Byzantine Stable Matching

Andrei Constantinescu
ETH Zurich
Zurich, Switzerland
aconstantine@ethz.ch

Marc Dufay
ETH Zurich
Zurich, Switzerland
mdufay@ethz.ch

Diana Ghinea*
Lucerne University of Applied Sciences and Arts
Zug, Switzerland
diana.ghinea@hslu.ch

Roger Wattenhofer
ETH Zurich
Zurich, Switzerland
wattenhofer@ethz.ch

Abstract

In stable matching, one must find a matching between two sets of agents, commonly men and women, or job applicants and job positions. Each agent has a preference ordering over who they want to be matched with. Moreover a matching is said to be stable if no pair of agents prefer each other over their current matching.

We consider solving stable matching in a distributed synchronous setting, where each agent is its own process. Moreover, we assume up to t_L agents on one side and t_R on the other side can be byzantine. After properly defining the stable matching problem in this setting, we study its solvability.

When there are as many agents on each side with fully-ordered preference lists, we give necessary and sufficient conditions for stable matching to be solvable in the synchronous setting. These conditions depend on the communication model used, i.e., if parties on the same side are allowed to communicate directly, and on the presence of a cryptographic setup, i.e., digital signatures.

CCS Concepts

• **Theory of computation → Distributed computing models**; **Cryptographic protocols**.

Keywords

Stable matching, Byzantine faults, Distributed protocol

Related Version: A full version of this paper is available at [6].

---

1 Introduction

The *stable matching* problem (also known as *stable marriage*), first introduced by Gale and Shapley [11], has long been a cornerstone of combinatorial optimization and market design.

It involves finding a stable pairing between two distinct sets of agents — such as job seekers and job positions or students and universities — where each participant ranks the opposite set based on individual preferences. The stability criterion dictates that there is no *blocking pair*, i.e., no two unmatched agents should prefer each other over their assigned partners. This foundational task has multiple practical applications in resource assignment and subsequently led to Shapley and Roth winning the Nobel Memorial Prize in Economics in 2012 for their work on *the theory of stable allocations and the practice of market design* [1].

In their seminal work [11], Gale and Shapley proved that when the *n* agents are divided equally into the two sides, and each individual provides a complete preference ranking of the opposite set, a stable matching always exists. Moreover, they provided an algorithm finding such a matching with complexity $O(n^2)$. Further versions of this problem have been considered in [14], including variants where the individuals only provide *partial* preferences, or if ties are allowed within the preference rankings. The work of [14] has shown that a stable matching always exists even in such scenarios, although some individuals may not be matched.

The stable matching problem naturally extends to distributed settings, where each agent operates as an independent process or party. Through communication, agents determine their matches while ensuring stability — a *local* property. Furthermore, the Gale-Shapley algorithm inherently functions as a distributed algorithm, as it consists solely of marriage proposals and divorce declaration, both of which can be processed in parallel.

The distributed variant has been studied in various practical scenarios. For instance, the work of Maggs and Sitaraman [22] explores stable matching in content delivery networks, where stable matching is used for global load balancing by mapping client groups to server clusters. Moreover, stable matching has been leveraged in wireless networks: [3] employs this problem to pair primary and secondary users in a radio network, [8] relies on stable matching to pair users and uplink carriers when performing channel assignment, and numerous other studies have explored similar applications [2, 13, 26].

We note that, in such scenarios, it is important for the stable-matching-based mechanisms to be resilient to potential faults. Notably, the work of Maggs and Sitaraman [22] regarding stable matching in content delivery networks has pointed out the potential of (crash) failures. The mitigation strategy proposed by [22] relies on *leader election*: although crashes do not necessarily degrade the matching obtained, this is a point of failure if the leader misbehaves. To the best of our knowledge, prior works in distributed stable matching assume that parties follow the protocol, or that there is some central trusted unit which can gather all inputs, perform the stable matching algorithm and return the result. This motivates us to investigate scenarios where no such safety exists, and parties may not only crash, but also become byzantine and hence exhibit malicious behavior. Concretely, we ask the following question:

> *Can we achieve stable matching in a network*
> *even if some of the parties are byzantine?*

Our Contribution. We firstly define the *byzantine stable matching* problem bSM, taking into account that byzantine parties may choose not to participate in the protocol, and preventing honest parties from matching with the same byzantine party. We then investigate the necessary and sufficient conditions for achieving bSM under various synchronous network topologies, both with and without cryptographic assumptions (digital signatures), and we provide tight conditions. We denote the two sets by L and R, with $|L| = |R| = k$, and we assume that at most t_L parties in L and at most t_R parties in R may be byzantine. We consider fully-connected networks, *bipartite* networks (where the parties can only communicate to parties on the other side), and a topology in-between, which we call a *one-sided* networks: this maintains the communication channels of a bipartite network, but additionally provides the parties in side R with complete communication. We summarize our findings below:

- When no cryptographic setup is available, bSM can be solved if and only if:
  - $t_L < k/3$ or $t_R < k/3$ in a fully-connected network.
  - the following hold in a bipartite network: (i) $t_L, t_R < k/2$; (ii) $t_L < k/3$ or $t_R < k/3$.
  - the following hold in a one-sided network: (i) $t_R < k/2$; (ii) $t_L < k/3$ or $t_R < k/3$.
- Assuming digital signatures, bSM can always be solved if the network is fully connected. Otherwise, bSM can be solved if and only if:
  - any of the following holds in a bipartite network: (i) $t_L, t_R < k$; (ii) $t_L < k/3$ or $t_R < k/3$.
  - $t_R < k$ or $t_L < k/3$ in a one-sided network.

Our settings enable us to establish our conditions' sufficiency by reducing bSM to Byzantine Broadcast [20], with one notable exception: this approach falls short in bipartite networks with digital signatures, where one side may be completely byzantine, leaving the honest side disconnected. For this setting, we provide a protocol that simulates a *synchronous fully-connected network with omissions* for the disconnected side, allowing us to achieve bSM. We also add that our impossibility arguments prove, in fact, even stronger results: even a *simplified* version of bSM (where parties hold a single

favorite as input as opposed to a complete preference list) cannot be solved unless the stated conditions hold.

Related Work. While the stable matching problem has not been previously explored in the context of byzantine behaviour, malicious strategies such as *lying* were considered. Concretely, Roth [27] showed that stable matching is not *truthful*: there are scenarios where an individual can get a more favorable result by lying about their preferences. However, Gale and Shapley [11] proved that their algorithm is truthful for one side: an individual on the side doing the proposals can never gain by lying. The case where multiple individuals on the proposing side can collude in the Gale-Shapley algorithm [17] or where the preference lists can have some mistakes [23] have also been studied . We note that such adversarial models essentially consider manipulations of the *preferences lists*. In contrast, byzantine fault tolerance protocols are mainly concerned with providing reasonable guarantees even when byzantine parties attempt to prevent honest parties from obtaining *a solution*.

As the Gale-Shapley algorithm [11] naturally adapts to distributed settings, lower bounds regarding the number of queries between the two sides have been the topic of interest. Gonczarowski et al [12] provided a lower bound of $\Omega(n^2)$ boolean queries. However, this does not take into account communication between parties on the same side. In this case, there is still a gap between the best-known lower bound of $\Omega(n^2)$ and upper bound of $O(n^2 \log n)$. When preference lists are similar, Khanchandani and Wattenhofer [18] describe an algorithm with better complexity and provide a lower bound depending on the similarity of the lists. Approximation algorithms have also been a topic of interest as a strategy to circumvent the lower bound of [12]. Various definitions for an *approximation* of a stable matching have been analyzed, considering the number of blocking pairs [25], the number of matches which would have to be broken [12] or how blocking each pair is [19].

While some of our necessary conditions enable bSM to be reduced to well-established problems such as Byzantine Broadcast and Byzantine Agreement [20], we also encounter settings where bSM is strictly weaker than these fundamental problems, requiring novel insights. We also note that the term *matching* has occurred in previous works regarding byzantine faults or *self-stabilization* (starting from an arbitrary state, the system needs to reach a *legitimate configuration* eventually). Most of these works are concerned with finding a *maximal* matching [5, 16, 24], or a maximum matching [15] in a bipartite graph as opposed to a *stable* matching. A notable exception [21] focuses on self-stabilizing stable matching. Self-stabilizing protocols assume that all parties will stop being faulty at some point and that no decision is final: any party can decide to *unmatch* at any time. This contrasts with our work which is resilient to some parties being permanently faulty and guarantees that a final decision is reached within a bounded time.

2 Preliminaries

The *stable matching* problem can be informally described as the task of pairing two sets of parties in such a way that no two unmatched parties should prefer each other over their assigned partners. Stable matching problems have been typically framed in contexts such as matching men with women, students with universities, or producers with consumers.

Standard Stable Matching. We consider a set of $n = 2k$ parties $\mathcal{P}$, which are divided into two disjoint sets L and R with $|L| = |R| = k$: L can represent, for instance, the set of men/students/producers while R can represent the set of women/universities/consumers. In the *stable matching problem*, every party u in L (resp. R) has as input a *preference list* (a permutation) π_u over the parties in R (resp. L). We say that u *prefers* v over w if v appears before w in π_u. In addition, u prefers any party in its preference list π_u over being alone.

The objective of this problem is to determine a *stable matching*: a matching M between L and R such that there is no *blocking* pair. A (non-matched) pair of parties $(u, v) \in L \times R$ is *blocking* if u and v prefer each other compared to who they are currently matched to. As parties always prefer being matched to being alone, a pair of two unmatched parties on opposite sides is considered blocking. This implicitly requires the matching to be *maximal* (all parties are matched). It has been proven that a stable matching always exists, and it can be found using the Gale-Shapley algorithm $\mathcal{A}_{\text{G-S}}$ [11].

Theorem 1 ([11]). *There is a deterministic algorithm $\mathcal{A}_{G-S}$ that takes as input the preference lists π of all parties in L and R and returns a stable matching M.*

Stable Matching in a Network. We now define the stable matching problem in a distributed setting. Here, the parties in $L \cup R$ are processors running a protocol over a network, exchanging messages via *bidirectional authenticated communication channels*. We assume that the network is *synchronous*: the parties have synchronized clocks, all parties start at time 0, and every message is delivered within a publicly known amount of time Δ. This allows protocols to operate in rounds. As depicted in Fig. 1, we will explore different network topologies, described below.
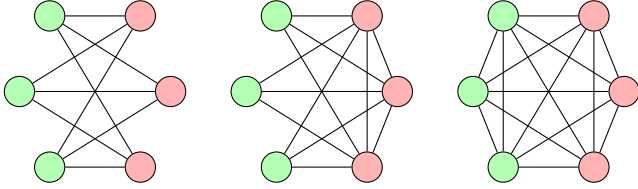
Figure 1: The different kinds of communication networks we consider. From left to right: bipartite, one-sided and fully-connected networks. Note that even when communication is possible within L or R, the matching is still between parties on opposite sides (not within L or R).

- **Fully-connected network:** Parties are pairwise connected. This model is relevant in scenarios such as forming partnerships within a close-knit social group.
- **One-sided network:** Parties are pairwise connected, except parties within L, which cannot communicate directly. This structure is applicable in contexts such as kidney donations, where privacy constraints prevent recipients from directly interacting with each other.
- **Bipartite network:** Only pairs of parties in $L \times R$ are connected. This setup is relevant in cases such as matching international job applicants, where communication is restricted solely to potential matches across the two sets.

We remark that each model is strictly stronger than the previous.

The parties will be then running a protocol Π where each party holds a preference list as input, and each party obtains as output its match (from the opposite side). In this setting, Π achieves *distributed stable matching* if the following properties hold:

- **Termination:** Each party outputs a party on the opposite side to match with;
- **Symmetry:** If party u decides to match party v, then v decides to match party u;
- **Stability:** There are no blocking pairs.

Faults. So far, we have defined the stable matching problem in a *fault-free* setting. From now on, we assume an adversary that may (permanently) corrupt up to t_L parties in L and up to t_R parties in R. The corrupted parties become *byzantine*: they may deviate arbitrarily (even maliciously) from the protocol. A party is *honest* if it never became byzantine. Our protocols will assume that the adversary is *adaptive*: it may choose to corrupt parties at any point of the protocol's execution. Our impossibility results, however, hold even against a *static* adversary, which needs to choose which parties to corrupt at the beginning of the protocol's execution.

Refining the Problem. Byzantine parties require us to refine the definition of the stable matching problem. First, we need to take into account that our properties should only be concerned with the outputs of *honest* parties. Second, the byzantine parties may choose not to participate in the protocol, preventing us from obtaining a maximal matching. Consequently, we adjust the previous properties as follows:

- **Termination:** Every *honest* party outputs: either a party on the opposite side *or nobody*.
- **Symmetry:** For two *honest* parties u and v, if u decides to match v, then v decides to match u.
- **Stability:** There are no blocking pairs made of *honest* parties.

Note that these properties are not strong enough to lead to a *relevant* matching: multiple honest parties may be matched to the same byzantine party (in the figure on the right, if the orange party is byzantine, the depicted matching satisfies symmetry and stability).

Therefore, we introduce an additional intuitive condition that prevents such scenarios:

- **Non-competition:** If two honest parties output $u, v \in L \cup R$, then $u \neq v$.

We may now present the formal definition of byzantine stable matching.

Definition 1 (Byzantine Stable Matching (bSM)). *Consider a protocol Π where every party in $L \cup R$ holds as input a preference list over the parties on the other side. Then, Π achieves byzantine stable matching (bSM) with respect to t_L and t_R if it satisfies the following even when up to t_L parties in L and t_R parties in R are byzantine: termination, symmetry, stability, non-competition.*
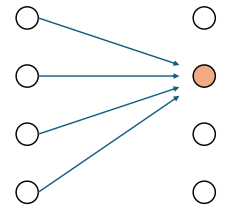
Cryptographic Assumptions. As we will see, the solvability of bSM in a given setting depends on whether we assume a trusted setup and cryptographic primitives. We will use the term *unauthenticated setting* to refer to settings where no cryptographic assumptions are made. In contrast, we use the term *authenticated setting* to refer to a setting where a public key infrastructure and a secure digital signature scheme are available. For simplicity of presentation, we assume that signatures are unforgeable. When replaced with real-world instantiations, our feasibility results in the authenticated setting still hold except for negligible probability (in the scheme's security parameter) against computationally-bounded adversaries.

Warm-up Solution. Synchronous networks come with communication primitives that often allow us to reduce bSM to an offline problem. One such primitive is *Byzantine Broadcast* (BB) [20].

Definition 2 (Byzantine Broadcast (BB)). *Let Π be a protocol where a designated party S (the sender) holds a value v_S. We say that Π achieves Byzantine Broadcast (BB) if the following hold even when up to some number t of the parties are corrupted (alternatively for our setting, up to t_L in L and up to t_R in R):*

- ***Termination****: All honest parties output;*
- ***Validity****: If S is honest, every honest party outputs v_S;*
- ***Consistency****: Honest parties output the same value.*

A BB protocol allows the sender to disseminate its preferences so that all parties obtain identical views of them. If each party runs an invocation of BB to distribute its preferences, then by the end, all parties will have identical views of everyone's preferences. This enables them to run $\mathcal{A}_{G\text{-}S}$ offline and obtain the same stable matching, thereby solving bSM. This provides us with the lemma below. The formal proof is available in the full version [6].

Lemma 1. *Whenever BB is available, bSM is solvable.*

3 Simplified Stable Matching

For most of our impossibility results, we do not require the inputs to be complete preference lists: we rely only on parties' favorites. Therefore, we introduce the *simplified stable matching* problem (sSM), which mostly follows the same rules as bSM. The main difference is that a party's input is a party on the other side, not a preference list. If party u has as input party v, we say that u's *favorite* is v. The stability property is then replaced by simplified stability:

- **Simplified stability**: If two honest parties are each other's favorites, they output each other.

Our impossibility proofs will describe settings where a protocol cannot simultaneously achieve termination, symmetry, non-competition, and this simplified property. In the full version of this paper [6], we show that sSM can be reduced to bSM, enabling us to state the following result:

Lemma 2. *Whenever sSM is not solvable, bSM is not solvable.*

We finish with a helpful technical lemma allowing our impossibility arguments to only focus on proving that sSM cannot be solved in settings with few parties. The lemma then generalizes our arguments to settings with more parties.

Lemma 3. *Let Π be a protocol solving sSM, supporting up t_L byzantine parties in L and t_R byzantine parties in R. Then, for any $0 < d \leq k = n/2$, there exists a protocol Π' solving sSM on $2d$ parties (d on each side) that supports up to $\lfloor \frac{t_L}{\lceil k/d \rceil} \rfloor$ byzantine parties on the left side and $\lfloor \frac{t_R}{\lceil k/d \rceil} \rfloor$ byzantine parties on the right side.*

Proof. We partition L into d disjoint sets $L_1, \ldots, L_d$ such that $1 \leq |L_1|, \ldots, |L_d| \leq \lceil |L|/d \rceil = \lceil k/d \rceil$. Similarly, we partition R into d disjoint sets $R_1, \ldots, R_d$ such that $1 \leq |R_1|, \ldots, |R_d| \leq \lceil |R|/d \rceil = \lceil k/d \rceil$. From each of these sets, we pick one representative: $l_1, \ldots, l_d, r_1, \ldots, r_d$. We build Π' solving sSM for $2d$ parties: $l'_1, \ldots, l'_d$ on the left side and $r'_1, \ldots, r'_d$ on the right side, as follows:

- Each party l'_i in Π' simulates all the parties in L_i running Π. Similarly, each party r'_j in Π' simulates all the parties in R_j running Π.
- Input: If the input (favorite) of l'_i is r'_j, then we assign r_j as the favorite of l_i. Similarly, if the input of r'_j is l'_i, then we assign l_i as the favorite of r_j. For parties that are not representatives of their group, we assign arbitrary favorites.
- Output: For a given l_i, if there is a r_j such that l_i matches r_j, then l'_i declares that it matches r'_j. Otherwise l'_i declares that it matches nobody.

We are essentially running the sSM algorithm on the whole graph, but only looking at the representative of each set and discarding anything unrelated to them. As a consequence, Π' achieves termination, symmetry, simplified stability, and non-competition since Π achieves termination, symmetry, simplified stability, and non-competition. As each party in Π' simulates up to $\lceil k/d \rceil$ parties from Π and Π supports up to t_L byzantine parties in L and t_R byzantine parties in R, the bound on the number of byzantine parties supported by Π' follows immediately. □

4 Solvability in Unauthenticated Settings

In this section, we describe tight conditions for solving bSM in unauthenticated settings (no cryptographic assumptions). In the following, we first present our findings in the fully-connected network case. Afterwards, we focus on the one-sided and bipartite network cases.

4.1 Fully-Connected Network

The conditions for the fully-connected network case are presented in Theorem 2, stated below.

Theorem 2. *bSM is solvable in a fully-connected unauthenticated network if and only if $t_L < k/3$ or $t_R < k/3$.*

We first focus on the feasibility part, for which we recall Lemma 1: if BB can be achieved in a setting, then bSM also can. As stated in the lemma below, BB is, in fact, solvable for our conditions. Hence, the two lemmas together enable us to conclude that bSM is solvable in a fully-connected unauthenticated network whenever $t_L < k/3$ or $t_R < k/3$, as desired. Lemma 4 is a corollary of [10, Theorem 2]. We note that [10] focuses on *general adversaries*. Roughly, this is an adversarial model where the adversary has to choose which parties to corrupt from a predefined subset-closed list of options.

Lemma 4. *BB is solvable in a fully-connected network if $t_L < k/3$ or $t_R < k/3$.*

As mentioned above, this is a corollary of [10, Theorem 2]. We again highlight that [10] assumes a *general adversary*, which we briefly introduce next. In this adversarial model, the corruption power of the adversary is specified by a (subset-closed) *adversarial structure* $\mathcal{Z} \subseteq 2^{\mathcal{P}}$, where $\mathcal{P}$ denotes the set of parties. In particular, the adversary may choose to corrupt any set of parties in $\mathcal{Z}$. For instance, if $\mathcal{P} := \{P_1, P_2, \ldots, P_5\}$, a potential adversarial structure $\mathcal{Z}$ is $\{\varnothing, \{P_1\}, \{P_2\}, \{P_1, P_2\}, \{P_4\}\}$, which means that the adversary may choose between corrupting no parties, corrupting parties P_1, P_2 (or only one of the two), or corrupting only party P_4. In contrast, one often considers a *threshold adversary*, which may corrupt up to t of the n parties (as is the case in most literature): this is a particular case of the general adversary model where $\mathcal{Z}$ is the set of all subsets of at most t parties. The adversary assumed in our work sits in-between these two: we assume that the adversary may corrupt up to t_L parties in L and up to t_R parties in R, hence our adversary structure is $\mathcal{Z}^{\star} := \{S_L \cup S_R \mid S_L \subseteq L, S_R \subseteq R, |S_L| \leq t_L, |S_R| \leq t_R\}$. This can be thought of as the product of two threshold adversary structures.

In the general adversaries model, [10, Theorem 2] states the following:

Theorem 3 ([10, Theorem 2]). *Assume a fully-connected unauthenticated network, and an adversary structure $\mathcal{Z}$ such that for any three sets $Z_1, Z_2, Z_3 \in \mathcal{Z}$ it holds that $Z_1 \cup Z_2 \cup Z_3 \neq \mathcal{P}$. Then, there is a protocol achieving BB in this setting.*

Then, to prove Lemma 4, we only need to show no three sets in our adversary structure $\mathcal{Z}^{\star}$ cover the set of n parties:

PROOF OF LEMMA 4. Consider our adversarial structure $\mathcal{Z}^{\star} := \{S_L \cup S_R \mid S_L \subseteq L, S_R \subseteq R, |S_L| \leq t_L, |S_R| \leq t_R\}$, and let $Z_1, Z_2, Z_3 \in \mathcal{Z}^{\star}$ be arbitrary. We show that $Z_1 \cup Z_2 \cup Z_3 \neq L \cup R$.

Without loss of generality, we may assume that the condition $t_L < k/3$ holds (the case where $t_R < k/3$ and $t_L \geq k/3$ is analogous). As every $Z \in Z^{\star}$ contains at most t_L parties in L, it follows that $Z_1 \cup Z_2 \cup Z_3$ contain at most $3 \cdot t_L < 3 \cdot k/3 = k$ parties in L. Hence, at least one element of L is uncovered by $Z_1 \cup Z_2 \cup Z_3$, from which $Z_1 \cup Z_2 \cup Z_3 \neq L \cup R$.

Then, we may apply Theorem 3 and conclude that there is a protocol achieving BB in our setting. □

We now show that at least one of the conditions $t_L < k/3$ and $t_R < k/3$ holding is necessary. To do so, we prove this property for the special case $n = 6$ for sSM. One of our proof's ingredients is the shifting scenarios proof technique from [9], i.e., defining a *larger* system to reach a contradiction. Then, Lemma 3 enables us to conclude that for arbitrary n, sSM is not solvable if $t_L \geq k/3$ and $t_R \geq k/3$. Finally, Lemma 2 lifts this impossibility result to bSM, completing the proof of Theorem 2.

Lemma 5. *Assume a fully-connected unauthenticated network and $n = 6$. Then, no protocol achieves sSM for $t_L = t_R = 1$.*

PROOF. Assume for a contradiction that Π achieves sSM in this setting. We start with a high-level outline of the proof. We will construct a larger system (i.e., for 12 parties) by 'duplicating' the communication graph and consider running Π in this new system, with

each party following the intended behavior of the corresponding party in the original system. By choosing specific pairs of byzantine parties and their strategies in the original system, the adversary will be able to simulate being in the new system towards the honest parties. Going even further, we will present three different setups of the original system where the adversary can achieve this simulation. Each setup will provide different insights into how the parties should behave in the larger system, leveraging that Π is correct for any setup of the original system. However, these findings will ultimately be contradictory, proving that Π cannot exist.

We denote the three parties in L by a, b, c, and the three parties in R by u, v, w. By duplicating each party, we obtain a graph with 12 nodes such that each node is connected to 3 parties on the opposite side, as depicted in Fig. 2 (i). Consider an execution of Π in the new system with the following inputs: c_1 and v_1 have each other as their favorite, a_2 and v_2 have each other as their favorite, and all other inputs are arbitrary. So far, because Π is running in a non-standard network, we cannot say anything about the output of the parties (or even whether they terminate or not). However, using indistinguishability arguments, we will prove that there exists a normal-operation scenario for Π where two honest parties match the same party, giving a contradiction:

- First, we consider the setting where a_2, b_2, u_2 and v_2 are honest while all the remaining nodes are being simulated (internally) by c and w, which are byzantine. This matches Fig. 2 (ii). This setting is valid for Π: there is at most one byzantine party in L and one byzantine party in R. As such, a_2, b_2, u_2 and v_2 must terminate. Moreover using the simplified stability property, because a_2 and v_2 prefer each other, we get that a_2 decides to match party v_2.

- Then, we consider the setting where b_1, c_1, v_1 and w_1 are honest, while the remaining nodes are being simulated by two byzantine parties a and u. This matches Fig. 2 (iii). Similarly to before, we get that c_1 must terminate and decide to match party v_2.

- The last setting is where c_1, a_2, u_2 and w_1 are honest, while the remaining nodes are being simulated by two byzantine parties b and v. This matches Fig. 2 (iv). We remark that a_2 (resp. c_1) cannot distinguish between this setting and Fig. 2 (ii) (resp. (iii)). As such, as we proved above, a_2 and c_1 will both decide to match v (previously we had written v_1 and v_2 to distinguish between the two copies, but here there is only one). Moreover, this setting is valid for Π, which means that its output should satisfy the properties of stable matching. However, a_2 and c_1 are both honest and decide to match the same party v, breaking the non-competition rule, hence we obtain a contradiction. □

4.2 Bipartite and One-Sided Networks

The theorems below give the necessary and sufficient conditions for the bipartite and one-sided communication cases. Note that, in contrast to the fully-connected case, each theorem requires one additional condition (i) to hold on top of the previous condition (ii) that was already required for the fully-connected case.
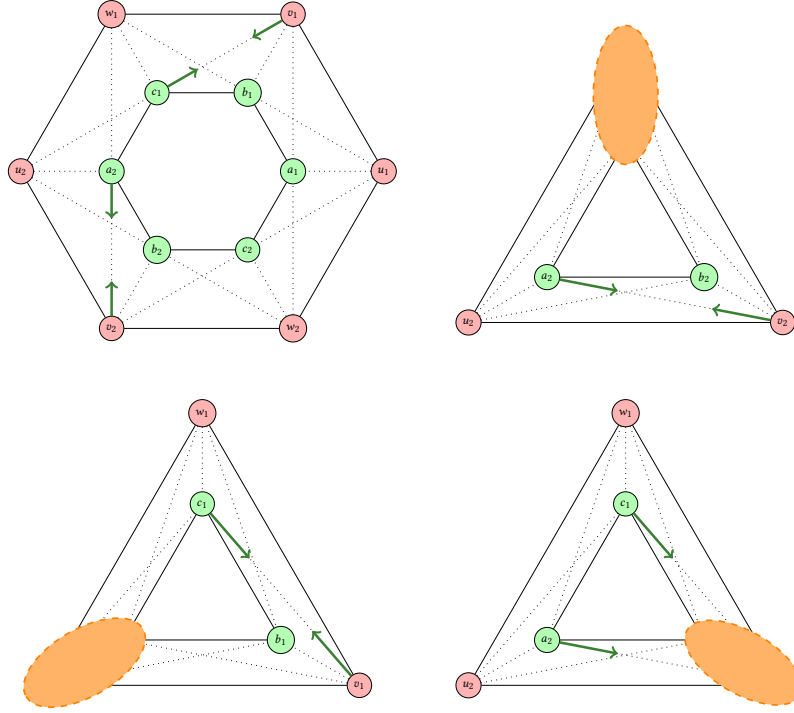
Figure 2: (i) Top left: system constructed in the proof of Lemma 5; (ii) Top right: indistinguishable execution where a_2, b_2, u_2 and v_2 are correct while the remaining nodes are simulated by the byzantine parties; (iii) Bottom left: indistinguishable execution where b_1, c_1, v_1 and w_1 are correct; (iv) Bottom right: indistinguishable execution where c_1, a_2, u_2 and w_1 are correct.

Theorem 4. *bSM is solvable in a bipartite unauthenticated network if and only if both of these conditions are satisfied: (i) $t_L, t_R < k/2$; (ii) $t_L < k/3$ or $t_R < k/3$.*

Theorem 5. *bSM is solvable in a one-sided unauthenticated network if and only if both of these conditions are satisfied: (i) $t_R < k/2$; (ii) $t_L < k/3$ or $t_R < k/3$.*

The main idea we will use to get the feasibility parts of the previous theorems is that for the stated conditions, we may actually return to assuming a fully-connected network. In particular, two parties u, v on the same side that do not share a communication channel may simulate such a channel between them by using the parties on the opposite side as a proxy: u sends the desired message to all parties on the other side, who in turn forward it to v, which takes a majority vote to decide on the sent message. This works as long as there is an honest majority on the other side, giving us the following lemma:

Lemma 6. *Let S and S' denote the two sides. If the parties in S are disconnected and $t_{S'} < k/2$, we may assume the parties in S are fully-connected.*

PROOF. Let u, v be parties in S. We want to simulate an authenticated channel between u and v, i.e the receiver knows who the sender is.

If u wants to send a message M to v, it sends the message $(u \rightarrow v, M)$ to every party in S'. Then if a party in S' receives a message $(u \rightarrow v, M)$ from u, it forwards it to v. Finally, if v receives the same

message $(u \rightarrow v, M)$ from a majority (i.e strictly more than $k/2$) of S', it considers it received message M from u.

Using this strategy, we can see that sending a message takes a bounded amount of time (at most 2Δ). Moreover, if u is honest and sends a message M, at least $k - t_{S'} > k/2$ parties from S' will forward it to v which will therefore accept it. If v accepts a message M from u, this means strictly more than $k/2$ parties from S' forwarded it. Because $t_{S'} < k/2$, at least one honest party forwarded it, meaning u intended to send this message (being honest or not). □

This provides us with the corollaries below, enabling us to prove the stated conditions to be sufficient when combined with Theorem 2, which has assumed a fully-connected network.

Corollary 1. *In a one-sided network, we may assume a fully-connected network if $t_L < k/2$.*

Corollary 2. *In a bipartite network, we may assume a fully-connected network if $t_L, t_R < k/2$.*

To prove that the conditions in Theorems 4 and 5 are also necessary, we make use of the lemma below. Similarly to the proof of Lemma 5, our argument includes a technique from [9] that enables us to reach a contradiction by working with a non-standard system.

Lemma 7. *Assume a one-sided unauthenticated network and $n = 4$. Then, no protocol achieves sSM for $t_L = 0$ and $t_R = 1$.*
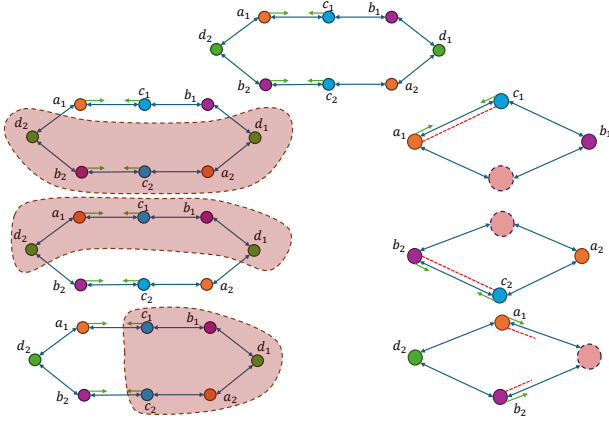
Figure 3: The eight parties in a cycle. In the first case, a_1 and c_1 are both honest and each other's favorites. Therefore, they must match each other. In the second case, we similarly get that b_2 and c_2 must match. Finally, in the last case, we get that both a_1 and b_2 try to match party c which is byzantine, but the non-competition property does not allow this.

PROOF. Write a, b for the nodes in L and c, d for the nodes in R. All nodes are connected in the communication network except a and b. We will prove that no such protocol exists even in the no-harder setting where *exactly* one party in R is corrupted, which we henceforth assume.

It will suffice to prove the impossibility for the bipartite network case, i.e., without the edge c-d. In particular, we claim that messages sent across this edge cannot be helpful. To see this intuitively, recall our assumption that *exactly* one party in R is byzantine, say d. Party c knows that d is byzantine, meaning any messages received from d could be completely arbitrary. Therefore, c may as well simulate receiving them by replacing them with a default value. Henceforth, we assume that the communication network is bipartite.

Assume for a contradiction that Π is a protocol achieving sSM for $n = 4$ parties $L = \{a, b\}$ and $R = \{c, d\}$ in a bipartite unauthenticated network given that no party in L is corrupted and *exactly* one party in R is corrupted.

The key insight in our proof is that the bipartite communication network actually forms the undirected cycle a-c-b-d-a. We construct a larger system by duplicating each party and linking them into a cycle twice as long: a_1-c_1-b_1-d_1-a_2-c_2-b_2-d_2-a_1. This is depicted in the first row of Fig. 3. We consider running Π in this setting by running the protocol used for a on a_1 and a_2, the protocol used for b on b_1 and b_2, and so on. We will now assign favorites (inputs) to the vertices: we make a_1 and c_1 each other's favorites and b_2 and c_2 each other's favorites. Other vertices are assigned favorites arbitrarily. We will show that by running protocol Π in this setting, we get a contradiction.

First (second row in Fig. 3), we consider the case where a_1, c_1 and b_1 are honest while d is byzantine and simulating d_1-a_2-...-d_2. In this case, because a_1 and c_1 are honest and both each other's favorites, by simplified stability, they must match each other.

By symmetry (third row in Fig. 3), b_2 and c_2 must match each other.

Last (forth row in Fig. 3), we consider the case where b_2, d_2 and a_1 are honest while c is byzantine and simulating c_1-b_1-...-c_2. In this case, a_1 and b_2 are both honest and both decide to match the same party c (by the previous two cases), which is prohibited by non-competition, giving us a contradiction. □

We conclude the section by presenting the proofs of Theorem 4, giving the conditions for bipartite networks, and Theorem 5, providing the conditions for one-sided networks.

PROOF OF THEOREM 4. Lemma 6 implies that the bipartite communication model is weaker than the one-sided communication model, and therefore Theorem 5 enables us to conclude that the conditions in our Theorem's statement are necessary. Note that, for the condition $t_L < k/2$ we need to apply Lemma 6 for $S = L$ and $S' = R$, while for $t_R < k/3$ we need $S = R$ and $S' = L$. For sufficiency, the condition $t_L, t_R < k/2$ enables us to apply Corollary 2 and assume a fully-connected network. Afterwards, Theorem 2 ensures that bSM is solvable. □

PROOF OF THEOREM 5. Lemma 7 proves that for $n = 4$ parties, sSM cannot be achieved in this setting if $t_R \geq 1$. Then, using Lemma 3, this proves that for an arbitrary n, sSM cannot be achieved if $t_R \geq k/2$ in a one-sided network. Therefore, using Lemma 2, this proves that $t_R < k/2$ is necessary for bSM as well. When $t_R < k/2$ holds, Corollary 2 enables us to assume a fully-connected network. Then, we may conclude using Theorem 2 that it is both necessary and sufficient that one of the conditions $t_L < k/3$, $t_R < k/2$ holds. □

5 Solvability in Authenticated Settings

We now consider authenticated settings. Assuming digital signatures will enable us to solve bSM up to much higher corruption thresholds in contrast to the unauthenticated case. This section is organized similarly to Section 4: we first analyze the fully-connected network case, and afterwards focus on one-sided and bipartite networks.

5.1 Fully-Connected Network

In the fully-connected network case, bSM is always solvable: we utilize the Dolev-Strong protocol [7], which achieves BB resilient against $t < n$ corruptions assuming PKI. Then, Lemma 1 directly implies the theorem below.

Theorem 6. *bSM is solvable in a fully-connected authenticated network.*

5.2 Bipartite and One-Sided Networks

The one-side and bipartite communication models offer more interesting restrictions, as described by the theorems below. Note that these results imply that, given PKI, bSM is solvable even when one side is fully byzantine, i.e. $t_R = k$. This may seem counter-intuitive, as the honest parties' communication graph may be completely disconnected. However, we need to highlight that, if one side is completely byzantine, the bSM definition allows the honest parties to simply match with nobody: any partial matching that satisfies non-competition suffices.

Theorem 7. *bSM is solvable in a bipartite authenticated network if at least one of these conditions holds: (i) $t_L, t_R < k$; (ii) $t_L < k/3$ or $t_R < k/3$.*

Theorem 8. *bSM is solvable in a one-sided authenticated network if $t_R < k$ or $t_L < k/3$.*

Sufficient Conditions. In the following, we first focus on showing that the conditions described by the theorems above are sufficient. In the unauthenticated setting, our conditions enabled us to provide the parties in the disconnected side with full communication. In turn, this allowed us to prove sufficiency by reducing the one-sided/bipartite network cases to the fully-connected network case. Assuming signatures enables us to proceed similarly: we may provide parties in side S with complete communication *whenever side S' (which may or may not be connected) contains at least one honest party.* Roughly, the parties in S will send signed messages to parties in S', and the (honest) parties in S' will forward these messages. Parties in S will then accept messages signed correctly. We present the formal proof in the full version [6].

Lemma 8. *Denote the two sides L and R by S and S'. If the parties in side S are disconnected, $t_{S'} < k$, and the network is authenticated, we may assume the parties in S are fully connected.*

The next corollaries follow from Lemma 8. Note that Corollary 3 enables us to conclude that $t_R < k$ is a sufficient condition for bSM in a one-sided network with PKI. Similarly, Corollary 4 implies that $t_L, t_R < k$ is a sufficient condition for bSM in a bipartite authenticated network.

Corollary 3. *In a one-sided authenticated network, we may assume a fully-connected authenticated network if $t_R < k$.*

Corollary 4. *In a bipartite authenticated network, we may assume a fully-connected authenticated network if $t_L, t_R < k$.*

Note that, for the one-sided network case we may already conclude that the conditions presented in Theorem 8 are sufficient. The condition $t_R < k$ follows from Corollary 3, which enables us to reduce this case to the fully-connected authenticated setting described by Theorem 6. Moreover, the condition $t_L < k/3$ is sufficient due to Theorem 5, which states that bSM can be solved even in an unauthenticated one-sided network when $t_L < k/3$.

For the bipartite network case, the condition $t_L < k$ and $t_R < k$ being sufficient follows directly from Corollary 4. Showing that having $t_L < k/3$ or $t_R < k/3$ is also sufficient introduces, however, different challenges. From this point on, we may assume without loss of generality that $t_L < k/3$, which implies that the side R may be fully byzantine. Note that this may cause the honest parties in L to be completely disconnected.

While we cannot assume that the parties in L are in a fully-connected network, we will be able to assume that they are *in a fully-connected network with omissions*: a message may either be received within $2 \cdot \Delta$ units of time, or it is never delivered. Moreover, omissions occur *only* if all parties in side R are byzantine. We achieve this using the following strategy: whenever a party in $P \in L$ needs to send a message m to a party in $P' \in R$, it sends the *signed* message $\text{m}' := (P \to P', \tau, \text{id}, \text{m})$ to all parties in R, where id is a message identifier and τ is the time when m' is sent. The (honest) parties in R forward the signed message m' to P'. P'

accepts this message only if the signature is valid and at most $2 \cdot \Delta$ time has passed since time τ. We add that, as byzantine parties cannot forge signatures on the honest parties' behalf, this ensures reliable communication (up to omissions).

We may then design a protocol in the bipartite network case as follows: we (attempt) to provide the parties in L with all parties' preferences lists. The parties in L will run $\mathcal{A}_{\text{G-S}}$ locally, which enables them to obtain their own matches and inform the parties in R about their matches. Due to the forwarding mechanism, we may assume that the parties in L are in a fully-connected network where omissions only occur if all parties in R are byzantine.

To provide the honest parties in L with identical views over the preferences' list, we rely on two building blocks: a synchronous BB protocol Π_{BB}, and a synchronous *Byzantine Agreement* (BA) protocol Π_{BA}. We recall the definition of BA below.

DEFINITION 3 (BYZANTINE AGREEMENT). *Let Π be a protocol where every party holds a value as input. We say that Π achieves BA if the following hold even when up to t parties are corrupted:*

- ***Termination***: *All honest parties output;*
- ***Validity***: *If all honest parties hold the same input value v, they output v;*
- ***Agreement***: *All honest parties output the same value.*

The potential for omissions will require us to enhance these protocols by adding a few properties when omissions occur: termination, and *weak agreement*, described below. Note that we do not require any validity condition.

- ***Weak agreement***: *If P and P' are honest and output $v \neq \bot$ and $v' \neq \bot$ respectively, $v = v'$.*

The theorems below describe our building blocks. Π_{BA} is obtained by making adjustments to the protocol of [4], and Π_{BB} is a simple reduction to Π_{BA}. For constructions, see the full version [6].

Theorem 9. *Assume the k parties in L are in a fully-connected synchronous network with delay Δ. If $t_L < k/3$, there is a k-party protocol Π_{BA} achieving BA within $\Delta_{\text{BA}}(\Delta)$ time. Moreover, if omissions occur, Π_{BA} still achieves weak agreement and termination within $\Delta_{\text{BA}}(\Delta)$ time.*

Theorem 10. *Assume the k parties in L are in a fully-connected synchronous network with delay Δ. If $t_L < k/3$, there is a k-party protocol Π_{BB} achieving BB within $\Delta_{\text{BB}}(\Delta)$ time. Moreover, if omissions occur, Π_{BB} still achieves weak agreement and termination within $\Delta_{\text{BB}}(\Delta)$ time.*

We present the code of our protocol below.

Protocol Π_{bSM}

Code for party $P \in R$ with input π

1: Whenever you receive a properly signed message $\text{m}' = (P'' \to P', \tau, \text{id}, \text{m})$ from $P'' \in L$, forward the signed message to $P' \in L$.
2: Send your preference list σ to every party in L.
3: At time $\max(\Delta_{\text{BA}}(2\Delta) + \Delta, \Delta_{\text{BB}}(2\Delta)) + \Delta$:
4:     $M_p :=$ matching suggestions received from parties in L.
5:     Decide to match according to the most common suggestion in M_p (breaking ties arbitrarily).

Code for party $P \in L$ with input π

1: id := 0. Whenever you need to send a message m to $P' \in L$, let
$\tau :=$ the current time. Send the signed message $(P \to P', \tau, \mathsf{id}, \mathsf{m})$
to all parties in R and increment id.
2: In parallel:
3:      Send π to all parties via Π_{BB}. Let σ_ℓ denote the list received
     via Π_{BB} from party $P_\ell \in L$.
4:      Wait Δ time to receive preference lists from parties in R.
     Join an invocation of Π_{BA} for every party P_r in R: with
     input π_r if you have received π_r from P_r, and with a default
     preference list otherwise. Obtain outputs σ_r.
5: At time $\max(\Delta_{\mathsf{BA}}(2\Delta) + \Delta, \Delta_{\mathsf{BB}}(2\Delta))$:
6:      If any value in $(\sigma_v)_{v \in L \cup R}$ is $\bot$:
7:          Decide to match with nobody and terminate.
8:      Run $\mathcal{A}_{\mathsf{G\text{-}S}}$ locally with input $((\sigma_l)_{l \in L}, (\sigma_r)_{r \in R})$, and
     obtain output M.
9:      Send to each party $P_r \in R$ whom they should match to
     according to M.
10:      Decide who to match to according to M.

The next lemma states the guarantees of Π_{bSM}.

Lemma 9. Π_{bSM} *achieves bSM in a bipartite authenticated network if $t_L < k/3$.*

We split the proof of Lemma 9 into three lemmas. First, Lemma 10 describes the communication among the parties in L in Π_{bSM}, allowing us to assume that the parties in L are in a fully-connected network where omissions may only occur if all parties in R are byzantine. Under this assumption, Lemma 11 shows that bSM is achieved when no party in R is honest, and Lemma 12 shows that bSM is achieved when R contains at least one honest party.

Lemma 10. *We may assume the parties in L are in a fully-connected network with maximum delay $2 \cdot \Delta$ where omissions occur only if all parties in R are byzantine.*

PROOF. Let (u, v) be two parties in L and assume that u wants to send a message m to v. u sends a signed message $(u \to v, \tau, \mathsf{id}, \mathsf{m})$ to all parties in R, τ being the current timestamp and id being a message identifier. Parties in R then forward this signed message to v. If v receives a message $(u, \to, v, \tau, \mathsf{id}, \mathsf{m})$ properly signed by u such that τ is at most 2Δ units of time in the past and u has not seen id has not been seen before, it accepts message id from u.

With this approach, if at least one party in R is honest, messages always get forwarded and received within 2Δ units of time. Otherwise, note that byzantine parties cannot forge signatures on the honest parties' behalf: the byzantine parties may choose whether to forward the message or not, then causing an omission. □

Lemma 11. *If every party in R is byzantine and $t_L < k/3$, Π_{bSM} achieves bSM.*

PROOF. According to Lemma 10, the parties in L run Π_{BA} and Π_{BB} run in a fully-connected network with omissions. As a consequence, the weak agreement and termination properties hold according to Theorem 9 and Theorem 10: if the parties receive non-$\bot$ outputs, then these outputs are consistent.

Since Π_{BA} and Π_{BB} achieve termination, Π_{bSM} achieves termination as well.

Because all parties in R are byzantine, symmetry and stability are immediate: these properties concern two honest parties on opposite sides, which never happens here because one side is fully byzantine. We note that some honest parties in L may have received $\bot$ and decided to match with nobody but this still results in a stable matching for this specific setting.

As for non-competition, weak agreement guarantees that the honest parties who have obtained preference lists in each of the Π_{BA} and Π_{BB} invocations run $\mathcal{A}_{\mathsf{G\text{-}S}}$ with the same input. Theorem 1 ensures that these parties obtain the same matching M. Therefore, we conclude that Π_{bSM} achieves bSM whenever all parties in R are byzantine.

□

Lemma 12. *If R contains an honest party and $t_L < k/3$, Π_{bSM} achieves bSM.*

PROOF. Parties in L are in a fully-connected network with no omissions according to Lemma 10, hence Π_{BA} and Π_{BB} achieve respectively BA according to Theorem 9 and BB according to Theorem 10. Consequently, all honest parties run $\mathcal{A}_{\mathsf{G\text{-}S}}$ locally on the same input due to agreement and termination. Moreover, the validity properties of Π_{BA} and Π_{BB} ensure that honest parties' preference lists are received correctly and used as input in the local run of $\mathcal{A}_{\mathsf{G\text{-}S}}$. We therefore obtain that the honest parties in L run the same instance of $\mathcal{A}_{\mathsf{G\text{-}S}}$ locally, and every honest party's preference list in $\mathcal{A}_{\mathsf{G\text{-}S}}$ is the same as its original input. Therefore, the stable matching M computed by $\mathcal{A}_{\mathsf{G\text{-}S}}$ also satisfies our bSM definition.

The last step is to prove that every honest party decides according to M. This is immediate for honest parties in L. As for honest parties in R, they decide according to the most common option sent by parties in L. Since $k - t_L > t_L$ of the parties in L are honest, each party in R receives its match in M as the majority option, and decides on this match. Therefore, all honest parties decide according to M. Consequently Π_{bSM} achieves bSM whenever R contains at least one honest party. □

Necessary Conditions. We still need to show that the conditions presented in Theorem 7 and Theorem 8 are necessary. We write our proof for one-sided communication, and the bipartite network case will be a corollary.

Lemma 13. *If $t_R = k$ and $t_L \geq k/3$, then achieving bSM is a one-sided network is impossible.*

PROOF. We assume by contradiction that there is a protocol achieving bSM in this setting. Using Corollary 2 and Lemma 3, this means that there exists a protocol Π which solves sSM on $n := 6$ nodes with $t_R = 3$ and $t_L = 1$. We denote the six parties by $L = \{a, b, c\}$ and $R = \{u, v, w\}$, and assume that b and all parties in R are byzantine. In the following, we fix an input configuration, and we describe an adversarial strategy that breaks the guarantees of Π in this setting: honest parties a and c will match with the same byzantine party v in R, hence breaking non-competition.

To do so, we define a and c's favorite as v. Moreover, each byzantine party will internally simulate two instances of themselves running protocol Π. For example, v will internally simulate two instances of itself v_1 and v_2 such that v_1's favorite is a and v_2's favorite is b. Each of the remaining byzantine parties $x \in \{b, u, w\}$ simulates two instances of itself x_1 and x_2 with any arbitrary inputs.
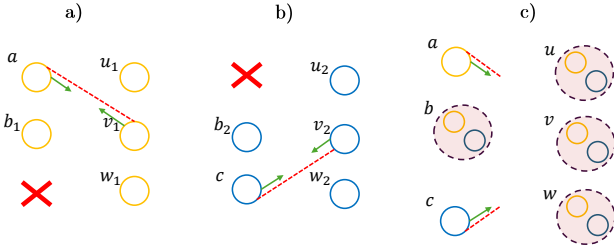
Figure 4: a) From the point of view of a, all parties are honest except c (which crashed), simplified stability guarantees that a matches with v. b) From the point of view of c, all parties are honest except a (which crashed), simplified stability guarantees that c matches with v. c) What is actually happening is that all byzantine parties are simulating two versions of themselves except a and c, but both a and c are honest and try to match v, which is not allowed by the non-competition property.

Each communication edge has at least one of its two endpoints being a byzantine party in our setting: since the parties in L are only connected to parties in R in a one-sided network, byzantine parties have full control over the communication network. They may therefore divide the communication network in two groups: $\{a, b_1, u_1, v_1, w_1\}$ and $\{b_2, c, u_2, v_2, w_2\}$. Messages only get sent and received within a group, meaning that if party a running Π wants to send a message to w, w_1 will receive it. Any message sent from u_1, v_1 and w_1 to c is never received.

We consider the output of a and c after running protocol Π in this setting. We have $t_L = 1$ and $t_R = 3$, which satisfies Π's requirements. Consequently, termination holds: a and c must decide to either match with some party or no one.

We then consider a new scenario for party a: parties a, b, u, v, w are all honest with the same favorites as the previous scenario's first group. Party c is byzantine and crashes at the beginning, i.e., it does not send any message. In this scenario, $t_L = 1$ and $t_R = 0$, which satisfies the requirements of Π. Therefore, termination holds and a obtains an output. Since both a and v are honest and each other's favorite, they must match with each other according to simplified stability. However, we remark that a cannot distinguish between this scenario and the previous one: it receives the exact same messages in both cases. Therefore, in the first scenario, a also decides to match with v.

We may construct a symmetric scenario for party c: this time, parties b, c, u, v, w are honest with the same inputs as in the first scenario's second group. Party a is byzantine and crashes at the beginning of the protocol's execution. As $t_L = 1$ and $t_R = 0$, the requirements of Π are satisfied: termination and simplified stability hold. Therefore, c outputs v. Moreover, this scenario is indistinguishable to c from the first scenario, hence c matches with v in the first scenario as well.

We consequently obtain a contradiction: in the first scenario, both a and c are honest and match with the same party, which breaks non-competition.   □

As the bipartite communication model is weaker than the one-sided model, Lemma 13 provides the following corollary.

Corollary 5. *If $t_R = k$ (resp. $t_L = k$) and $t_L \geq k/3$ (resp. $t_R \geq k/3$), then achieving bSM in a bipartite network is impossible.*

Putting It All Together. We conclude the section by providing the formal proofs of Theorem 7 and Theorem 8. We first present the proof of Theorem 7, focusing on a bipartite network.

PROOF OF THEOREM 7. We first discuss sufficiency. If $t_L < k$ and $t_R < k$, Corollary 4 enables us to assume a fully-connected network. Therefore, using Theorem 6, we obtain that bSM is solvable. If $t_L < k/3$ and $t_R \leq k$, Lemma 9 describes a protocol achieving bSM. The case $t_R < k/3$ and $t_L \leq k$ is symmetrical.

Otherwise, if $t_L \geq k/3$ and $t_R = k$ or the opposite, we may apply Corollary 5 and conclude that bSM is impossible.   □

We now prove Theorem 8, discussing the one-sided network case.

PROOF OF THEOREM 8. For sufficiency, when $t_R < k$, we may apply Lemma 8 and hence assume a fully-connected network. Then, Theorem 6 enables us to conclude that bSM is solvable. If $t_R = k$ and $t_L < k/3$, Theorem 5 guarantees that bSM is solvable.

When none of these conditions holds, i.e., if $t_R = k$ and if $t_L \geq k/3$, Lemma 13 enables us to conclude that bSM is impossible.   □

6 Conclusion

We investigated whether stable matching can be achieved in a synchronous network where some of the parties involved may be byzantine. We analyzed this problem under various network topologies, both with and without cryptographic assumptions. For each setting, we gave necessary and sufficient conditions, assuming that each party holds as input a complete ranking of the parties on the other side.

Our work highlights multiple promising directions for further research. A first direction could be generalizing our results to the *stable roommate* problem. Instead of assuming that the parties to be matched are in two disjoint sets, the stable roommate problem seeks a stable matching within the same set. Note that our necessary conditions also apply to a byzantine variant of the stable roommate problem, even though there is no longer a distinction between byzantine parties on the two sides. However, the stable matching problem comes with the guarantee that a stable matching always exists, while the stable roommate problem does not. Hence, definitions and properties need to be refined to account for this.

Another interesting direction would be to extend our question to the asynchronous model. Using our current definitions, one can prove that even if only one party known in advance can be byzantine, stable matching is not solvable. Therefore, the properties required for the stable matching would have to be relaxed for this problem to be of interest.

Finally, while our work has provided a complete characterization in terms of solvability, there are multiple aspects in which our feasibility results could be improved. This includes improvements in terms of efficiency (i.e., communication complexity), but also improvements in terms of guarantees, such as providing some degree of privacy.

References

[1] 2013. Stable Allocations and the Practice of Market Design: The Royal Swedish Academy of Sciences. *The Indian Economic Journal* 60, 4 (2013), 3–34. https://doi.org/10.1177/0019466220130402

[2] Siavash Bayat, Raymond H. Y. Louie, Zhu Han, Yonghui Li, and Branka Vucetic. 2012. Distributed stable matching algorithm for physical layer security with multiple source-destination pairs and jammer nodes. 2688–2693. https://doi.org/10.1109/WCNC.2012.6214256

[3] Siavash Bayat, Raymond H. Y. Louie, Yonghui Li, and Branka Vucetic. 2011. Cognitive Radio Relay Networks with Multiple Primary and Secondary Users: Distributed Stable Matching Algorithms for Spectrum Access. In *2011 IEEE International Conference on Communications (ICC)*. 1–6. https://doi.org/10.1109/icc.2011.5962935

[4] Piotr Berman, Juan A. Garay, and Kenneth J. Perry. 1989. Towards Optimal Distributed Consensus. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science (SFCS '89)*. IEEE Computer Society, USA, 410–415. https://doi.org/10.1109/SFCS.1989.63511

[5] Subhendu Chattopadhyay, Lisa Higham, and Karen Seyffarth. 2002. Dynamic and self-stabilizing distributed matching. In *Proceedings of the Twenty-First Annual Symposium on Principles of Distributed Computing* (Monterey, California) *(PODC '02)*. Association for Computing Machinery, New York, NY, USA, 290–297. https://doi.org/10.1145/571825.571877

[6] Andrei Constantinescu, Marc Dufay, Diana Ghinea, and Roger Wattenhofer. 2025. Byzantine Stable Matching. https://doi.org/10.48550/arXiv.2502.05889 Full version of this paper.

[7] Danny Dolev and H. Raymond Strong. 1983. Authenticated algorithms for Byzantine agreement. *SIAM J. Comput.* 12, 4 (1983), 656–666. https://doi.org/10.1137/0212045

[8] Ahmad M. El-Hajj, Zaher Dawy, and Walid Saad. 2012. A stable matching game for joint uplink/downlink resource allocation in OFDMA wireless networks. In *2012 IEEE International Conference on Communications (ICC)*. 5354–5359. https://doi.org/10.1109/ICC.2012.6364329

[9] Michael J. Fischer, Nancy A. Lynch, and Michael Merritt. 1985. Easy Impossibility Proofs for Distributed Consensus Problems. In *4th ACM PODC*, Michael A. Malcolm and H. Raymond Strong (Eds.). ACM, New York, NY, USA, 59–70. https://doi.org/10.1145/323596.323602

[10] Matthias Fitzi and Ueli M. Maurer. 1998. Efficient Byzantine Agreement Secure Against General Adversaries. In *Proceedings of the 12th International Symposium on Distributed Computing (DISC '98)*. Springer-Verlag, Berlin, Heidelberg, 134–148.

[11] D. Gale and L. S. Shapley. 1962. College Admissions and the Stability of Marriage. *The American Mathematical Monthly* 69, 1 (1962), 9–15. http://www.jstor.org/stable/2312726

[12] Yannai A. Gonczarowski, Noam Nisan, Rafail Ostrovsky, and Will Rosenbaum. 2019. A stable marriage requires communication. *Games and Economic Behavior* 118 (2019), 626–647. https://doi.org/10.1016/j.geb.2018.10.013

[13] Yunan Gu, Yanru Zhang, Miao Pan, and Zhu Han. 2015. Matching and Cheating in Device to Device Communications Underlying Cellular Networks. *IEEE Journal on Selected Areas in Communications* 33, 10 (2015), 2156–2166. https://doi.org/10.1109/JSAC.2015.2435361

[14] Dan Gusfield and Robert W. Irving. 1989. *The stable marriage problem: structure and algorithms.* MIT Press, Cambridge, MA, USA.

[15] Rachid Hadid and Mehmet Karaata. 2009. Stabilizing maximum matching in bipartite networks. *Computing* 84 (04 2009), 121–138. https://doi.org/10.1007/s00607-009-0025-z

[16] Su-Chu Hsu and Shing-Tsaan Huang. 1992. A self-stabilizing algorithm for maximal matching. *Inform. Process. Lett.* 43, 2 (1992), 77–81. https://doi.org/10.1016/0020-0190(92)90015-N

[17] Chien-Chung Huang. 2006. Cheating by men in the gale-shapley stable matching algorithm. In *Proceedings of the 14th Conference on Annual European Symposium - Volume 14* (Zurich, Switzerland) *(ESA'06)*. Springer-Verlag, Berlin, Heidelberg, 418–431. https://doi.org/10.1007/11841036_39

[18] Pankaj Khanchandani and Roger Wattenhofer. 2017. Distributed Stable Matching with Similar Preference Lists. In *20th International Conference on Principles of Distributed Systems (OPODIS 2016) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 70)*, Panagiota Fatourou, Ernesto Jiménez, and Fernando Pedone (Eds.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 12:1–12:16. https://doi.org/10.4230/LIPIcs.OPODIS.2016.12

[19] Alex Kipnis and Boaz Patt-Shamir. 2009. Brief announcement: a note on distributed stable matching. In *Proceedings of the 28th ACM Symposium on Principles of Distributed Computing* (Calgary, AB, Canada) *(PODC '09)*. Association for Computing Machinery, New York, NY, USA, 282–283. https://doi.org/10.1145/1582716.1582766

[20] Leslie Lamport, Robert Shostak, and Marshall Pease. 1982. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems* 4, 3 (1982), 382–401. https://doi.org/10.1145/357172.357176

[21] Marie Laveau, George Manoussakis, Joffroy Beauquier, Thibault Bernard, Janna Burman, Johanne Cohen, and Laurence Pilard. 2017. Self-stabilizing Distributed Stable Marriage. In *Stabilization, Safety, and Security of Distributed Systems*, Paul Spirakis and Philippas Tsigas (Eds.). Springer International Publishing, Cham, 46–61.

[22] Bruce M. Maggs and Ramesh K. Sitaraman. 2015. Algorithmic Nuggets in Content Delivery. *SIGCOMM Comput. Commun. Rev.* 45, 3 (July 2015), 52–66. https://doi.org/10.1145/2805789.2805800

[23] Tung Mai and Vijay V. Vazirani. 2018. Finding Stable Matchings That Are Robust to Errors in the Input. In *26th Annual European Symposium on Algorithms (ESA 2018) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 112)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 60:1–60:11. https://doi.org/10.4230/LIPIcs.ESA.2018.60

[24] Fredrik Manne, Morten Mjelde, Laurence Pilard, and Sébastien Tixeuil. 2007. A New Self-stabilizing Maximal Matching Algorithm. In *Structural Information and Communication Complexity*. Springer Berlin Heidelberg, Berlin, Heidelberg, 96–108. https://doi.org/10.1016/j.tcs.2008.12.022

[25] Rafail Ostrovsky and Will Rosenbaum. 2015. Fast Distributed Almost Stable Matchings. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing* (Donostia-San Sebastián, Spain) *(PODC '15)*. Association for Computing Machinery, New York, NY, USA, 101–108. https://doi.org/10.1145/2767386.2767424

[26] Francesco Pantisano, Mehdi Bennis, Walid Saad, Stefan Valentin, and Mérouane Debbah. 2013. Matching with externalities for context-aware user-cell association in small cell networks. In *2013 IEEE Global Communications Conference (GLOBECOM)*. 4483–4488. https://doi.org/10.1109/GLOCOMW.2013.6855657

[27] Alvin E. Roth. 1982. The Economics of Matching: Stability and Incentives. *Math. Oper. Res.* 7 (1982), 617–628. https://doi.org/10.1287/moor.7.4.617