BFT protocol, USER represents the preferences of the user, and MODE represents the selection mode of the system. Selection occurs through matching the PROTOCOL profile with the USER preferences according to the mapping: $f : \Sigma \longmapsto \psi$; this yields the 'preferred' protocol among all competing protocols. Here we introduce the definition of the 'preferred' protocol:

Definition 1. *A protocol p_i with profile $PROTOCOL_i$ is called the 'preferred' protocol among a set of available protocols ψ with respect to a specific user with preferences $USER_j$ if and only if according to an evaluation function $e : \Sigma \longmapsto \Re$, e($PROTOCOL_i$, $USER_j$,ϕ) is maximal.*

The interpretation of PROTOCOL, USER, and MODE is as follows:

PROTOCOL. Each protocol is described in a profile of KQIs and default weights: PROTOCOL=$\{A_P, A_U, B_P, B_V\}$, where A_P is a vector of a KCIs:

$A_P = (\alpha_1, \alpha_2, ..., \alpha_a)$, and A_U represents the vector of the corresponding default weights of these KCIs: $A_U = (u_1, u_2, ..., u_a)$. B_P, however, is a vector of b KPIs: $B_P = (\beta_1, \beta_2, ..., \beta_b)$, and B_V represents the vector of the corresponding default weights of these KPIs: $B_V = (v_1, v_2, ..., v_b)$.

USER. Each user defines his preferences in USER=$\{U, V, M\}$, where U (resp., V) is a vector of user defined weights corresponding to the KCIs (resp., KPIs) of the PROTOCOL's preference A_P (resp., B_P). M defines the mode required by the user, i.e, either Dynamic, Static, or Heuristic.

MODE. The selection can occur in three different modes: Static, Dynamic, or Heuristic. In the former, the selection occurs only once, i.e., at the time the user requires a service; afterwards, the user does not change his selection (i.e., the used protocol) until the system is halted/rebooted and, thus a new selection is provoked. On the other hand, the Dynamic mode makes the system react dynamically to the changes of the system state. This mode allows the system to adapt to the upcoming conditions at runtime and hence the user will be using multiple protocols at once. The Heuristic mode is similar to the Dynamic mode but, in addition, it uses some heuristics to adjust the preferences of the user, especially V, to improve his choices in some cases. The heuristics are represented in a vector W similar to V, but its values are modified as the system state changes according to some predefined heuristics. The default mode is Static.

In the Static mode, A_P and B_P are predefined. They are calculated before selection occurs, and even before the advertisement of the protocol is done. However, in the Dynamic and Heuristic modes, the values of B_P change dynamically while the system is running (e.g., using runtime predictions). When the impact factors of the system change, a new dynamic evaluation and selection phase is needed to move, probably, to a new protocol that performs better than the current one under the new conditions. Due to the size limitations of this paper, we explain the Static mode in details, and we address the other modes in a future paper.

3 Selection Mechanism

3.1 Overview

The selection mechanism of the most convenient protocol according to the user preferences, i.e., the preferred protocol, is achieved through computing the evaluation score E of the competing protocols, and then electing the protocol that corresponds to the maximum score.

Selection (and evaluation) occurs only once in the Static mode, i.e., at deployment time. Otherwise, the deployed service should be stopped and restarted again, probably, using another selected protocol if the user preferences have changed. For any state s, and protocol $p_i \in \psi$ that has an evaluation score $E_{i,s}$; a protocol p_{pref} is chosen according to Equation 2:

$$p_{pref} = p_i, \ s.t. \ E_{i,s} = \max_{1 \leq j \leq n} E_{j,s}. \qquad (2)$$

If the mode of the system is Dynamic or Heuristic, the evaluation process of protocols can take place at any instant in a dynamic way. The KPIs are computed at runtime, and the system chooses the protocol that has the highest evaluation score E among all protocols to launch it in the next phase, only if it is worthy to switch (see more details in [20]).

In addition, to make computations easier, we define a new operator, i.e., the OR product $\dot{\vee}$.

Definition 1 *Consider two boolean matrices $A \in \{0,1\}^{n \times l}, B \in \{0,1\}^{l \times m}$ with entries a_{ij}, and b_{ij}, respectively. The OR product $A \dot{\vee} B$ is a matrix $C = A \dot{\vee} B \in \mathbb{N}^{n \times m}$, where its elements are defined by: $c_{ij} = \sum_{k=1}^{m} a_{ik} \vee b_{kj}$. The operator $\vee$ is the logical OR operator.*

3.2 Evaluation of Protocols

As mentioned above, the selection of the preferred protocol is represented by the mapping: $f : \Sigma \longmapsto \psi$; where, $f = p_{pref}$ in Equation 2 is an example of such mapping. The evaluation score E is calculated according to the formulas introduced in Equation 3. In the following, we interpret and discuss these formulas. To make the idea easier to understand, we conduct some concrete examples along with the discussion.

$$\begin{cases} E = C \circ P \\ where\ C = \left\lfloor \dfrac{1}{a}.\left(A \ \dot{\vee} \ (e_n - U)\right) \right\rfloor \\ and\ P = B^{\pm}.(V \circ W). \end{cases} \tag{3}$$

3.2.1 The Evaluation Matrix *E*

The evaluation matrix E is the Schur product of the KCI matrix C and the KPI matrix P. C represents the part of the evaluation that deals with the KCIs of the profiles of the protocols; whereas, P represents the evaluation part that deals with the KPIs. E is calculated after computing the values of C and P. If the mode of the system is Dynamic or Heuristic, then E may change at runtime as P changes.

3.2.2 The KCI Matrix *C*

The KCI matrix $C = \left\lfloor \frac{1}{a}.\left(A \ \dot{\vee} \ (e_n - U)\right) \right\rfloor$ matches the user preferences against the profiles of different protocols. a represents the number of KCIs considered. The operator $\lfloor \ \rfloor$ is the absolute value operator (it is sometimes indicated by [ ] too). The operator $\dot{\vee}$ was defined in Definition 1. The matrices A, U, and e_n are explained next.

Matrix *A*. This matrix represents the profiles of the protocols. The dimension of A is $n \times a$; where n is the number of candidate protocols and a is the number of KCIs considered in the evaluation. Each row of the matrix represents a KCI vector profile of a protocol. For instance, consider the KCIs of the BFT protocols depicted in Table 1. If a protocol has a certain property, i.e., has the value 'Yes', it takes the value 1; otherwise, it lacks that property and thus takes the value 0. Example 1 presents a sample matrix A. Each column (i.e., protocol) in Table 1 represents a row in the matrix A. Note that, the values in the table are not necessarily 100% accurate, but they are acceptable to conduct our examples. Deciding the values of KCIs is out of the scope of this paper.

Matrix *U*. This is a vector matrix that represents the preferences of the user. According to this matrix, the protocols that satisfy all user requirements will be considered for selection (i.e., will continue the competition). On the contrary, the protocol that lacks a single property among those demanded by the user will be out of selection. If the user did not define his preferences in U, the default replacement A_U provided by the system will be used (see Section 2.2). U is a 1-column matrix (of dimension $a \times 1$). In Example 1, we present a sample matrix U. The values of this matrix correspond to the KCIs (Non-speculative, Byzantine clients, Tolerates, Contention, No IP Multicast, Self-Recovery, Obfuscated, Requires $\leq 3f+1$). This matrix tells that the user needs a protocol that tolerates Byzantine clients and requires less than $3f+1$ replicas, and that he does not care for the other properties.

Note that, if the user chooses the vector $U = e_a$, then any discrete matrix has no effect on the selection, and the protocols are only selected regarding the KPI matrix P. On the contrary, if $U = 0_a$, then no protocol will be selected unless it satisfies every property. In our case, none of the protocols satisfy this condition, and thus none will be selected.

Matrix e_n. This is a column matrix of dimension $a \times 1$. We use this matrix is to invert the values of the matrix U to $-U$.

Computing *C*. After defining the matrices A and U, the computation of C becomes straightforward. Example 1 describes how computation occurs. Through analyzing Example 1, important notes can be drawn. First, notice that the computation is automated using these formulas. Imagine the time and effort needed to achieve the result of the matrix C by hand, especially when larger problem scale is considered. Second, the selection depends, strictly, on the user preferences and, thus, important protocols (e.g., Q/U and OBFT) can be evicted out of competition if the user preferences are not matched.

Example 1

$$A = \begin{pmatrix} 1\,1\,1\,1\,1\,0\,1 \\ 0\,1\,1\,1\,1\,0\,1 \\ 0\,1\,0\,0\,1\,1\,0 \\ 0\,1\,1\,1\,1\,0\,1 \\ 0\,1\,0\,0\,0\,1\,1 \\ 0\,1\,1\,1\,1\,0\,1 \\ 0\,0\,1\,1\,1\,1\,0 \\ 0\,1\,1\,1\,0\,0\,1 \end{pmatrix} \begin{array}{l} \longleftarrow \; PBFT \\ \longleftarrow \; Zlight \\ \longleftarrow \; Q/U \\ \longleftarrow \; HQ \\ \longleftarrow \; Quorum \\ \longleftarrow \; Ring \\ \longleftarrow \; OBFT \\ \longleftarrow \; Chain \end{array} ; and\ U = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \begin{array}{l} \longleftarrow \; Non-speculative \\ \longleftarrow \; Tolerates Byzantine clients \\ \longleftarrow \; Tolerates contention \\ \longleftarrow \; No IP multicast \\ \longleftarrow \; Recovery phase \\ \longleftarrow \; Obfuscated \\ \longleftarrow \; Requires \leq 3f+1 replicas \end{array}$$

$$D = \left\lfloor \frac{1}{a} . \left(A \dot{\vee} (e_a - U)\right)\right\rfloor$$

$$= \left\lfloor \frac{1}{7} . \left(\begin{pmatrix} 1\,1\,1\,1\,1\,0\,1 \\ 0\,1\,1\,1\,1\,0\,1 \\ 0\,1\,0\,0\,1\,1\,0 \\ 0\,1\,1\,1\,1\,0\,1 \\ 0\,1\,0\,0\,0\,1\,1 \\ 0\,1\,1\,1\,1\,0\,1 \\ 0\,0\,1\,1\,1\,1\,0 \\ 0\,1\,1\,1\,0\,0\,1 \end{pmatrix} \dot{\vee} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} - \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \right) \right\rfloor = \left\lfloor \frac{1}{7} . \begin{pmatrix} 7 \\ 7 \\ 6 \\ 7 \\ 7 \\ 7 \\ 5 \\ 7 \end{pmatrix} \right\rfloor = \left\lfloor \begin{pmatrix} 1 \\ 1 \\ 6/7 \\ 1 \\ 1 \\ 1 \\ 5/7 \\ 1 \end{pmatrix} \right\rfloor = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

3.2.3 The KPI Matrix *P*

This matrix is used to complete the selection process by considering the KPIs of the protocols, seeking better performance. The KPI matrix P is defined in the formula: $P = B^{\pm}.(V \circ W)$. The matrices B, V, and W are discussed in the following.

Matrices B and $B^{\pm}$. B represents the KPI profiles of each protocol. Each profile is presented in one row. $B^{\pm}$ is a normalized version of B. B and $B^{\pm}$ have the same dimension $n \times b$ where n is the number of protocols and b is the number of KPIs considered. The entries of the matrix $B^{\pm}$ are denoted by $\beta^{\pm}$ and are calculated from the entries of B that are denoted by β.

The notion of $\beta^{\pm}$ KPIs. The $\beta^{\pm}$ values of matrix $B^{\pm}$ can be either β^{+}, or β^{-}; they are normalized values of the entries of B (i.e., they belong to the interval [0,1]). We say that a KPI has the property Tendency='high', if a higher value means better evaluation score E, e.g., throughput; this KPI is denoted by β^{+}. On the contrary, a KPI of type β^{-} has the property Tendency='low', e.g., latency, and a higher KPI value means worst evaluation score E.

Suppose the number of β-KPIs is b, then the matrix B can be divided into b column matrices (i.e., vectors): $B_1, B_2, B_i, ...,$ and B_b. Let the maximum (resp., minimum) value of the entries of each vector B_i be max_i (resp., min_i). Then, the entries of the matrix $B^{\pm}$ can be calculated as follows:

$$\begin{cases} \beta^{+}_{ji} = 1 - \dfrac{max_i - \beta_{ji}}{max_i - min_i}; \\ \beta^{-}_{ji} = 1 - \dfrac{\beta_{ji} - min_i}{max_i - min_i}; \\ where\ i \leq b\ and\ j \leq n. \end{cases} \quad (4)$$

Table 2 conveys an analytic evaluation of the studied protocols. Based on this table, we obtain the KPI values β of Table 3 considering three KPIs: Throughput, Latency, and Capacity (i.e., the estimated number of clients that can be tolerated). The values of Table 3 are calculated as follows: Throughput=10/C, and Latency=D. We use 10/C (and not 1/C) in throughput to show the effect of normalization in the next sections. As for Capacity, we estimate this value based on the message patterns of the protocols and some experiments we conducted

	PBFT	Zyzzyva	Q/U	HQ	Quorum	Ring	OBFT	Chain
A	**4**	**4**	6	**4**	**4**	**4**	**4**	**4**
B	10	5	6	6	**2**	≈8	**2**	3
C	4	3	**2**	4	**2**	≈9	4	5
D	3	2	**1**	3	**1**	2	**1**	**1**

Table 2: Analytic evaluation for the state of the art BFT protocols tolerating f faults using MACs for authentication, and assuming preferred optimization (without batching): A represents the number of replicas needed to tolerate f Byzantine replicas; B represents the number of MAC operations on the bottleneck replica; C is the number of one-way latencies needed for each request; and D represents the number of send/to kernel calls on the bottleneck replica. Bold entries denote protocols with the lowest known cost.

	PBFT	Zyzzyva	Q/U	HQ	Quorum	Ring	OBFT	Chain
Throughput (β^+)	**1**	2	1.67	1.67	**5**	1.25	**5**	3.33
Latency (β^-)	4	3	**2**	4	**2**	**9**	4	5
Capacity (β^+)	6	7	2	5	**1**	**10**	7	8

Table 3: Analytical values of β^+ and β^- for the state of the art BFT protocols.

for this sake. Note that the values of Table 3 can be inaccurate; however, our experience shows that they give a good approximation that is enough to explain our idea. Filling this table is out of the scope of this paper, we rather keep it for future work.

Next, we explain the calculation of $B^{\pm}$ from B in Example 2. Example 3 presents the two matrices B and $B^{\pm}$. Each column in Table 3 represents a row in the matrix B.

Example 2 *The matrix B can be divided into 3 column matrices B_1, B_2, and B_3. The corresponding maxima and minima of these column matrices are as follows: (max_1=5,min_1=1), (max_2=9, min_2=2), (max_3=10,min_3=1). The calculation of $\beta^{\pm}$ is done according to the formulas in Equation 4. For instance, the $\beta^{\pm}$ KPIs of PBFT are as follows:*

- *the throughput KPI (β^+) of PBFT is: β^+_{11}=$(1 - \frac{5-1}{5-1})$=0.*
- *the latency KPI (β^-) of PBFT is: β^-_{12}=$(1 - \frac{4-2}{9-2})$=0.71.*
- *the capacity KPI (β^+) of PBFT is: β^+_{13}=$(1 - \frac{10-6}{10-1})$=0.55.*

The $\beta^{\pm}$ values of the other protocols (i.e., other rows) are calculated in a similar way, and finally we obtain the matrix $B^{\pm}$ in Example 3.

Matrix *V*. This matrix represents the KPI user preferences used to recommend a protocol. V is a column matrix of dimension $b \times 1$, where b is the number of KPIs considered in the evaluation. The entries of this matrix follow two constraints: (1) all entries $\in$ [0,10], and (2) their sum $\sum_{i=1}^{b} v_{i1} = 10$. Notice that, this differs from the preferences of the discrete matrix U, since the values of $U \in \{0, 1\}$. As an example of V, we suppose that the requirements of the user are default, i.e., a protocol with acceptable throughput, latency, and capacity (i.e., tolerates a high number of clients); thus the matrix becomes V=(3,3,4) as depicted in Example 3.

Matrix *W*. This matrix is a column matrix used in the Heuristic mode only. W is important to adjust the user preferences given in V by considering the system state to improve his choice. Thus, it can happen that the user chooses a 'low' weight for some KPI in V; however, according to the heuristics of the system, the weight is modified by the corresponding value in W to 'high'. In some cases, this improves the performance of the system significantly. W is a column matrix with the same dimension as V and its entries follow the same constraints. Since we assume that the default mode is Static, then the entries of W are equal to 1, i.e., W=e_b, and the matrix has no effect on the evaluation. We leave the discussion of W to Section 5.

Computing *P*. After establishing the matrices $B^{\pm}$, V, and W, the calculation of P becomes straightforward as described in Example 3. The results show the evaluation of the performance of the protocols by considering the three KPIs throughput, latency, and capacity together. The result is reasonable as it conveys that OBFT, Zyzzyva, and Chain achieve the best scores according to the current user preferences in V (recall that OBFT does not tolerate Byzantine clients). The final evaluation of E, however, yields a different result. The reason is that the KCI preferences of the user have to be taken into consideration.

Example 3

$$B^{\pm} = \begin{pmatrix} 0 & 0.71 & 0.55 \\ 0.25 & 0.86 & 0.67 \\ 0.17 & 1 & 0.11 \\ 0.17 & 0.71 & 0.44 \\ 1 & 1 & 0 \\ 0.06 & 0 & 1 \\ 1 & 0.71 & 0.67 \\ 0.58 & 0.57 & 0.78 \end{pmatrix}; B^{\pm} = \begin{pmatrix} 0 & 0.71 & 0.55 \\ 0.25 & 0.86 & 0.67 \\ 0.17 & 1 & 0.11 \\ 0.17 & 0.71 & 0.44 \\ 1 & 1 & 0 \\ 0.06 & 0 & 1 \\ 1 & 0.71 & 0.67 \\ 0.58 & 0.57 & 0.89 \end{pmatrix}; V = \begin{pmatrix} 3 \\ 3 \\ 4 \end{pmatrix} \begin{array}{l} \longleftarrow \textit{Throughput} \\ \longleftarrow \textit{Latency} \\ \longleftarrow \textit{Capacity} \end{array}$$

$$P = B^{\pm}.(V \circ W)$$

$$= \begin{pmatrix} 0 & 0.71 & 0.55 \\ 0.25 & 0.86 & 0.67 \\ 0.17 & 1 & 0.11 \\ 0.17 & 0.71 & 0.44 \\ 1 & 1 & 0 \\ 0.06 & 0 & 1 \\ 1 & 0.71 & 0.67 \\ 0.58 & 0.57 & 0.89 \end{pmatrix} . \begin{pmatrix} 3 \\ 3 \\ 4 \end{pmatrix} \circ \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 & 0.71 & 0.55 \\ 0.25 & 0.86 & 0.67 \\ 0.17 & 1 & 0.11 \\ 0.17 & 0.71 & 0.44 \\ 1 & 1 & 0 \\ 0.06 & 0 & 1 \\ 1 & 0.71 & 0.67 \\ 0.58 & 0.57 & 0.89 \end{pmatrix} . \begin{pmatrix} 3 \\ 3 \\ 4 \end{pmatrix} = \begin{pmatrix} 4.33 \\ 6.01 \\ 3.95 \\ 4.4 \\ 6 \\ 4.18 \\ 8.25 \\ 6.57 \end{pmatrix}$$

3.3 The Preferred Protocol

To achieve the preferred protocol, the evaluation scores of matrix E should be computed. This represents the overall evaluation of KCIs and KPIs together.

After calculating E, the 'preferred' protocol with respect to the user can be selected using the Equation 2. Example 4 shows the calculation of E. For the system state $s = Current$, and considering the matrices C and P computed above, $E_{8,Current} = max(E)$=6.57, and then p_{pref}=p_1 =Chain. Therfore, Chain is chosen as preferred protocol to this user. This is expected since Chain achieves the a highest throughput among other protocols as it requires the minimal number of MAC operations. The Dynamic mode shows that this selection is not always true in reality.

Example 4 $E = C \circ P = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix} \circ \begin{pmatrix} 4.33 \\ 6.01 \\ 3.95 \\ 4.4 \\ 6 \\ 4.18 \\ 8.25 \\ 6.57 \end{pmatrix} = \begin{pmatrix} 4.33 \\ 6.01 \\ 0 \\ 4.4 \\ 6 \\ 4.18 \\ 0 \\ 6.57 \end{pmatrix}$

4 Dynamic Mode

The Dynamic mode is designed for systems that exhibit fluctuating states, and thus the system needs to adapt to the new state, dynamically. In this mode, evaluation and selection can occur more than once, i.e., during the normal operation of the system. The motivation behind the Dynamic mode can be noticed clearly through observing the values of KPIs in the matrix B in Examples 3 and 4, where Chain has been chosen as a preferred protocol. Indeed, it is known that Chain achieves low throughput with few clients due to its long messaging pattern [12]. Thus, in case the system is supposed to low contention for a reasonable period of time, it is more efficient to switch to another protocol. In the Static mode, Chain could not be replaced by another protocol according to matrix E that is statically assessed using Table 2. However, in the Dynamic mode, the KPIs in the matrix B are modified at runtime which may yield another protocol that has a better performance.

Due to lack of space, we explain the Dynamic mode briefly in this paper, and we conduct a complete study in a future paper. The idea is to assess the values of B on the fly. Our method uses the famous Support Vector Machines for Regression [21] (SVR), a technique often used in Machine Learning. This mechanism allows the prediction of the KPIs of the protocols as the system is running. This technique leads to very accurate (up to 98%) KPI values which is not the case in our current examples (as shown in Table 3). For example, our experiements show that the throughput of Zyzzyva is very close to Chain in real systems; however in our examples, the β values of Zyzzyva in Table 3 (and in matrix B) are lower than Chain.

5 Heuristic Mode

The Heuristic mode is an advanced Dynamic mode. This mode uses some heuristics to adjust the user preferences of the KPIs. This is very useful in the cases where the user has no big experience or the system state is too fluctuating. Similar to the Dynamic mode, the evaluation of protocols is done at runtime and, probably, a new protocol is selected as the system state changes. Thus, the KPIs (i.e., the entries of matrix B) are calculated dynamically during the normal operation of the system.

However, in the Dynamic mode, it can happen that according to the chosen KPI weights of the user (i.e., the entries of matrix V), a certain protocol is recommended; however, under some specific conditions, other protocols can perform better (thus, the choice is not the best). For instance, if the system is deprived from contention, the latency becomes more important than throughput and capacity, even if the user has given higher weights for these values. Another example is when the system is jammed with requests from numerous clients. In this case, the system should run the protocol that tolerates more clients even if the client haas given low weights to the capacity KPI. The role of the heuristics is to adjust the user preferences through using the weights (in matrix W) as the system state changes. These weights can modify the weights already defined by the user in V, but just temporarily, according to the underlying system conditions. To explain the idea, we consider the two heuristic rules:

– Heuristic 1: In contention-free cases, Latency is more important than Throughput and Capacity.
– Heuristic 2: Under contention, Capacity is more important than Throughput and Latency.

In the case where the system is deprived from contention, Heuristic 1 can be used, and is expressed in matrix W_1. Heuristic 2 can be used if high contention is imposed on the system, and is expressed in W_2. Example 5 explores the power of the Heuristic mode. Notice that, though the user requirements are almost equivalent for all KPIs (i.e., throughput, latency, and capacity), the heuristics can modify the weights defined in V through multiplying it by W. Consequently, the preferred protocol can change according to these heuristics. Example 5 shows that the preferred protocol in contention-free cases is Quorum which is very reasonable as it exhibits the shortest messaging pattern among all protocols [12]. However, the preferred protocol under high contention becomes Chain. This makes sense as Chain is deprived from messages interference under contention due to its chain-messaging fashion [12].

$$W_1 = \begin{pmatrix} 1 \\ 8 \\ 1 \end{pmatrix} ;\ and\ W_2 = \begin{pmatrix} 2 \\ 1 \\ 7 \end{pmatrix}$$

Example 5

$$P_1 = \begin{pmatrix} 0 & 0.71 & 0.55 \\ 0.25 & 0.86 & 0.67 \\ 0.17 & 1 & 0.11 \\ 0.17 & 0.71 & 0.44 \\ 1 & 1 & 0 \\ 0.06 & 0 & 1 \\ 1 & 0.71 & 0.67 \\ 0.58 & 0.57 & 0.89 \end{pmatrix} \cdot \begin{pmatrix} 3 \\ 3 \\ 4 \end{pmatrix} \circ \begin{pmatrix} 1 \\ 8 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 & 0.71 & 0.55 \\ 0.25 & 0.86 & 0.67 \\ 0.17 & 1 & 0.11 \\ 0.17 & 0.71 & 0.44 \\ 1 & 1 & 0 \\ 0.06 & 0 & 1 \\ 1 & 0.71 & 0.67 \\ 0.58 & 0.57 & 0.89 \end{pmatrix} \cdot \begin{pmatrix} 3 \\ 24 \\ 4 \end{pmatrix} = \begin{pmatrix} 19.24 \\ 24.07 \\ 24.95 \\ 19.31 \\ 27 \\ 4.18 \\ 22.72 \\ 18.98 \end{pmatrix}$$

$$P_2 = \begin{pmatrix} 0 & 0.71 & 0.55 \\ 0.25 & 0.86 & 0.67 \\ 0.17 & 1 & 0.11 \\ 0.17 & 0.71 & 0.44 \\ 1 & 1 & 0 \\ 0.06 & 0 & 1 \\ 1 & 0.71 & 0.67 \\ 0.58 & 0.57 & 0.89 \end{pmatrix} \cdot \begin{pmatrix} 3 \\ 3 \\ 4 \end{pmatrix} \circ \begin{pmatrix} 2 \\ 1 \\ 7 \end{pmatrix} = \begin{pmatrix} 0 & 0.71 & 0.55 \\ 0.25 & 0.86 & 0.67 \\ 0.17 & 1 & 0.11 \\ 0.17 & 0.71 & 0.44 \\ 1 & 1 & 0 \\ 0.06 & 0 & 1 \\ 1 & 0.71 & 0.67 \\ 0.58 & 0.57 & 0.89 \end{pmatrix} \cdot \begin{pmatrix} 6 \\ 3 \\ 28 \end{pmatrix} = \begin{pmatrix} 17.53 \\ 22.84 \\ 7.1 \\ 15.47 \\ 9 \\ 28.36 \\ 26.89 \\ 30.11 \end{pmatrix}$$

$$E_1 = C \circ P = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix} \circ \begin{pmatrix} 19.24 \\ 24.07 \\ 24.95 \\ 19.31 \\ 27 \\ 4.18 \\ 22.72 \\ 18.98 \end{pmatrix} = \begin{pmatrix} 19.24 \\ 24.07 \\ 0 \\ 19.31 \\ 27 \\ 4.18 \\ 0 \\ 18.98 \end{pmatrix} ; E_2 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix} \circ \begin{pmatrix} 17.53 \\ 22.84 \\ 7.1 \\ 15.47 \\ 9 \\ 28.36 \\ 26.89 \\ 30.11 \end{pmatrix} = \begin{pmatrix} 17.53 \\ 22.84 \\ 0 \\ 15.47 \\ 9 \\ 28.36 \\ 0 \\ 30.11 \end{pmatrix}$$

The above examples are conducted without changing the values of KPIs in matrix B (i.e., as if no Dynamic mode is chosen). Indeed, this makes the result a bit biased. For example, under high contention, Ring achieves a higher throughput than other protocols. However, the matrix B uses the throughput 0.06 of Ring, which is very different from reality. The issue arises here since we estimated the throughput of Ring according to the number of MAC authentications needed which could not be true as the system state changes. This indicates that when the KPIs in B are calculated dynamically, the results can be more accurate and reasonable. We address this issue in a future paper due to size limitations.

6 Conclusion

We presented a BFT selection model and algorithm to improve the quality of BFT services. The mechanism helps the user to choose the 'preferred' BFT protocol according his preferences. This is useful in large services that provide BFT as a service. For instance, cloud vendors can sell BFT services along with their IaaS, PaaS, and AaaS SLA contracts. This can be achieved by proposing our selection mechanism for the BFT user (e.g., an enterprise), and this user is in charge of choosing his preferred protocol using this mechanism. This can be implemented as a Web service.

Our mechanism tries to match the user preferences with the profiles of each protocol through a set of mathematical formulas based on matrices. The formulas are automated to choose the preferred protocol with respect to the demanding user. Our model considers three modes: (1) Static mode: this is the default mode where the user chooses a protocol only once; he can only change it when the service is rebooted. (2) Dynamic mode: which allows the user to use more than one protocol at once, where a running protocol can be stopped and another protocol is launched after performing the evaluation and selection process. The intuition is that the performance of protocols differ as the underlying system state changes, and thus adapting to the new state is required. In this mode, the performance measures (i.e., the KPIs) are calculated at runtime. (3) Heuristic mode: this mode is similar to the Dynamic mode; however, it allows to modify the weights (i.e., preferences) chosen by the user as the system state changes using some predefined heuristics. This mode is very useful since some KPIs are important under some conditions, but they become less important in other situations. For example, throughput is very important in normal cases, however, it is not basic in contention-free cases (in contrast to latency); thus, adjusting the weights of the KPIs will have great impact on the overall performance.

This work introduces many open problems for future work. First, defining a set of the most significant KCIs would be interesting. Second, establishing an evaluation method to evaluate the Static mode is required. Third, in the Dynamic mode, a prediction mechanism to assess the performance of protocols should be established. Fourth, defining the impact factors that affect the performance of protocols is crucial. Fifth, developing an event system that monitors the system and sends events to the prediction system upon potential changes in the underlying system is needed. Finally, defining a set of heuristics to be used in the Heuristic mode would be interesting.

References

1. Lamport, L., Shostak, R., Pease, M.: The byzantine generals problem. ACM Transactions on Programming Languages and Systems **4** (1982) 382–401
2. Castro, M.: Practical byzantine fault tolerance. In: Ph.D Thesis. (2001) 173–186
3. Castro, M., Liskov, B.: Practical byzantine fault tolerance and proactive recovery. ACM Trans. Comput. Syst. **20**(4) (2002) 398–461
4. Dwork, C., Lynch, N., Stockmeyer, L.: Consensus in the presence of partial synchrony. J. ACM **35**(2) (April 1988) 288–323
5. Bracha, G., Toueg, S.: Asynchronous consensus and broadcast protocols. Journal of the ACM (JACM) **32**(4) (1985) 824–840
6. Abd-El-Malek, M., Ganger, G.R., Goodson, G.R., Reiter, M.K., Wylie, J.J.: Fault-scalable byzantine fault-tolerant services. SIGOPS Oper. Syst. Rev. **39**(5) (2005) 59–74
7. Correia, M., Neves, N.F., Verssimo, N.P.: How to tolerate half less one byzantine nodes in practical distributed systems. In: IN PROCEEDINGS OF THE 23RD IEEE SYMPOSIUM ON RELIABLE DISTRIBUTED SYSTEMS. (2004) 174–183
8. Chun, B.G., Maniatis, P., Shenker, S., Kubiatowicz, J.: Attested append-only memory: making adversaries stick to their word. In: Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles. SOSP '07, New York, NY, USA, ACM (2007) 189–204

9. Veronese, G.S., Correia, M., Bessani, A.N., Lung, L.C.: Ebawa: Efficient byzantine agreement for wide-area networks (2010)
10. Kotla, R., Alvisi, L., Dahlin, M., Clement, A., Wong, E.: Zyzzyva: speculative byzantine fault tolerance. SIGOPS Oper. Syst. Rev. **41**(6) (2007) 45–58
11. Cowling, J., Myers, D., Liskov, B., Rodrigues, R., Shrira, L.: Hq replication: a hybrid quorum protocol for byzantine fault tolerance. In: OSDI '06: Proceedings of the 7th symposium on Operating systems design and implementation, Berkeley, CA, USA, USENIX Association (2006) 177–190
12. Guerraoui, R., Knežević, N., Quéma, V., Vukolić, M.: The next 700 bft protocols. In: EuroSys '10: Proceedings of the 5th European conference on Computer systems, New York, NY, USA, ACM (2010) 363–376
13. Rachid Guerraoui, Nikola Knezevic, Vivien Quema, Marco Vukolic: Stretching bft. Technical Report EPFL-REPORT-149105, EPFL (2011)
14. Guerraoui, R., Yabandeh, M., Shoker, A., Bahsoun, J.P.: Obfuscating bft. Technical Report LPD-REPORT-2011-5, EPFL (2011)
15. Ali Shoker and Jean-Paul Bahsoun: Recover to Self: Re-Abstract Family (regular paper). In: The International Conference on Computer and Management. CAMAN, IEEE (March 2012)
16. Veronese, G.S., Correia, M., Bessani, A.N., Lung, L.C.: Spin one's wheels? byzantine fault tolerance with a spinning primary. In: Proceedings of the 2009 28th IEEE International Symposium on Reliable Distributed Systems. SRDS '09, IEEE Computer Society (2009)
17. Clement, A., Wong, E., Alvisi, L., Dahlin, M., Marchetti, M.: Making byzantine fault tolerant systems tolerate byzantine faults. In: NSDI'09: Proceedings of the 6th USENIX symposium on Networked systems design and implementation, Berkeley, CA, USA, USENIX Association (2009) 153–168
18. Amir, Y., Coan, B., Kirsch, J., Lane, J.: Prime: Byzantine replication under attack. IEEE Trans. Dependable Secur. Comput. **8**(4) (July 2011) 564–577
19. Guerraoui, R., Knežević, N., Quéma, V., Vukolić, M.: The next 700 bft protocol. Technical Report LPD-REPORT-2008-008, EPFL (2008)
20. Shoker, A.: Byzantine fault tolerance: From static selection to dynamic switching. In: Ph.D Thesis. (December 2012)
21. Smola, Alex J. and Schölkopf, Bernhard: A tutorial on support vector regression. Statistics and Computing **14** (2004) 199–222