

# Hopskotch and Hackathon

Amanda Depoian

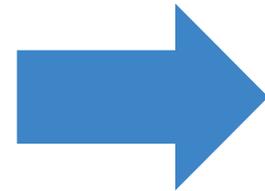
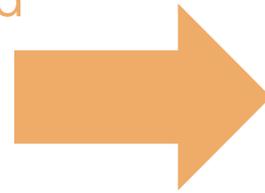
[adepoian@purdue.edu](mailto:adepoian@purdue.edu)

May 12, 2021

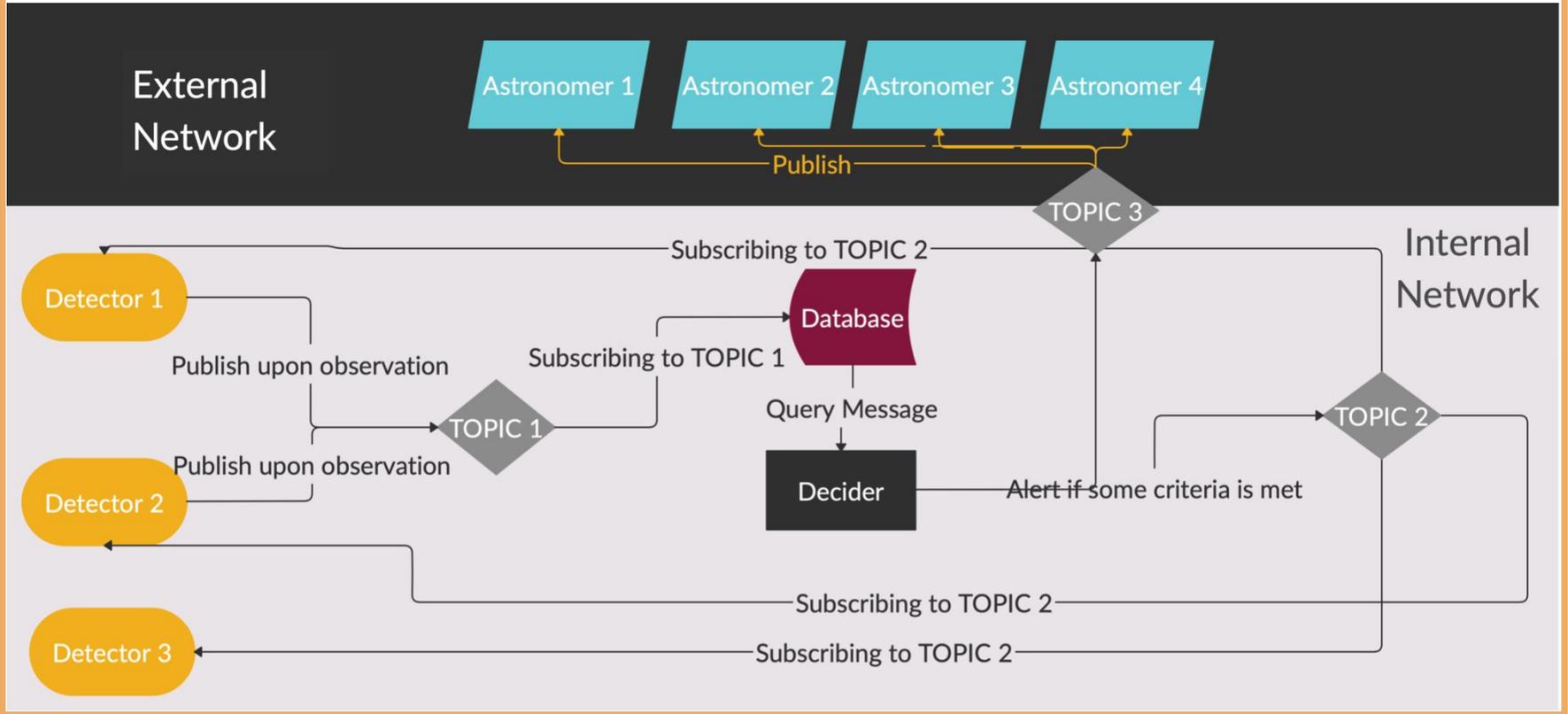
# SNEWS 2.0

Essential components:

- 1) Function to read messages and decide if there's an alert
- 2) Message interfacing for clients
- 3) Server infrastructure to coordinate messaging



# SNEWS 2.0 prototype: *snews* + Hopskotch



# SNEWS 2.0 prototype: *snews* + Hopskotch

External  
Network

Astronomer 1

Astronomer 2

Astronomer 3

Astronomer 4

Publish

Detector 1

Publish upon observation

Subscribing to TOPIC 2

Internal  
Network

Detector 2

Publish upon observation

Subscribing to TOPIC 1

Database

Query Message

Decider

Alert if some criteria is met

Detector 3

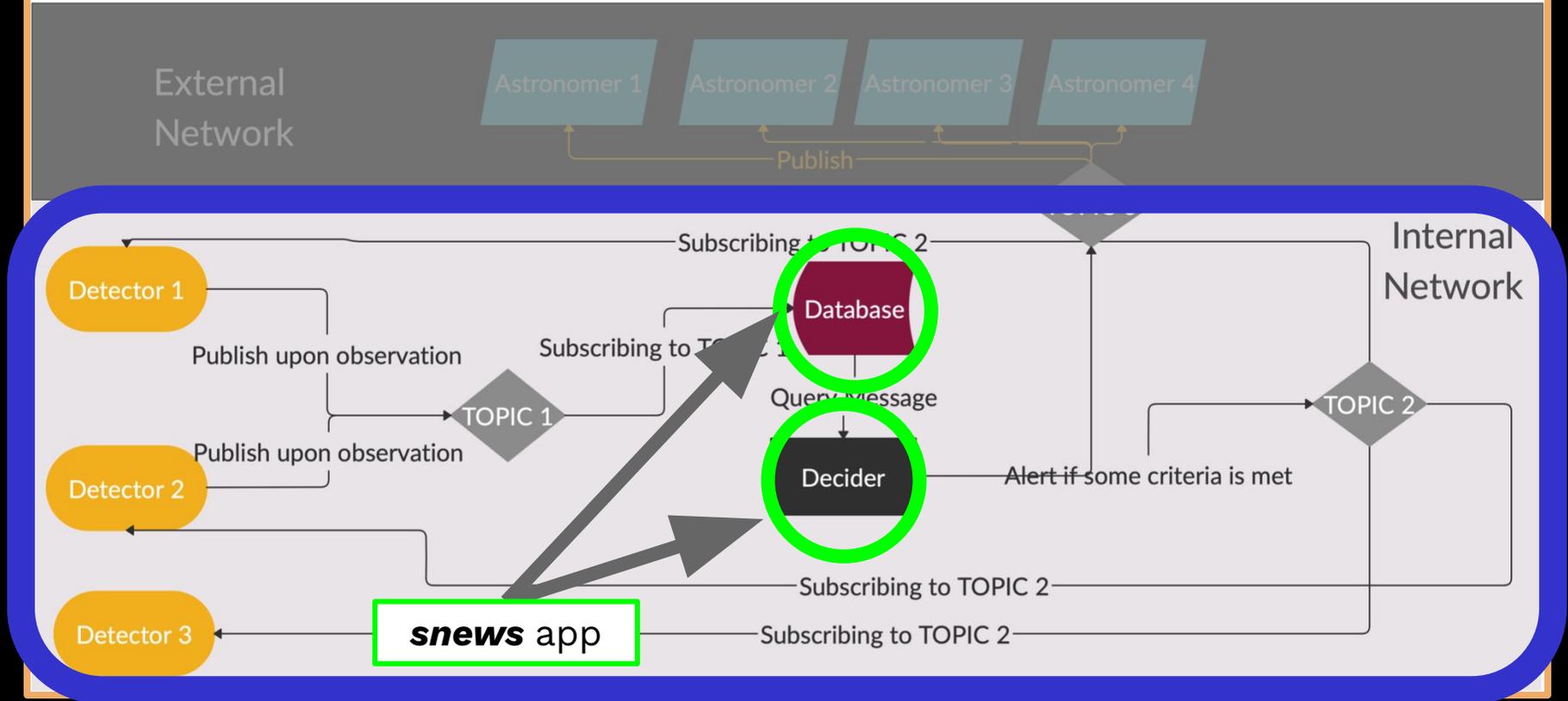
Subscribing to TOPIC 2

Subscribing to TOPIC 2

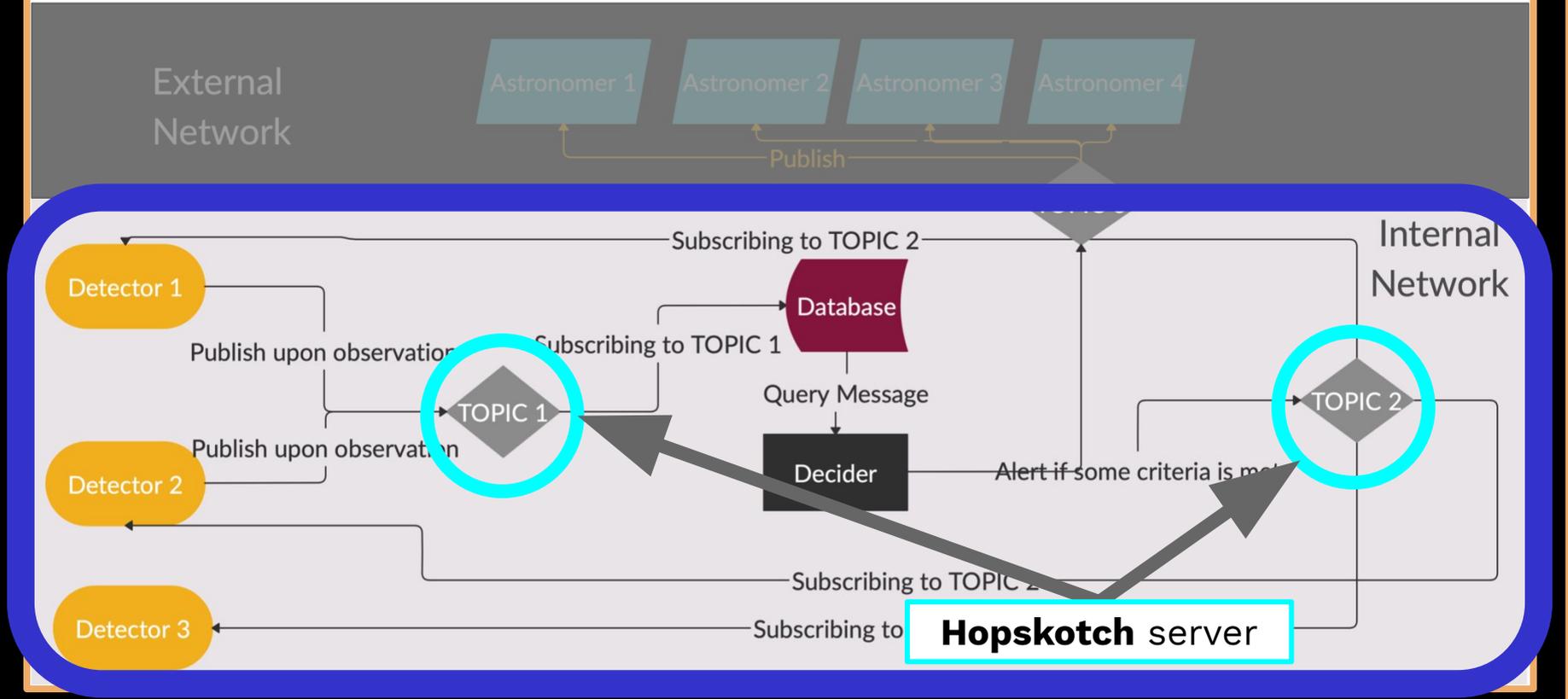
TOPIC 1

TOPIC 2

# SNEWS 2.0 prototype: *snews* + Hopskotch



# SNEWS 2.0 prototype: *snews* + Hopskotch



# Hackathon: Goals

1. Connect all experiments into SNEWS 2.0 with SNEWS 1.0 functionality.
  - Publish messages to “Topic 1”
  - Subscribe to messages from “Topic 2”
2. Add SNEWS 2.0 functionality.

# SNEWS 1.0 Functionality



# Messages

## Observation Message

- sent by a detector to SNEWS 2.0 server upon observation of possible supernova neutrinos

## Heartbeat Message

- sent by a detector regularly to SNEWS 2.0 server to tell they are online & “we saw nothing”

## Alert Message

- Sent by SNEWS 2.0 server to all detectors when coincidence requirement is met

# SNEWS 2.0 Functionality



# Hackathon: What to expect

## Daily Check-ins!

- Quick status updates to discuss progress and issues. By doing the same tasks we can hopefully help each other out and share what we learn.

## Lecture Series!

- Physics lectures to expand the knowledge of supernova and neutrino physics. Great for non-snews members → [Bring a friend!](#)
- Software lectures will allow participants to learn the different aspects of SNEWS 2.0 beyond hopskotch.

# **Backup Slides**

# SNEWS 2.0 prototype: *snews* app

- **storage**: store messages in MongoDB server for a variable length of time
  - Provides portable client-side message caching
- **decider**: instantiates *storage*, adds messages to cache, and applies deciding logic to determine possibility of a supernova event
  - Deciding logic involves checking time-coincidence and detector status
- **model**: instantiates a *decider* and connects it to message streams
  - *decider* parses/stores messages; *model* listens to the *decider* and sends an alert message if *decider* determines a supernova event
- Uses SCiMMA's hop-app stream API to connect with Hopskotch servers

# Internal Experiment Data Sharing



## Alert Tier

Send an alert when more events than background detected. This includes the current detector data-taking mode.



## Significance Tier

Send signal significance periodically or when the significance changes rapidly.



## Timing Tier

Send time series or time distribution of events for each channel.