Blockchain-Enabled Byzantine Fault Tolerance Mechanism for Spectrum Sharing

Source Code: https://github.com/Qin99113/spectrumsharingblockchain

Xiaoxuan Qin

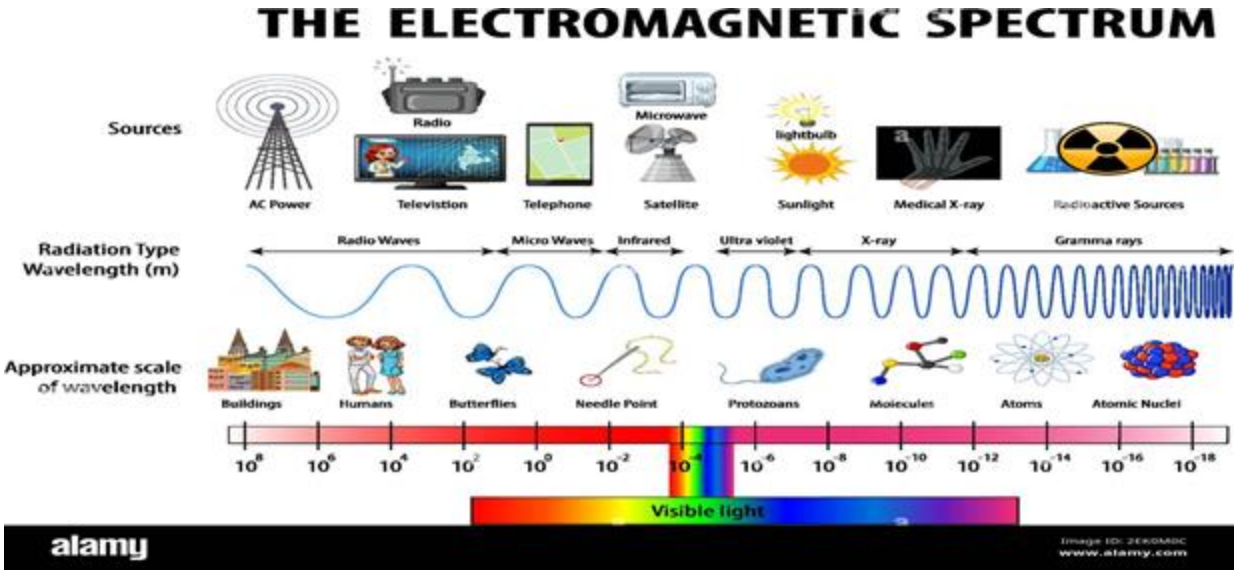
CS 3551: Adv Topic in Distributed System

University of Pittsburgh.

Background

Spectrum

Spectrum refers to the range of **electromagnetic wave frequencies**, widely used in wireless communication. It functions like "information highways," allowing devices to transmit and receive data over different frequency bands.



University of Pittsburgh.

Background

Spectrum Sharng

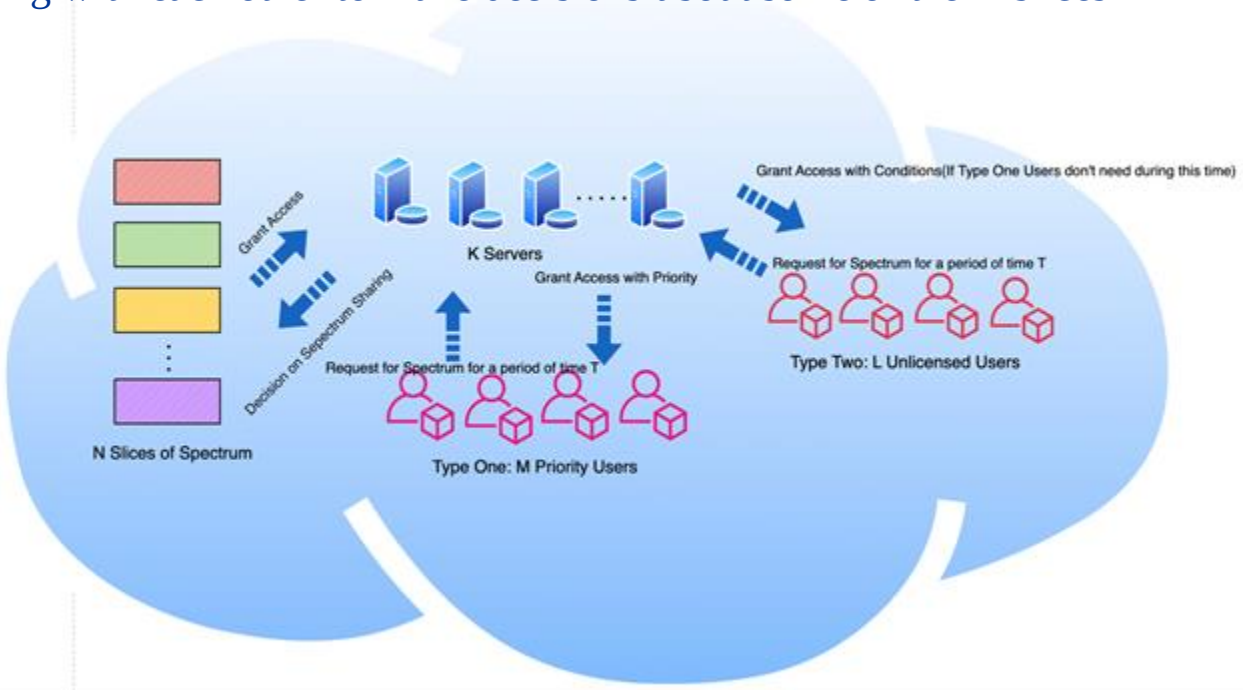
- In wireless communication systems, the spectrum is a limited resource that requires dynamic allocation among different users.

- Traditional static spectrum allocation methods result in low utilization efficiency, while spectrum sharing can dynamically enhance resource utilization.

Background

Decentralized Spectrum Sharing System

- Using environment sensing to determine whether a slice should be evacuated by a Type 2 user because a Type 1 user needs the spectrum
- Receiving notifications from Type 1 users about their impending need for spectrum slices
- Communicating with each other to make decisions about some of the N slices.



University of Pittsburgh.

Methodology

Reason

Why decentralized?

Fairness, Trust

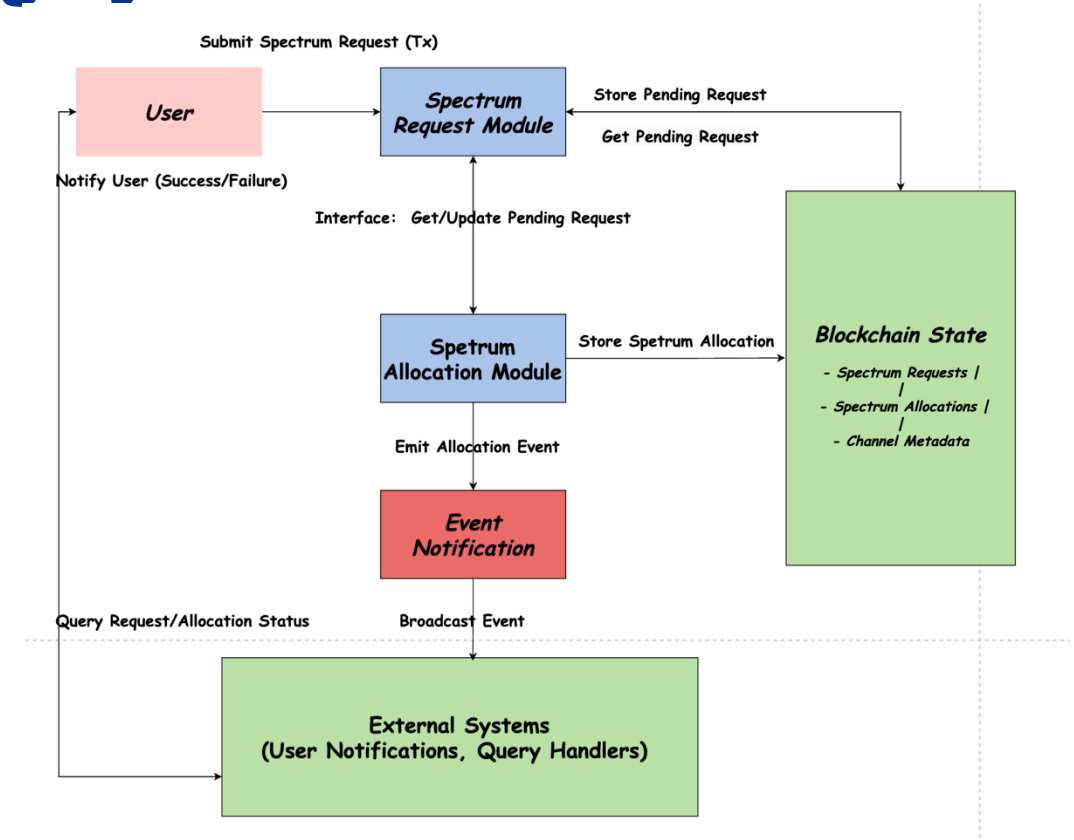
Why Blockchain? *(**weakness:** high energy consumption and processing speed issues)*

Transparency, Automation, and Robustness with PBFT

Bustamante, P., Gomez, M., Weiss, M., Murtazashvili, I., & Palida, A. (2023). **A Techno-Economic Study of Spectrum Sharing with Blockchain and Smart Contracts**. IEEE Communications Magazine, 61, 58-63. https://doi.org/10.1109/MCOM.001.2200317.

Pappa, P., Sarbhai, A., Baset, A., Kasera, S., & Buddhikot, M. (2020). S**pectrum Sharing in CBRS Using Blockchain.** 2020 IEEE 17th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), 631-639. https://doi.org/10.1109/MASS50613.2020.00082.

Overview of Blockchain-Enabled Spectrum Sharing System



University of Pittsburgh.

Methodology – Spectrum Request

Functions of Spectrum Request Module: *Create Request, Manage Status, Persistence, Module Interaction*

Field	*Type*	*Description*
Creator	*String*	*Address of the request creator*
Organization	*String*	*Organization to which the request belongs*
UserType	*String*	*Types of Users (AFC, LPI,…)*
Bandwidth	*Int32*	*Required bandwidth (MHz)*
Duration	*Int32*	*Duration of the required spectrum allocation (Seconds)*
BidAmount	*Coin*	*Bid Amount*
Status	*String*	*Status of the spectrum request (Pending, Failure and Success)*
RequestTime	*Int64*	*Time of the spectrum request*

RequestId	***Int64***	***ID of the spectrum request***

MsgCreateRequest* to *SpectrumRequest

University of Pittsburgh.

Methodology – Spectrum Allocation

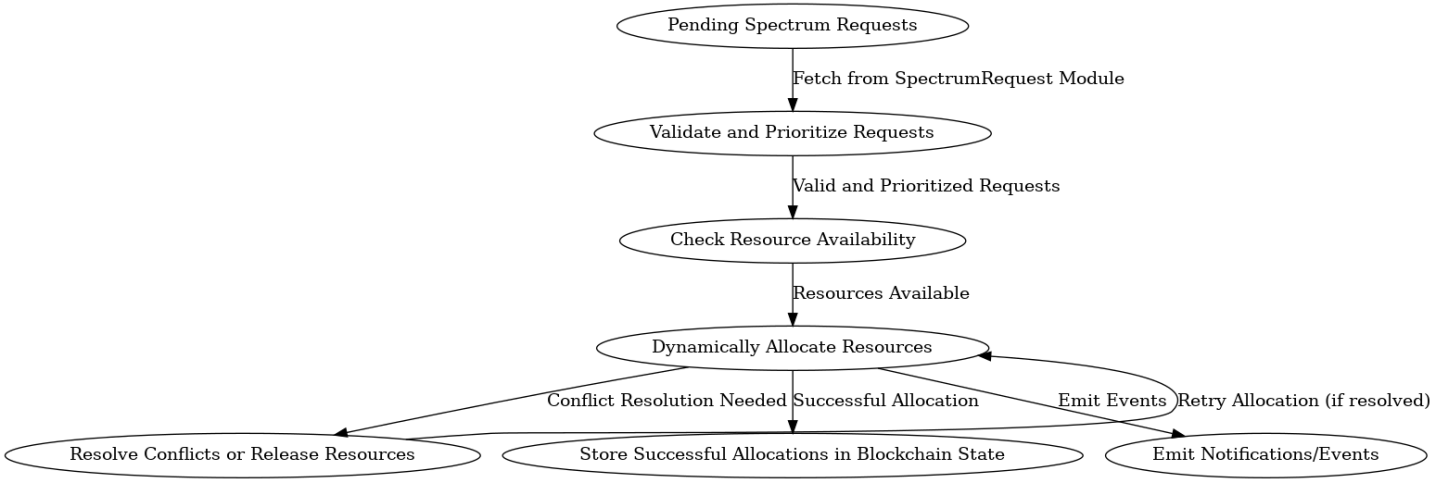
Field	*Type*	*Description*
AllocationId	*uint64*	*Automatically generated unique allocation record ID*
RequestId	*uint64*	*Associated request ID (inherited from SpectrumRequest)*
Creator	*string*	*Address of the allocated user (inherited from SpectrumRequest)*
Organization	*string*	*Organization to which the allocated user belongs (inherited from SpectrumRequest)*
User_type	*string*	*User type (e.g., SP, LPI)*
Channels	*repeated Channel*	*Information on allocated channels (supports multiple channels)*
Bandwidth	*int32*	*Total allocated bandwidth (MHz)*
Start_time	*int64*	*Start time of the allocation*
End_time	*int64*	*End time of the allocation*
Priority	*int32*	*Dynamically computed priority*
Status	*string*	*Status of the allocation: Active, Released, Pending*
AllocationType	*string*	*Type of allocation (e.g., Auction, Manual, Dynamic)*

```
allocation := types.SpectrumAllocation{
    AllocationId:   1,
    RequestId:      1,
    Creator:        "",
    Organization:   "",
    UserType:       "",
    Channels:       []*types.Channel{{Id: 1, Frequency: 6000, Bandwidth: 20, ChannelStatus: "Allocated"}},
    Bandwidth:      20,
    StartTime:      ctx.BlockHeader().Time.Unix() - 3600,
    EndTime:        ctx.BlockHeader().Time.Unix() - 1800,
    Priority:       10,
    Status:         "Active",
    AllocationType: "Auto",
}
```

University of Pittsburgh.

Methodology – Spectrum Allocation

- Initialize Spectrum Channels
- Fetch Pending Requests
- Priority-Based Allocation, Conflict Resolution, Expired Allocation Management
- Allocate Spectrum Resources
- Handle Allocation Failures, Event Notification, State Updates



University of Pittsburgh.

Methodology

Spectrum Sharing in 6GHz wifi-6E and 6G devices

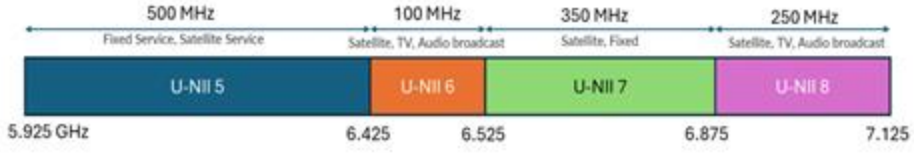


Figure 1: The allocation of spectrum in the 6 GHz bands in the US

```go
func (k Keeper) InitializeChannels(ctx sdk.Context) {
    channels := []types.Channel{}
    // 6 GHz band from 5925 MHz to 7125 MHz
    bandwidth := 20   // Bandwidth in MHz
    startFreq := 5925 // Starting frequency in MHz
    endFreq := 7125   // Ending frequency in MHz

    // Initialize channels in the 6 GHz band
    channelId := 0
    for freq := startFreq; freq < endFreq; freq += bandwidth {

        var status string
        var allowedUsers []string

        // Ensure the channel frequency is valid
        if freq+bandwidth > endFreq {
            break
        }
        switch {
        // U-NII-5 (5.925–6.425 GHz) and U-NII-7 (6.525–6.875 GHz)
        // These require AFC for SP users, and VLP users have no restrictions
        case (freq >= 5925 && freq < 6425) || (freq >= 6525 && freq < 6875):
            status = "Available"
            allowedUsers = []string{"SP", "LPI", "VLP"}
        // U-NII-6 (6.425–6.525 GHz) and U-NII-8 (6.875–7.125 GHz)
        // These are restricted to low-power indoor users only
        case (freq >= 6425 && freq < 6525) || (freq >= 6875 && freq < 7125):
            status = "Low Power Indoor Only" // LPI users only
            allowedUsers = []string{"LPI"}   // Only LPI users
        // Any undefined or protected frequency
        default:
            status = "Protected"      // Mark as protected or unavailable
            allowedUsers = []string{} // No users allowed
        }
```

Channel Distribution

LPI(Low Power Indoor) will be available in all 6GHz bands.

SP(Standard Power) and VLP(Very Low Power) are only available in U-NII 5 and U-NII 7

University of Pittsburgh.

Methodology

Spectrum Sharing in 6GHz wifi-6E and 6G devices

- **Priority Calculation**
- **User type and Organization priorities**, emphasizing public safety and regulated usage.
- **Bandwidth efficiency**, encouraging low-bandwidth usage.
- **Duration penalties**, preventing monopolization of the spectrum.
- **Bid normalization**, balancing monetary incentives with regulatory needs

Algorithm 1 CalculatePriority: Compute the priority for a spectrum request

Require: SpectrumRequest *request*
Ensure: Computed priority *priority*

Initialize $basePriority \leftarrow 0$
▷ Step 1: Add user type weight
if $request.UserType =$ "SP" **then**
  $basePriority \leftarrow basePriority + 100$
else if $request.UserType =$ "LPI" **then**
  $basePriority \leftarrow basePriority + 90$
else if $request.UserType =$ "VLP" **then**
  $basePriority \leftarrow basePriority + 70$
else
  $basePriority \leftarrow basePriority + 50$
end if
▷ Step 2: Add organization weight
$organizationPriority \leftarrow$ GetOrganizationPriority($request.Organization$)
$basePriority \leftarrow basePriority + organizationPriority$
▷ Step 3: Subtract bandwidth penalty
$basePriority \leftarrow basePriority - \left\lfloor \frac{request.Bandwidth}{10} \right\rfloor \cdot 5$
▷ Step 4: Subtract duration penalty
$durationPenalty \leftarrow \left\lfloor \frac{request.Duration}{3600} \right\rfloor \cdot 10$
if $durationPenalty > 100$ **then**
  $durationPenalty \leftarrow 100$
end if
$basePriority \leftarrow basePriority - durationPenalty$
▷ Step 5: Add bid amount normalization
$bidPriority \leftarrow \left\lfloor \frac{request.BidAmount.Amount}{1000} \right\rfloor$
if $bidPriority > 200$ **then**
  $bidPriority \leftarrow 200$
end if
$basePriority \leftarrow basePriority + bidPriority$
▷ Ensure priority is non-negative
if $basePriority < 0$ **then**
  $basePriority \leftarrow 0$
end if
return *basePriority*
procedure GetOrganizationPriority(*organization*)
  if *organization* = "Government" **then**
    return 50
  else if *organization* = "EmergencyServices" **then**
    return 100
  else if *organization* = "Commercial" **then**
    return 30
  else if *organization* = "NonProfit" **then**
    return 20
  else
    return 10
  end if
end procedure

University of Pittsburgh.

Methodology
Auto Spectrum Allocation Algorithm

- Retrieve and and Prioritize Pending Requests:
- Iterate Through Pending Requests
- Handle Conflicts
- Attempt Channel Allocation
- Successful Allocation
- Save Allocation
- Update Request Status
- Finalize and Log

Input : Pending spectrum requests
Output: Allocated spectrum resources
// Retrieve and sort pending requests by priority
pendingRequests ← *GetAllPendingSpectrumRequests*()
Sort(*pendingRequests* ByDescending(CalculatePriority)
ThenByAscending(*RequestTime*))
foreach *request* ∈ *pendingRequests* **do**
  // Check for allocation conflicts
  conflictingAllocations ← *CheckConflictingAllocations*(*request*)
  // Attempt to resolve conflicts
  successConflictResolution ←
   ReleaseLowPriorityAllocations(*request*, *conflictingAllocations*)
  if ¬*successConflictResolution* **then**
    request.Status ← "*Failed*"
     LogError("*Allocationconflictunresolvable*")
  end
  // Attempt channel allocation
  allocatedChannels ← *AllocateChannels*(*request*)
  if *allocatedChannels* = ∅ **then**
    // Create failed allocation record
    allocation ← *CreateFailedAllocation*(*request*)
     EmitEvent("*AllocationFailed*")
  end
  // Create successful allocation record
  allocationID ← *GenerateUniqueID*()
   priority ← *CalculatePriority*(*request*)
  allocation ←
   CreateAllocation(*allocationID*, *request*, *allocatedChannels*, *priority*)
  // Save and update records
  SaveAllocation(*allocation*) *request.Status* ← "*Allocated*"
  // Emit success event
  EmitEvent("*AllocationSuccessful*")
   LogSuccess(*request*, *allocation*)
end

Algorithm 1: Auto Spectrum Request Allocation

University of Pittsburgh.

Implementation

- Ignite CLI(Latest Version)
- Programming Language: Go

```
sherleyqin@Sherleys-MacBook-Pro spectrumsharingblockchain % ignite chain serve

  Blockchain is running

  👤 alice's account address: cosmos1cw2kqr252lt5p4ny53rfj8hrgs7jl4lvd20x4h
  👤 bob's account address: cosmos1kpeag6fkcmwezgj025ntalynphv9ha0wwfnqrp

  🌍 Tendermint node: http://0.0.0.0:26657
  🌍 Blockchain API: http://0.0.0.0:1317
  🌍 Token faucet: http://0.0.0.0:4500

  ⭐ Data directory: /Users/sherleyqin/.spectrumSharingBlockchain
  ⭐ App binary: /Users/sherleyqin/go/bin/spectrumSharingBlockchaind
```

University of Pittsburgh

Test – Spectrum Request

```
=== RUN   TestMsgServer
Response: status:"success" message:"Request created successfully with ID: 1"
Request: {Id:1 Creator:cosmos1rkrctacqxv4mcxz4ymvpv30yl26lsuxgzt8kc4 Organization:Test Organiza
tion UserType:SP Bandwidth:50 Duration:10 BidAmount:1000token Status:Pending RequestTime:123456
7890}
--- PASS: TestMsgServer (0.00s)
PASS
ok      spectrumSharingBlockchain/x/spectrumrequest/keeper      2.305s
```

Test – Channel

```
    /Users/sherleyqin/spectrumsharingblockchain/x/spectrumallocation/keeper/sepctrumallocation_
 test.go:36: Channel 58: ID=7, Frequency=6065 MHz, Bandwidth=20 MHz, Status=Available
    /Users/sherleyqin/spectrumsharingblockchain/x/spectrumallocation/keeper/sepctrumallocation_
 test.go:36: Channel 59: ID=8, Frequency=6085 MHz, Bandwidth=20 MHz, Status=Available
    /Users/sherleyqin/spectrumsharingblockchain/x/spectrumallocation/keeper/sepctrumallocation_
 test.go:36: Channel 60: ID=9, Frequency=6105 MHz, Bandwidth=20 MHz, Status=Available
 --- PASS: TestInitializeChannels (0.00s)
 PASS
 ok      spectrumSharingBlockchain/x/spectrumallocation/keeper   2.328s

Running tool: /usr/local/go/bin/go test -timeout 30s -run ^TestGetChannel$ spectrumSharingBlock
chain/x/spectrumallocation/keeper

=== RUN   TestGetChannel
--- PASS: TestGetChannel (0.00s)
PASS
ok      spectrumSharingBlockchain/x/spectrumallocation/keeper   2.266s


Running tool: /usr/local/go/bin/go test -timeout 30s -run ^TestReleaseChannels$ spectrumSharing
Blockchain/x/spectrumallocation/keeper

=== RUN   TestReleaseChannels
--- PASS: TestReleaseChannels (0.00s)
PASS
ok      spectrumSharingBlockchain/x/spectrumallocation/keeper   2.289s
```

Test – Spectrum Allocation

```
Running tool: /usr/local/go/bin/go test -timeout 30s -run ^TestGetAllSpectrumAllocations$ spect
rumSharingBlockchain/x/spectrumallocation/keeper

=== RUN   TestGetAllSpectrumAllocations
--- PASS: TestGetAllSpectrumAllocations (0.00s)
PASS
ok      spectrumSharingBlockchain/x/spectrumallocation/keeper   2.401s


Running tool: /usr/local/go/bin/go test -timeout 30s -run ^TestReleaseExpiredAllocations$ spect
rumSharingBlockchain/x/spectrumallocation/keeper

=== RUN   TestReleaseExpiredAllocations
--- PASS: TestReleaseExpiredAllocations (0.00s)
PASS
ok      spectrumSharingBlockchain/x/spectrumallocation/keeper   2.244s
```

University of Pittsburgh

Test – Spectrum Allocation

```
aringBlockchain/x/spectrumallocation/keeper

=== RUN   TestAutoAllocateRequests
Debug - Retrieved SpectrumRequest: {Id:2 Creator:cosmoslkpeag6fkcmwezgj025ntalynphv9hagwwfngrp
Organization:TestOrg2 UserType:SP Bandwidth:40 Duration:3600 BidAmount:1000token Status:Pending
 RequestTime:1234567890} true
--- PASS: TestAutoAllocateRequests (0.00s)
PASS
ok      spectrumSharingBlockchain/x/spectrumallocation/keeper   2.321s
```

- On Going

University of Pittsburgh.

Future Work

- Debug
- Evaluation: Robustness, Resilience
- Dynamic Spectrum Allocation(FCC Policies)

…