

RxPatterns Design Methodology

Objective: To Visualise the flow of notifications through the system.

Representation

Consumers and producers are represented as nodes.

Notification pathways are represented as connections..

You visualiase

A decomposition of intercatons

A  
Color coded node roles

Allows you to

Identify Inputs and Outputs

Identify Notification types and their use

Plan for pathway bandwidth limits

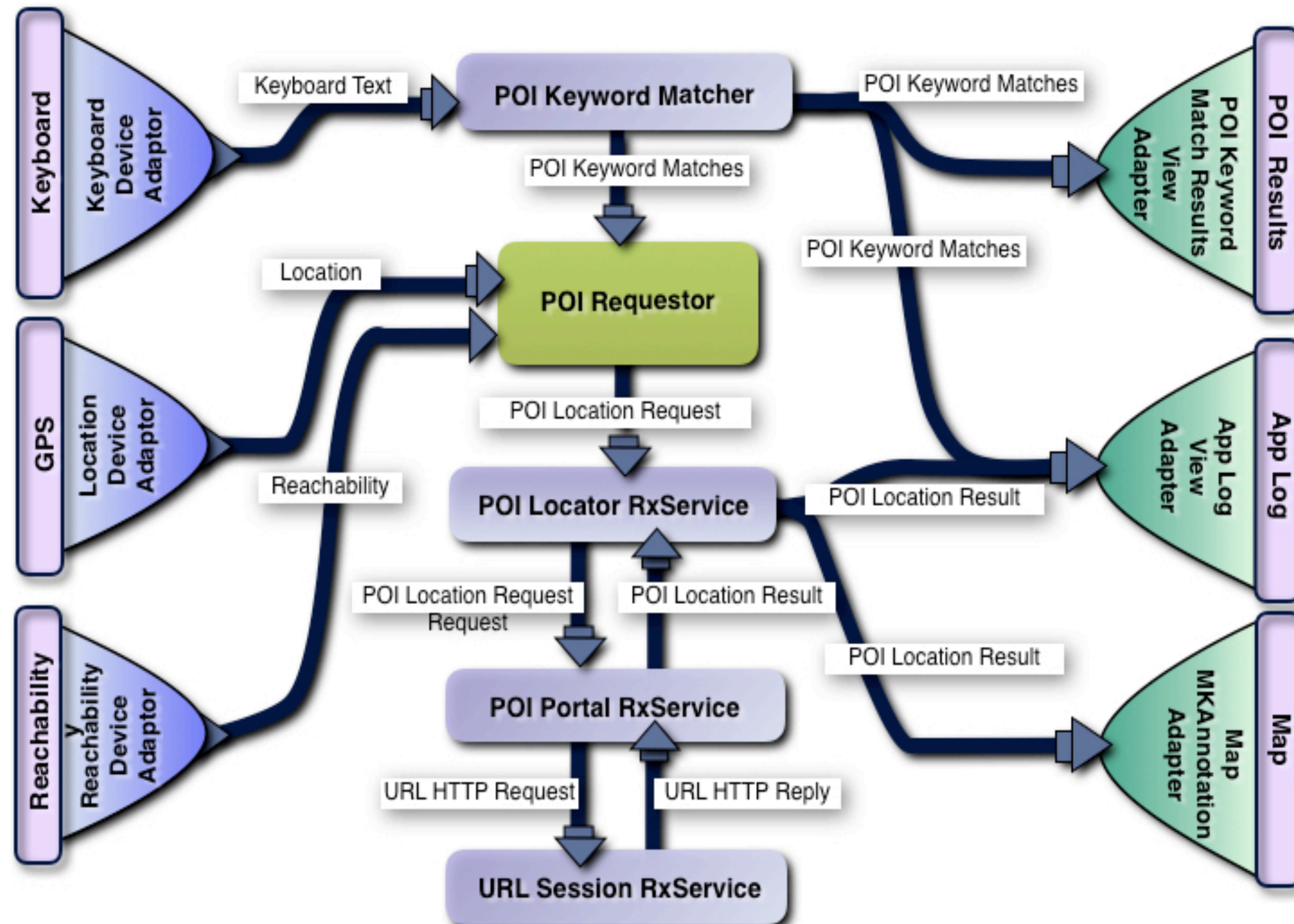
Identify core pathways and marginal pathways

Describe it to non-technical interested parties.

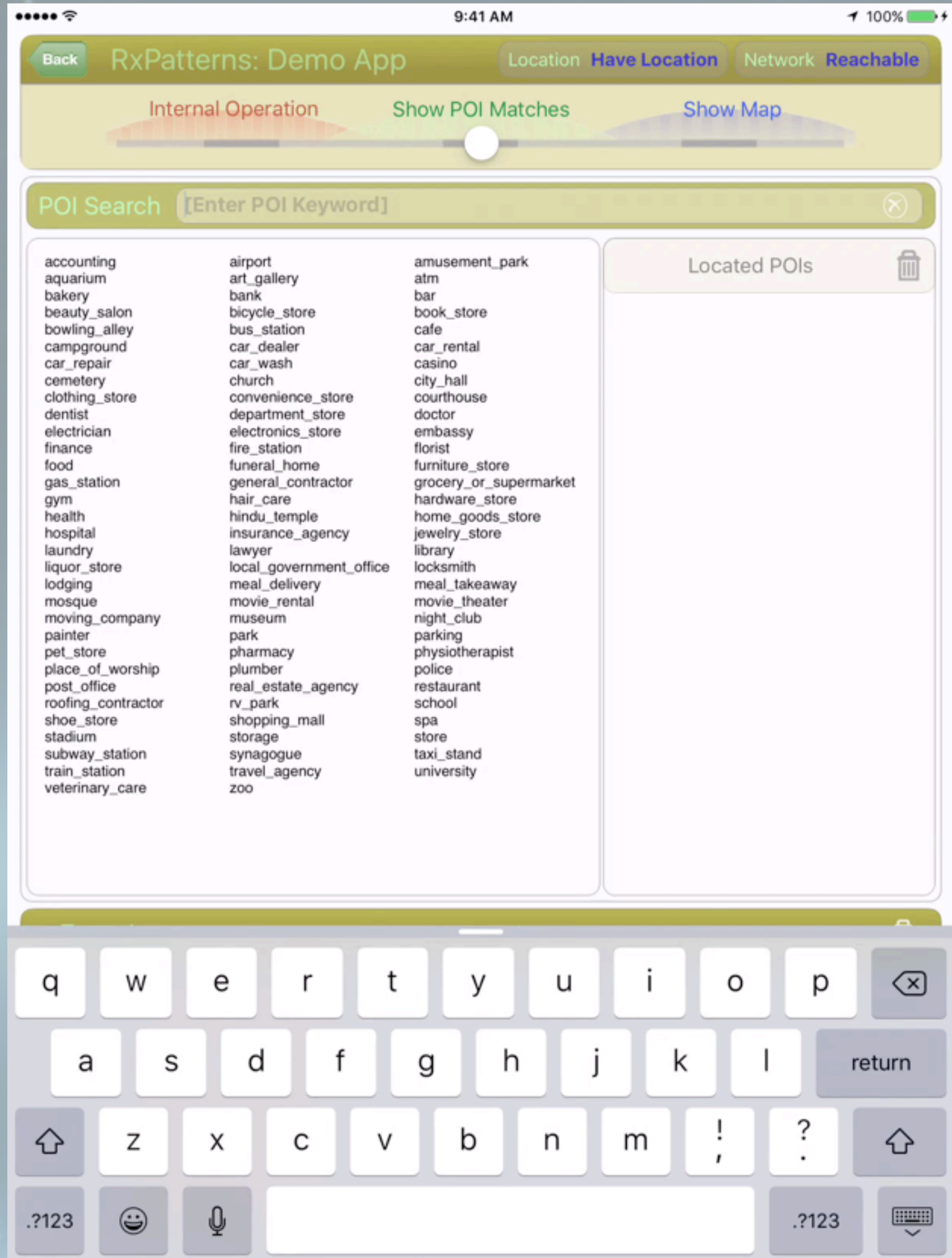


## Live App Operation Visualisation

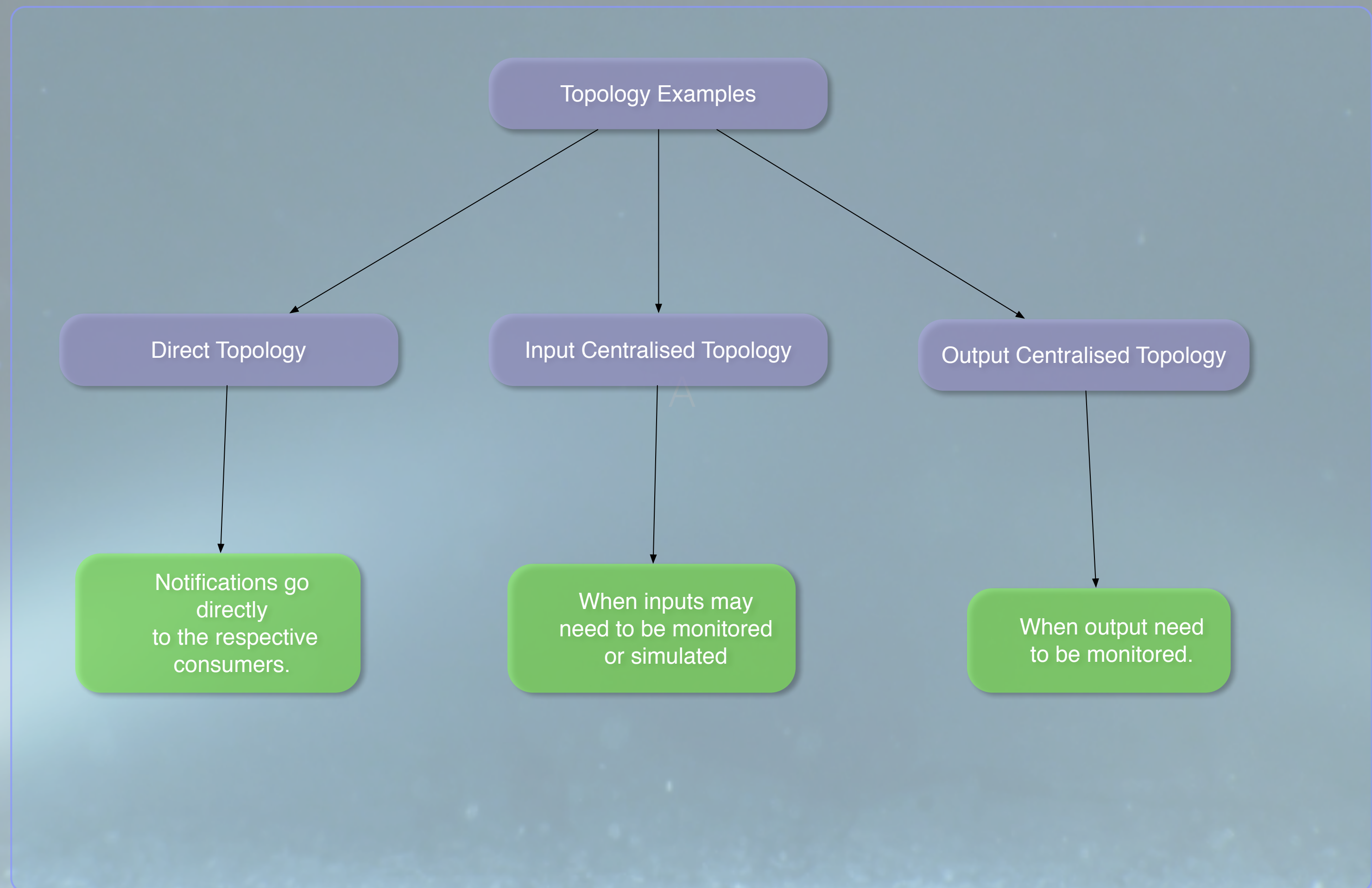
Visualisation of Event Notifications Flowing Through The App - enter text to observe







## Connection Topologies





## Benefits Of RxPatterns

## Concept Abstraction

Concepts of: Consumers, Producers, Notifiers, EvalQueue, Notification

Abstraction aides in design visualisation and design terminology.

## RxPatterns is a Design SDK (not an implementation library)

You develop your own operators (if required). Handle your specific error conditions and optimise for your specific conditions.

The Toolkit has the source implementation for practically all the standard MS Operators for use or as a source reference for your custom operators.

You can develop your own notifiers.

## Use of EvalQueues rather than threading

No locks required - cleaner code (possibly faster).

RxEvalQueue has thread deadlock detection - no lock up (unless threads are directly blocked)

## Internal operations are monitorable

Operations are instrumented, they can be viewed or logged. You can write your own monitor handler

## Multi Platform

On swift platforms which also support libdispatch (currently OS/X, iOS and linux coming)