Rendezvous Under Smart Jamming

Mohammad J. Abdel-Rahman and Marwan Krunz

Sponsored by: Raytheon

Background

Quorum System: A collection of nonempty sets (called quorums) that pairwise overlap by one or more elements.
Example: Q = { {3, 4}, {2, 3}, {2, 4} } is a quorum system on {2, 3, 4}.

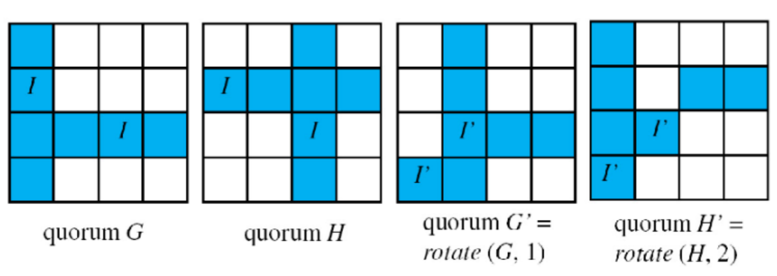
Grid Quorum System (GQS): The elements of the set are arranged into a square array. Each quorum consists of one column and one row.

1	2	3
4	5	6
7	8	9

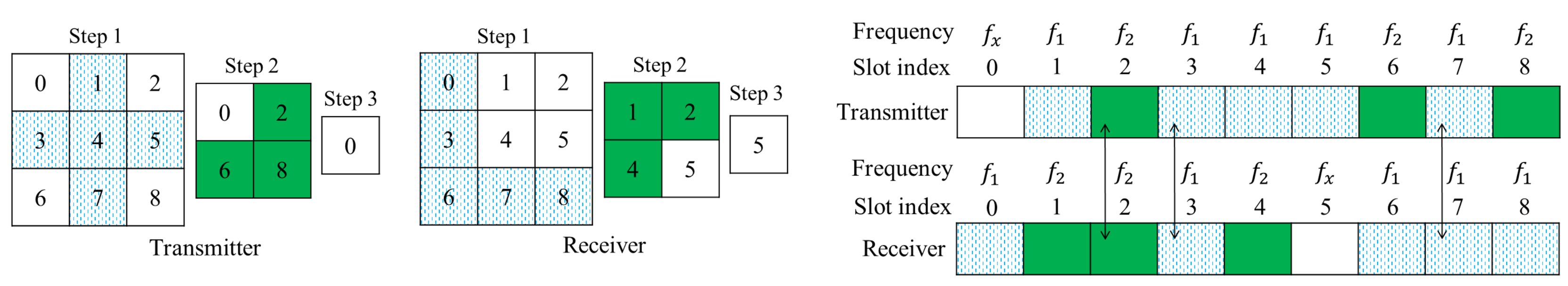
Example: Q = { {1,2,3,4,7}, {1,2,3,5,8}, {1,2,3,6,9}, {1,4,5,6,7}, {2,4,5,6,8}, {3,4,5,6,9}, {1,4,7,8,9}, {2,5,7,8,9}, {3,6,7,8,9} } is a GQS on {1, …, 9}.

Intersection property

Rotation closure property



Nested Grid-quorum-based Frequency Hopping Algorithm (NGQFH)



Problem Statement

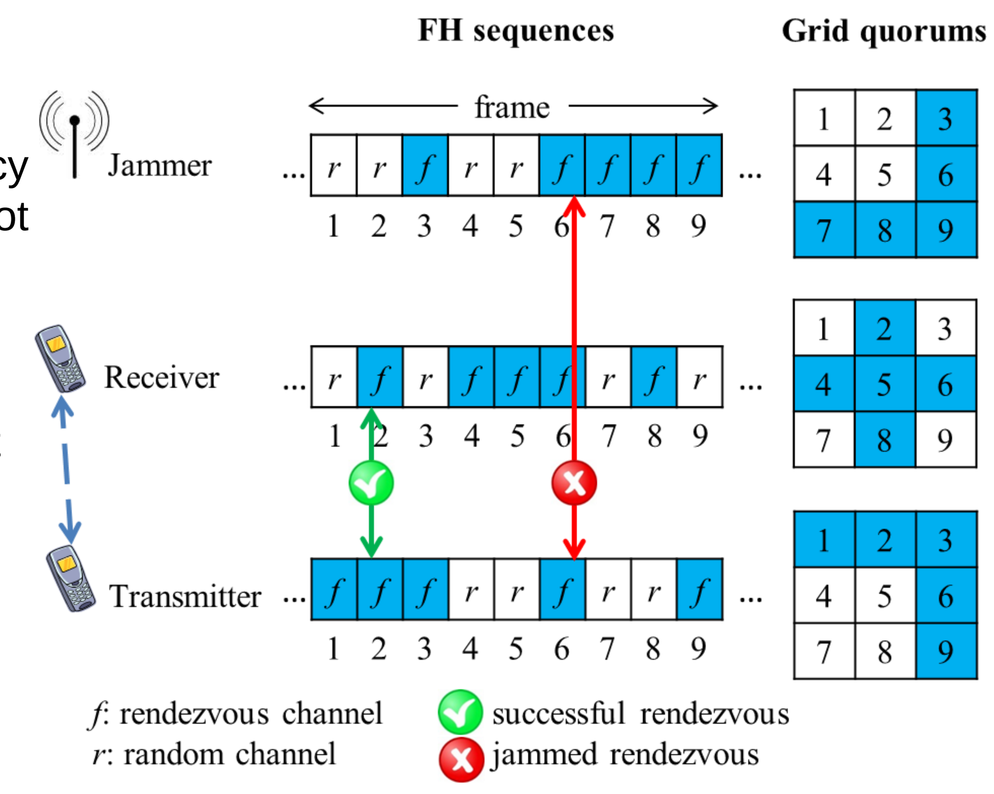
Two nodes aim at rendezvousing in the presence of an adversary.

System Model:

- Single link (unicast).
- Nodes operate in frequency hopping mode, with slot duration T.
- NGQFH algorithm is used.

Adversarial Model:

- Time-slotted jammer, with slot duration T.
- Jamming is carried out by a compromised node.
- Jammer is aware of the used NGQFH algorithm.



What does the jammer exactly know?

Frame length	Used grid quorum	Used rendezvous channel(s)
✓	✗	✓ ✗

Synchronous and asynchronous cases are considered.

Synchronous Rendezvous Over a Known Channel

Part I. Three-player Game

Players: Transmitter (T), Receiver (R), and Jammer (J)

Strategy: Which quorum to select

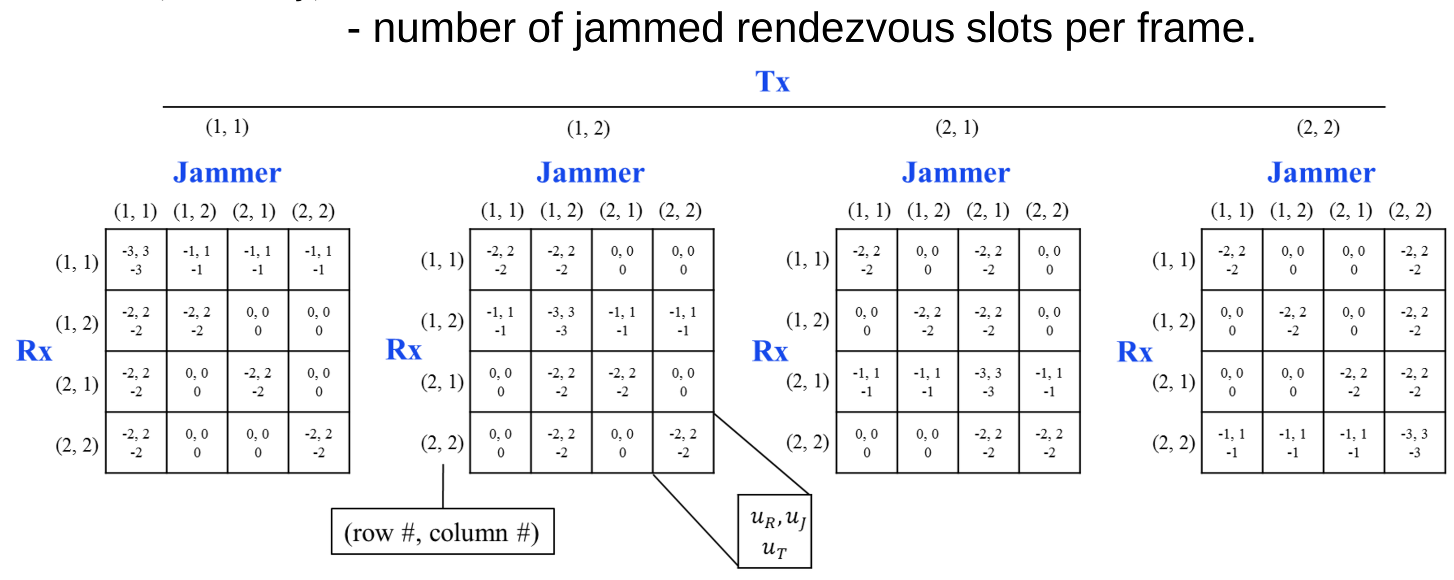
$s_T = (s_{T,r}, s_{T,c})$ T strategy

$s_R = (s_{R,r}, s_{R,c})$ R strategy

$s_J = (s_{J,r}, s_{J,c})$ J strategy

$\boxed{s_{T,r}, s_{R,r}, s_{J,r} \in S_r = \{1, 2, \dots, \sqrt{m}\}}$ $\boxed{s_{T,c}, s_{R,c}, s_{J,c} \in S_c = \{1, 2, \dots, \sqrt{m}\}}$

Utility: $u_T(s_T, s_R, s_J)$ = number of unjammed rendezvous slots per frame - number of jammed rendezvous slots per frame.



Theorem 1: The three-player game does not have a pure-strategy NE.

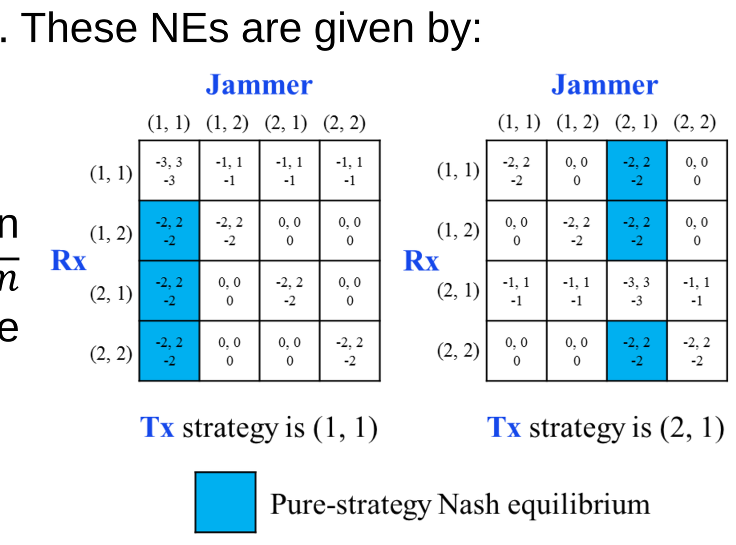
Part II. Two-player Game

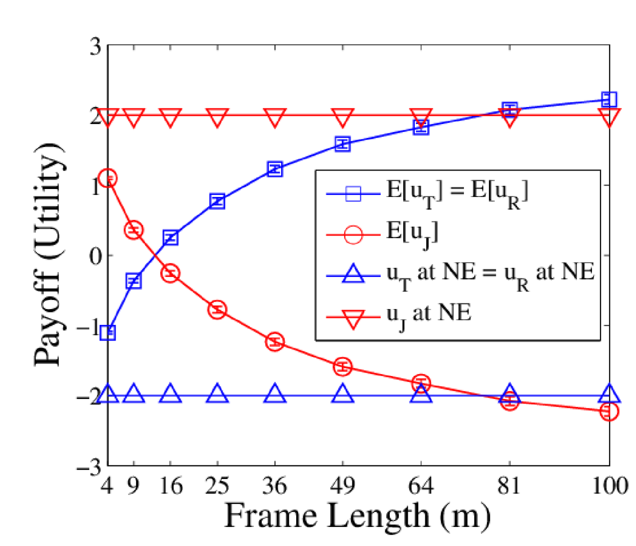
Theorem 2: For any $s_T = (s_{T,r}, s_{T,c})$, the $\sqrt{m} \times \sqrt{m}$ R/J game has at least $(\sqrt{m}-1)^2$ NEs, all of them result in $u_T = -2$. These NEs are given by:

$$s_{J,r} = s_{T,r},\ s_{J,c} = s_{T,c}$$
$$s_{R,r} \neq s_{T,r},\ s_{R,c} \neq s_{T,c}.$$

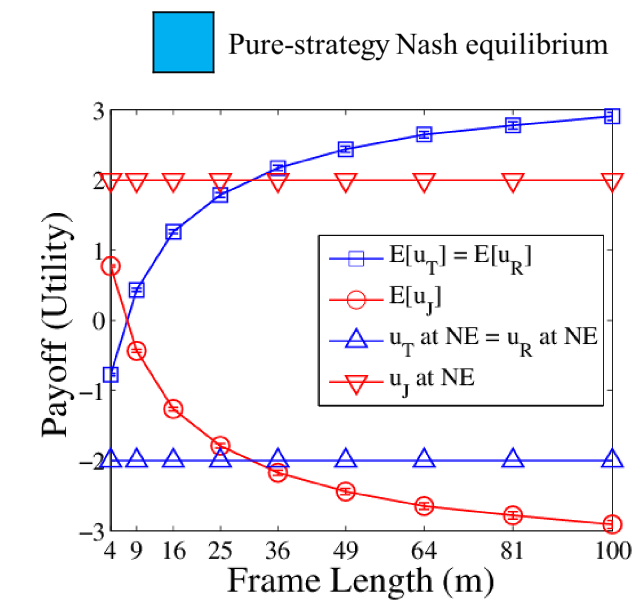
Proposition: The $(\sqrt{m}-1)^2$ NEs in Theorem 2 are the only NEs for the $\sqrt{m} \times \sqrt{m}$ game when $m \geq 9$. When $m = 4$, the game has additional NEs, given by:

$$s_{T,r} = s_{R,r} = s_{J,r} \text{ and } s_{J,c} = s_{T,c} \neq s_{R,c}$$
$$s_{J,r} = s_{T,r} \neq s_{R,r} \text{ and } s_{T,c} = s_{R,c} = s_{J,c}.$$





R and J have different beliefs about s_T



R and J have a common belief about s_T

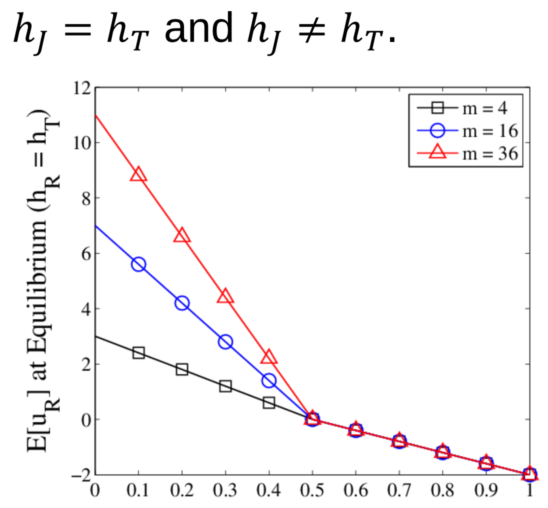
Synchronous Rendezvous Over an Unknown Channel (Bayesian Game)

Let h_T, h_R, and h_J denote the channels selected by T, R, and J, respectively. Then, R has two types: $h_R = h_T$ and $h_R \neq h_T$ and J has two types: $h_J = h_T$ and $h_J \neq h_T$.

Theorem 3: Let $p = \Pr\{h_J = h_T | h_R = h_T\}$, then the Bayesian NE of the above game is:

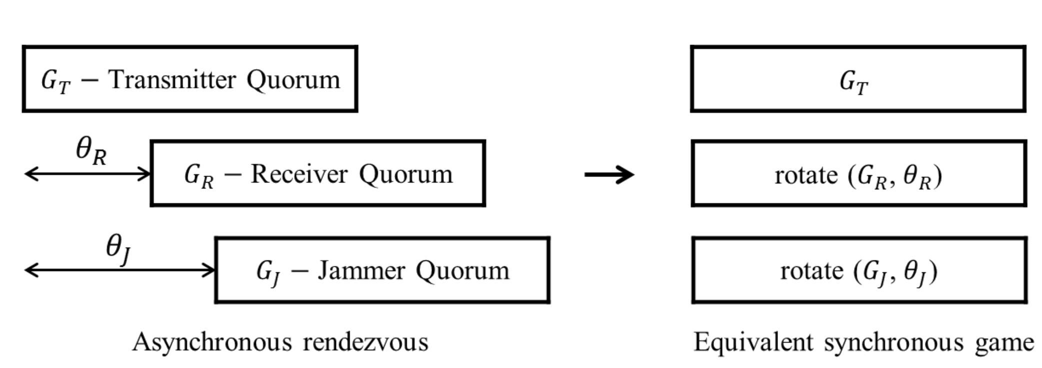
$$s_R \begin{cases} \begin{cases} = s_T, & \text{if } p < 0.5 \\ \neq s_T, & \text{if } p > 0.5, \\ \text{Does not matter}, & \text{if } p = 0.5 \end{cases} & \text{if } h_R = h_T \\ \text{Does not matter}, & \text{if } h_R \neq h_T \end{cases}$$

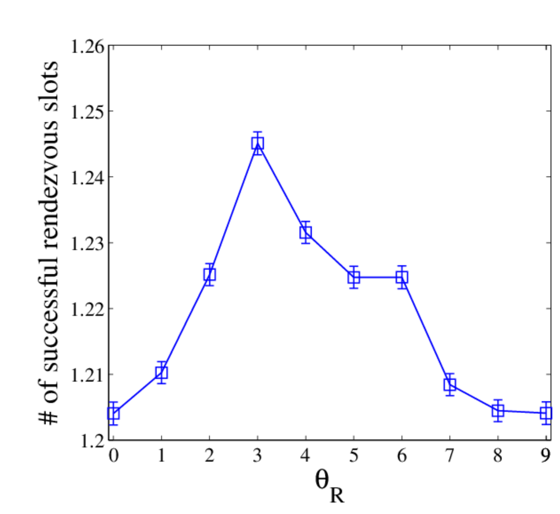
$$s_J \begin{cases} = s_T, & \text{if } h_J = h_T \\ \text{Does not matter}, & \text{if } h_J \neq h_T. \end{cases}$$



Asynchronous Rendezvous

The strategy of the player consists of a column and a sequence of $\sqrt{m}$ consecutive elements that do not necessarily form a row.





$\theta_J = 3$

Main Conclusions

Synchronous Case

1. R benefits from being, along with J, unaware of s_T. Furthermore, the benefits of R increase with the frame length.
2. It is beneficial for R if J has the same belief about s_T as it has.

Asynchronous Case

The number of successful rendezvous slots is maximized when $\theta_R = \theta_J$.

Ongoing/Future Work

1. Examine the more general case when the nesting degree is greater than one.
2. Design different sequential/parallel update mechanisms, including best-response update.
3. Study the convergence behavior of various updating mechanisms.
4. Consider other utility functions for the game formulation.
5. Consider the multicast rendezvous problem under smart jamming.