Mobile & Pervasive Computing Group

EXCELLENCE IN DISTRIBUTED GLOBAL ENVIRONMENTS

PAQ: Persistent Adaptive Query Middleware for Dynamic Environments

Vasanth Rajamani, Christine Julien
The University of Texas at Austin

Jamie Payton
The University of North Carolina, Charlotte

Catalin Roman
Washington University in Saint Louis

Mobile Pervasive Computing

- Variety of devices being embedded in the environment
- Convergence of computation, communication and control

Promise and Challenges

- Promise
  - Monitor and react to changes in the physical world in real time
- Reality
  - Ad hoc and fluid infrastructure
- Challenge
  - Programming applications that monitor and react to information across an open and dynamic network is difficult

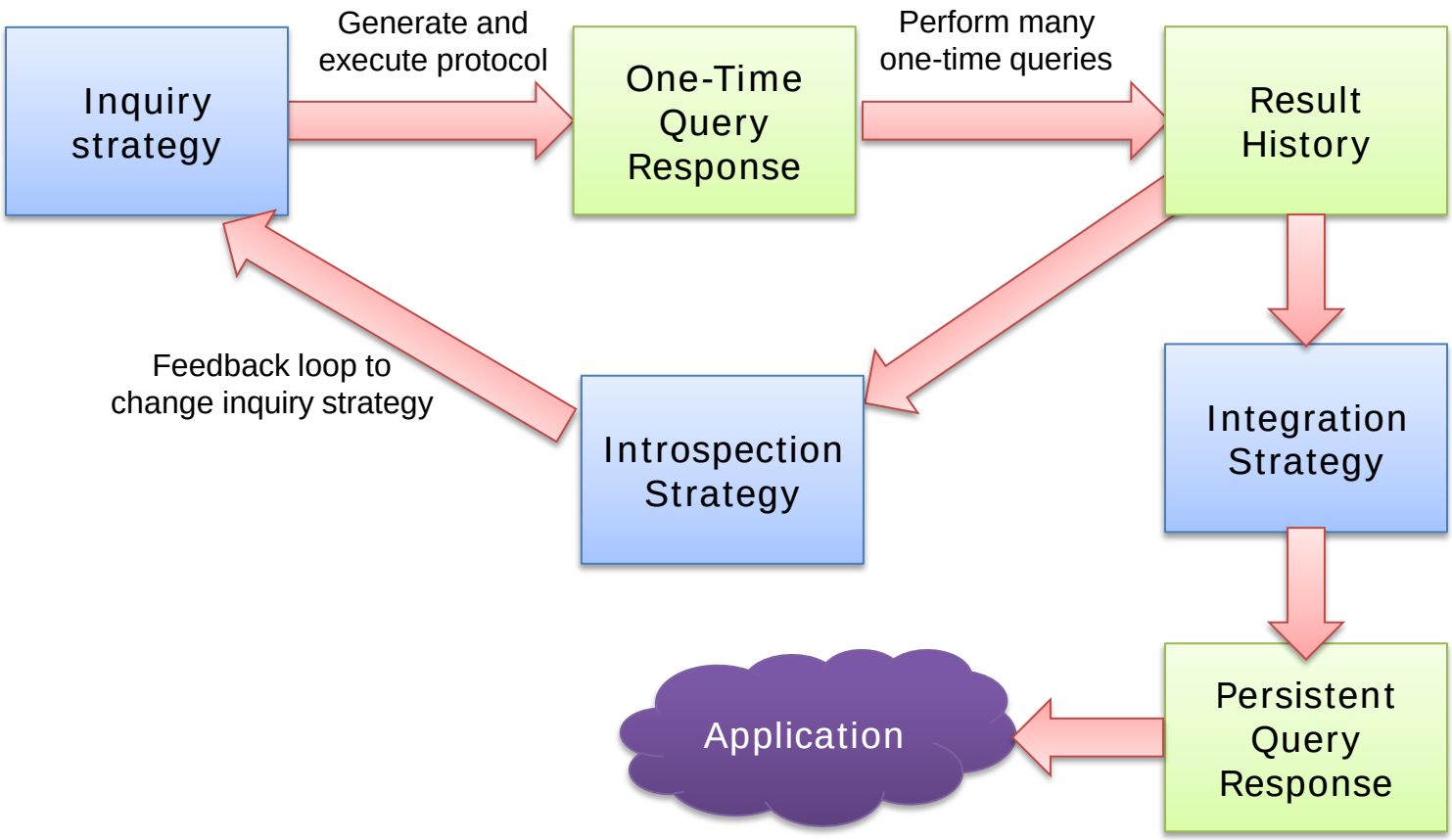
Query Based Application Development

- Query: Abstraction that allows specification of interest without worrying about network details
- Pervasive applications need information about the environment
- Application development through progressive query processing
  - Persistent Queries

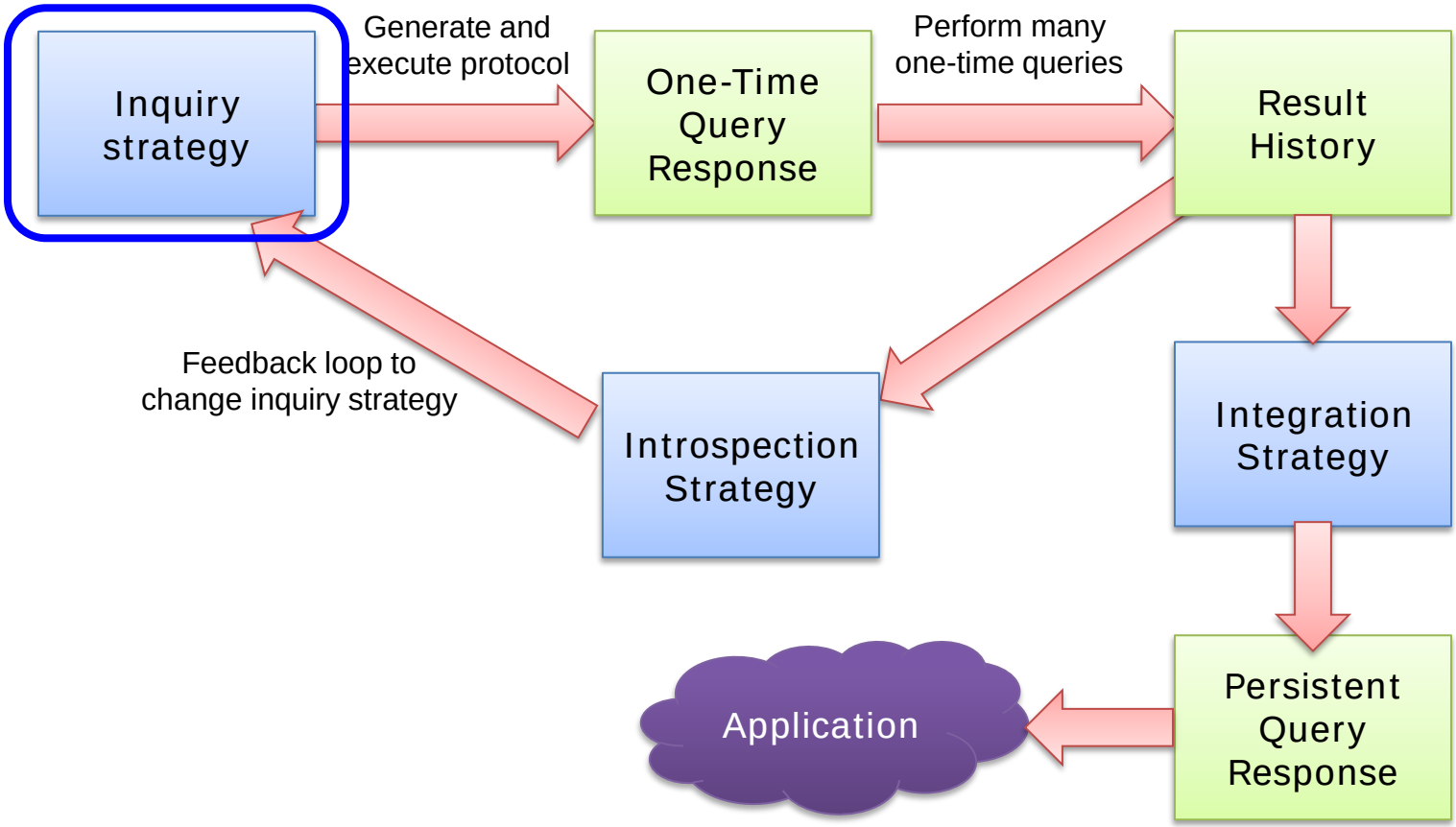
Persistent Queries

- Persistent Query: Continuous report of state change
  - Capturing a global consistent view is infeasible in practice
- Need to generate an approximation that is both reasonable and cost-effective
- PAQ: A middleware to compute the results of persistent query from a sequence of one-shot query results

PAQ Middleware Overview



PAQ Middleware Overview

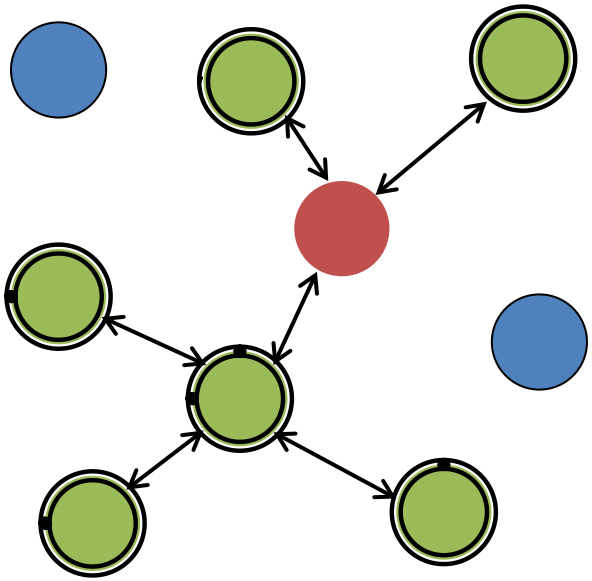


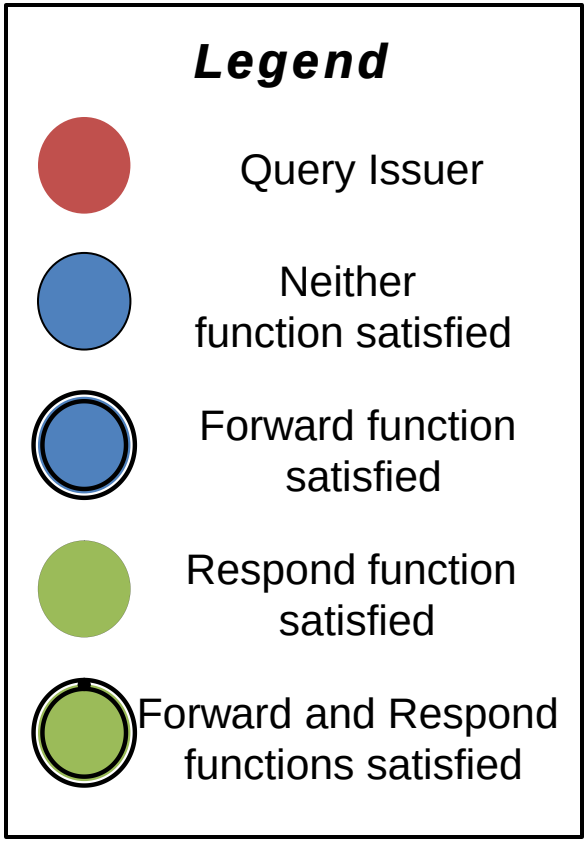
Inquiry Strategy

- Collecting Information
  - Protocol employed
  - Frequency of response

- Flooding inquiry strategy
  - Every receiver responds
  - Every receiver forwards

$$I = (true, true)$$



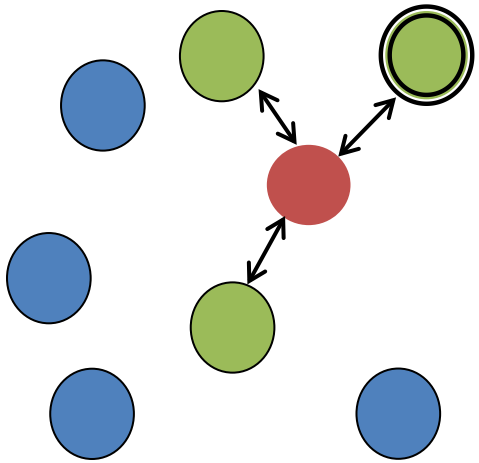


Inquiry Strategies

- Gossip inquiry
  - Every receiver responds
  - Every receiver forwards with a given probability

$$f_{probabilistic}(\Theta) = (\Theta < p)$$

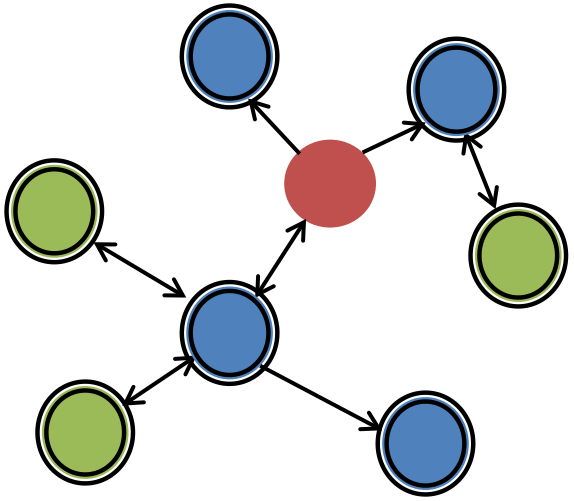
$$I = (f_{probabilistic}(rand), true)$$



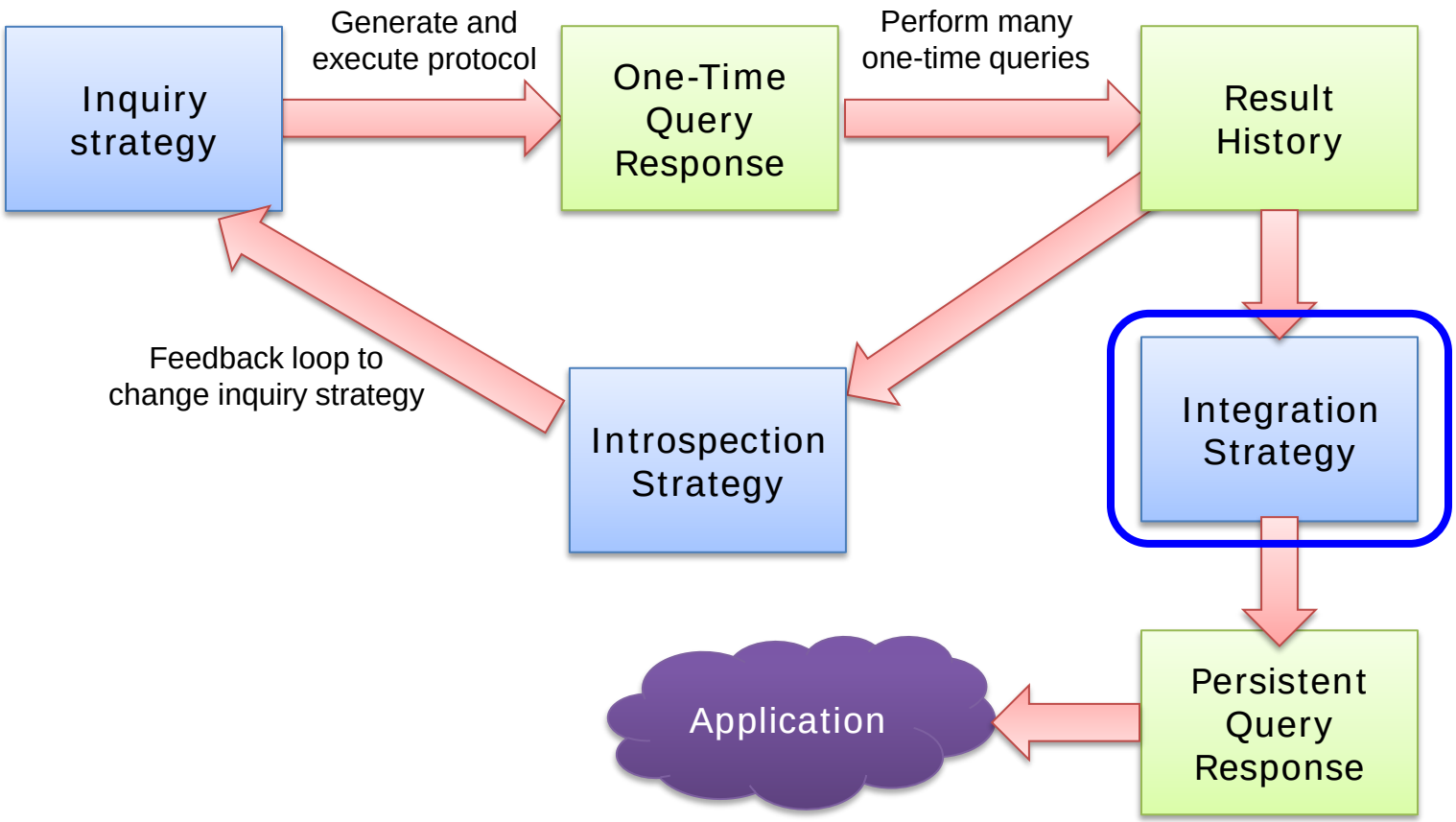
- Random sampling inquiry
  - Every receiver responds with a given probability
  - Every receiver forwards

$$r_{random}(\Theta) = (\Theta < k)$$

$$I = (true, r_{random}(rand))$$



PAQ Middleware Overview



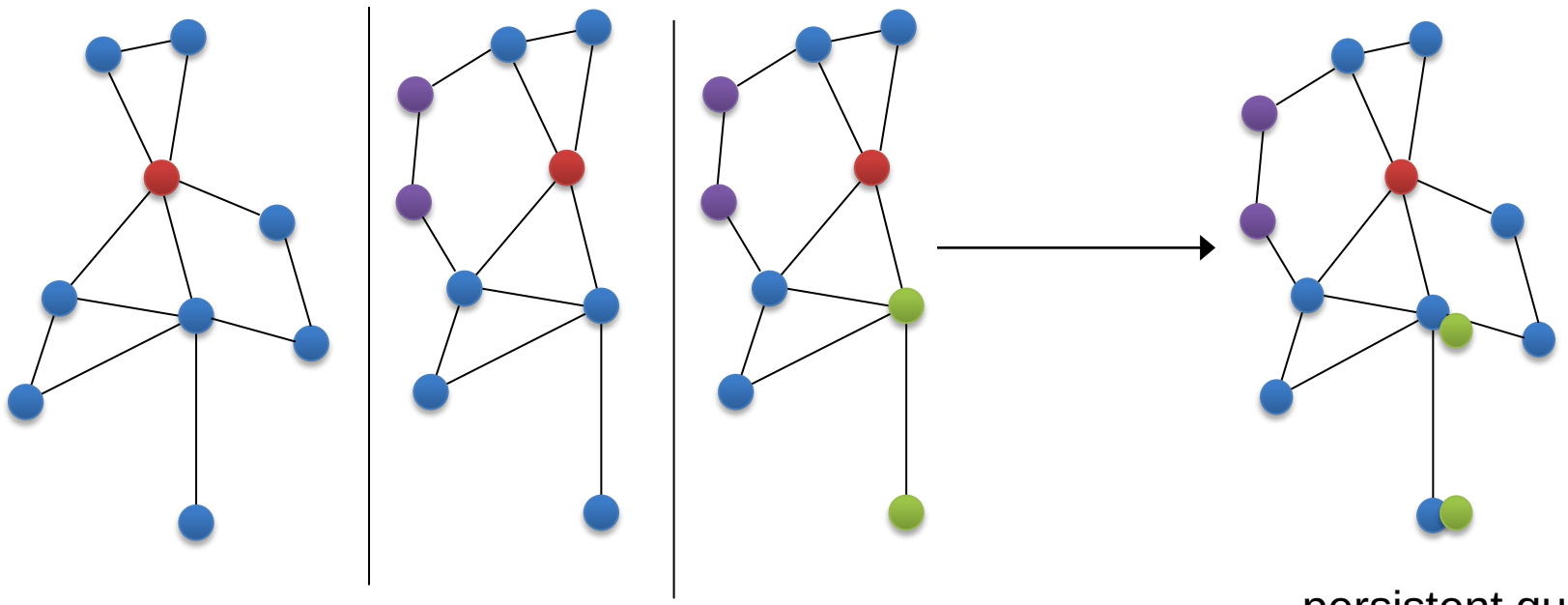
Integration Strategy

- Persistent query implemented as a composition of a history of one-time snapshot results
- Composed answer should resemble a continuous stream
  - Even when underlying network is dynamic
- Integration strategy: user-defined mechanism of composition
  - Express diverse composition mechanisms through set operators

Cumulative Integration

Result contains every data value ever present

$$\Pi_i = \Pi_{i-1} \cup \rho_i$$



one-shot query results (ρ_i)

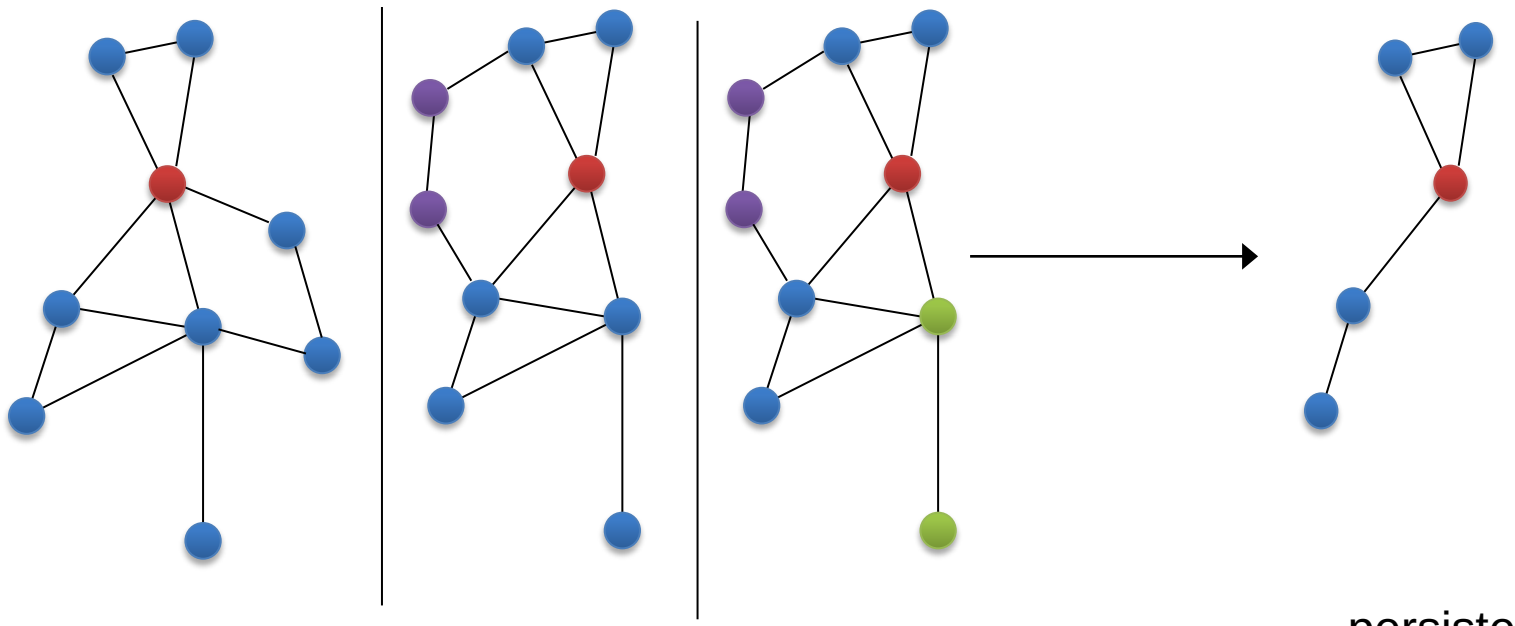
persistent query result (π_i)

Added Hosts Data Value Change

Stable Integration

Result contains data values always present

$$\Pi_i = \Pi_{i-1} \bigcap \rho_i$$



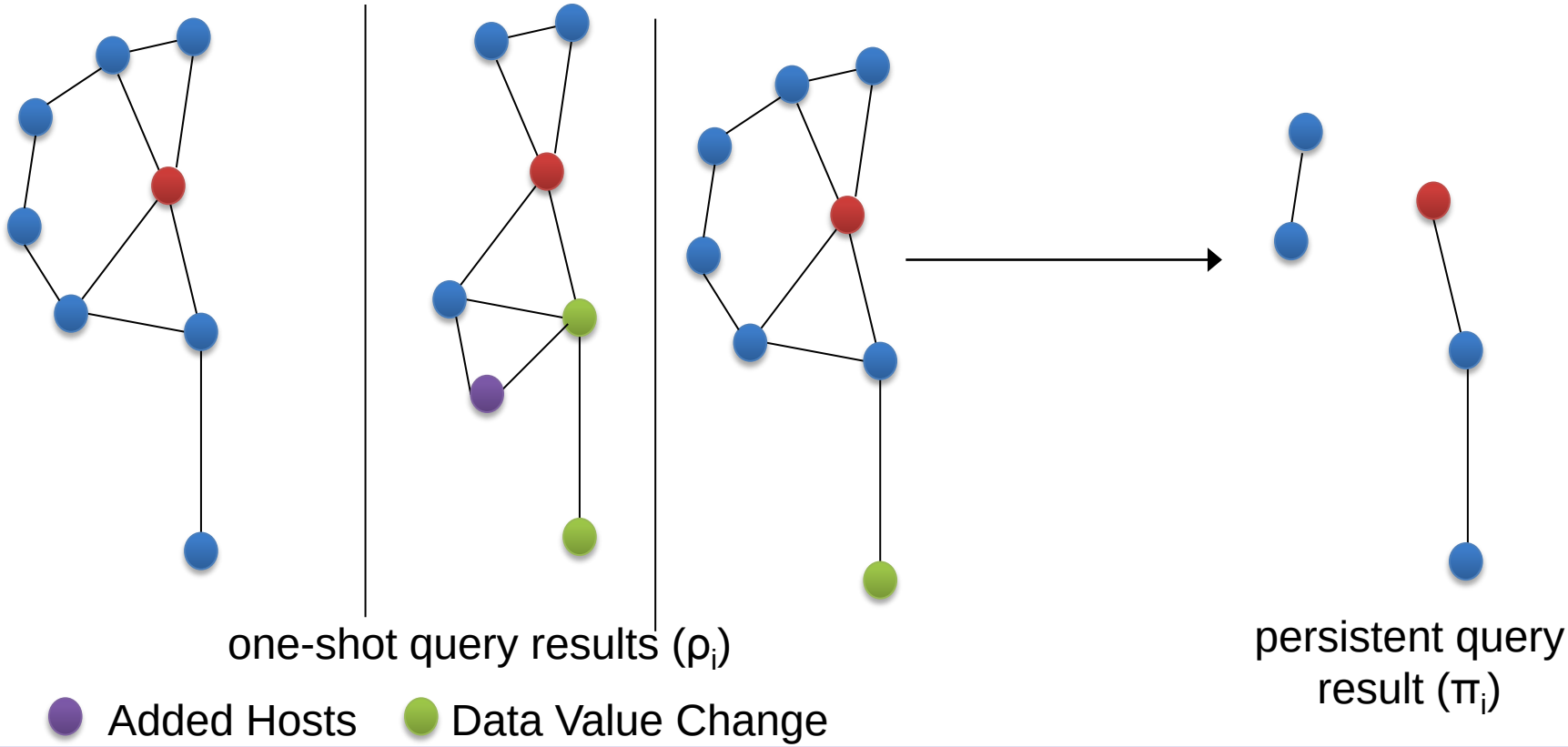
one-shot query results (ρ_i)

persistent query result (π_i)

Added Hosts Data Value Change

Transient Departure Integration

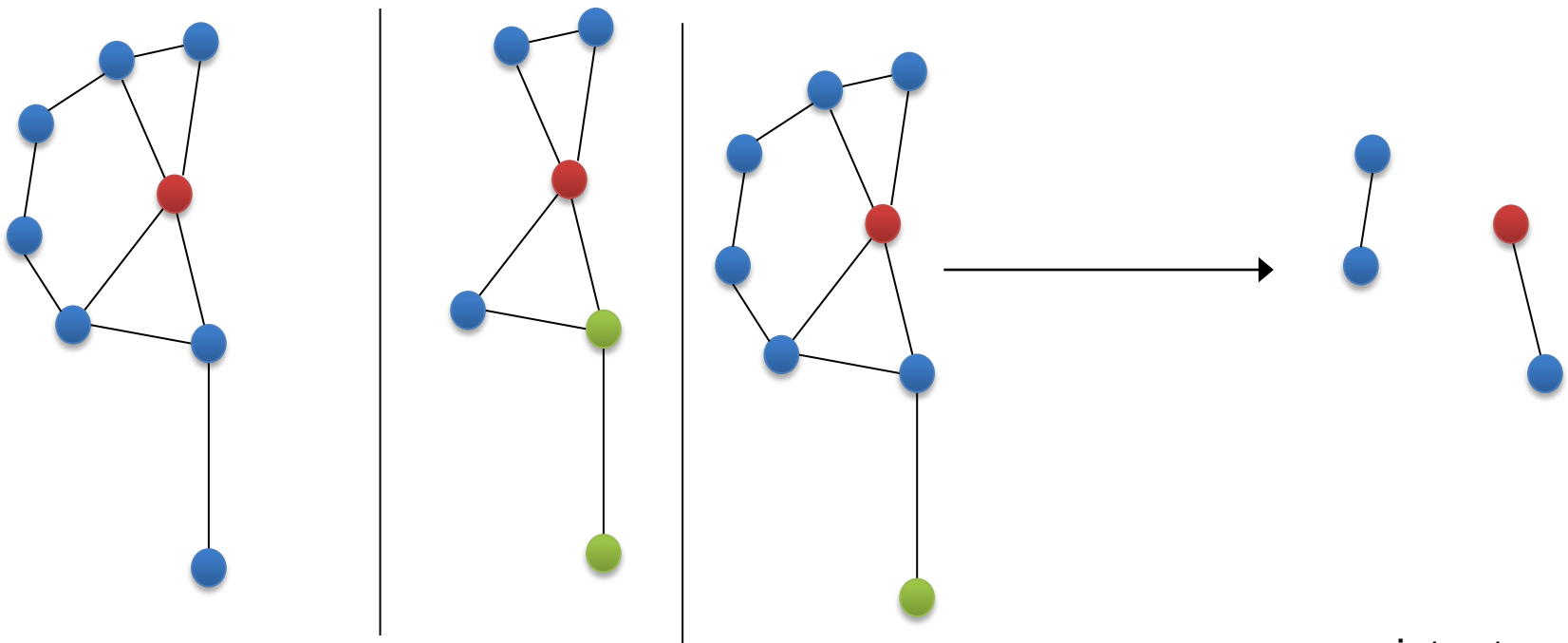
Result contains any data values departed, even if they subsequently reappeared



Returns Integration

Result contains data values that departed but returned

$$\Pi_i = \Pi_{i-1} \cup \left\langle set\, p : p \in \rho_i \wedge p \in \Pi_{i-1}^{t-departs} :: p \right\rangle$$

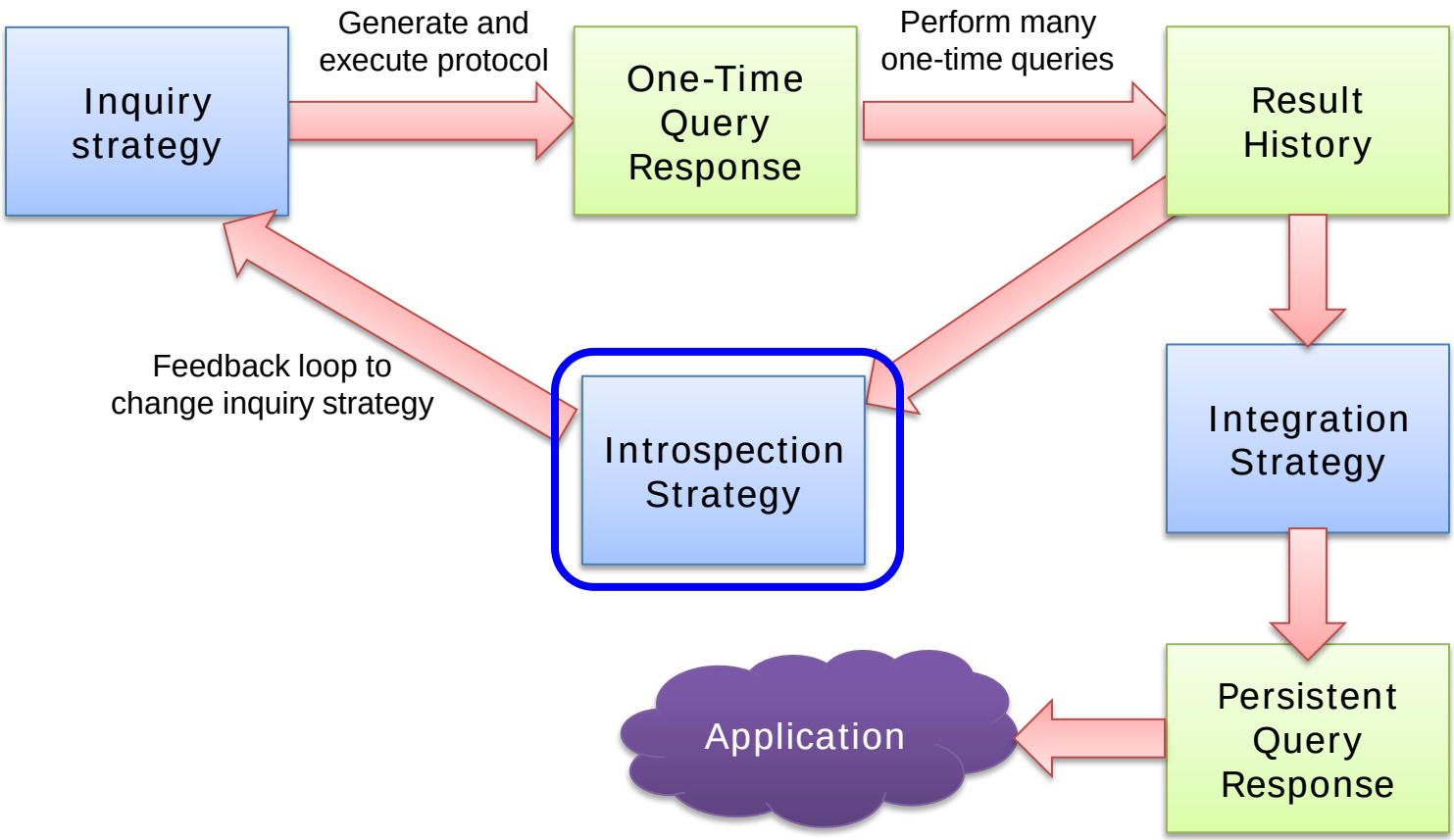


one-shot query results (ρ_i)

persistent query result (π_i)

Added Hosts Data Value Change

PAQ Middleware Overview



Introspection

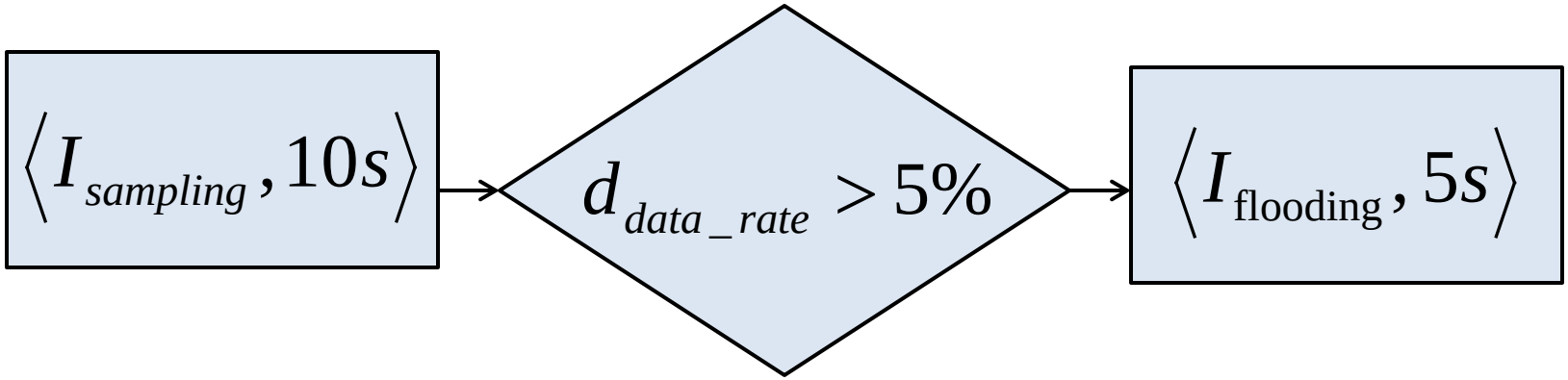
- Introspection: the ability to identify interesting state
- As part of introspection, we:
  - Determine the suitability of an Inquiry Strategy
  - Systematically compare the costs and benefits of Inquiry Strategies
- Generate a metric *d* that can be compared against an application-specified threshold
- Integration Strategies can be converted to Introspection Strategies, e.g.,
  - the fraction of values in the one-shot query that are "new" since the previous query

Introspection Strategy Examples

- Spatial Coverage Introspection
  - E.g, when the results are not reflective of the entire site, change from the gossip inquiry strategy
- Semantic Discovery
  - E.g., when a particular value is discovered, change the sampling strategy
- Data Change Rate Introspection
  - E.g., if the data is changing too rapidly (i.e., faster than the sampling strategy can sample), change the sampling strategy

Adaptation Strategy

- Introspection identifies need for new policy
- Adaptation Strategy allows applications to easily specify the alternate course of action
- User provides a threshold and a new policy



Application Examples

- PAQ simplifies application development
  - By hiding the complexities of persistent querying a dynamic environment
- PAQ enables expressive applications
  - By providing varying strategies for inquiry, integration, and introspection
  - And enabling extensibility
- To demonstrate, we employed PAQ to write two adaptive applications
  - Industrial leak monitoring
  - Traffic congestion

Hazard Monitoring Application

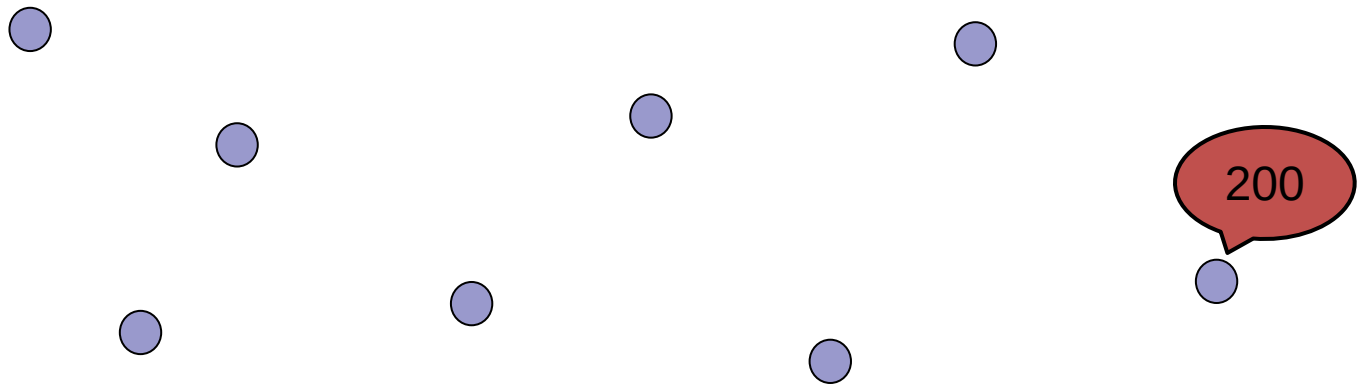
Phase :
Spot Check

Inquiry:
Random Sampling
(p =0.5, low rate)

Integration:
Cumulative

Introspection:
Semantic
(v = 200, δ = 1)

```
private void startQuery(){
  myInquiry = new Inquiry(new RandomSampling(0.5), 5000);
  myIntegration = new WindowedCumulativeIntgration(4);
  myIntrospection =
    new SemanticDiscoveryIntrospection(new Integer(thresh));
  PersistentQuery initialQuery =
    new PersistentQuery(myInquiry, myIntegration,
                        myIntrospection, initialAdaptation);
}
```



Application Phase II

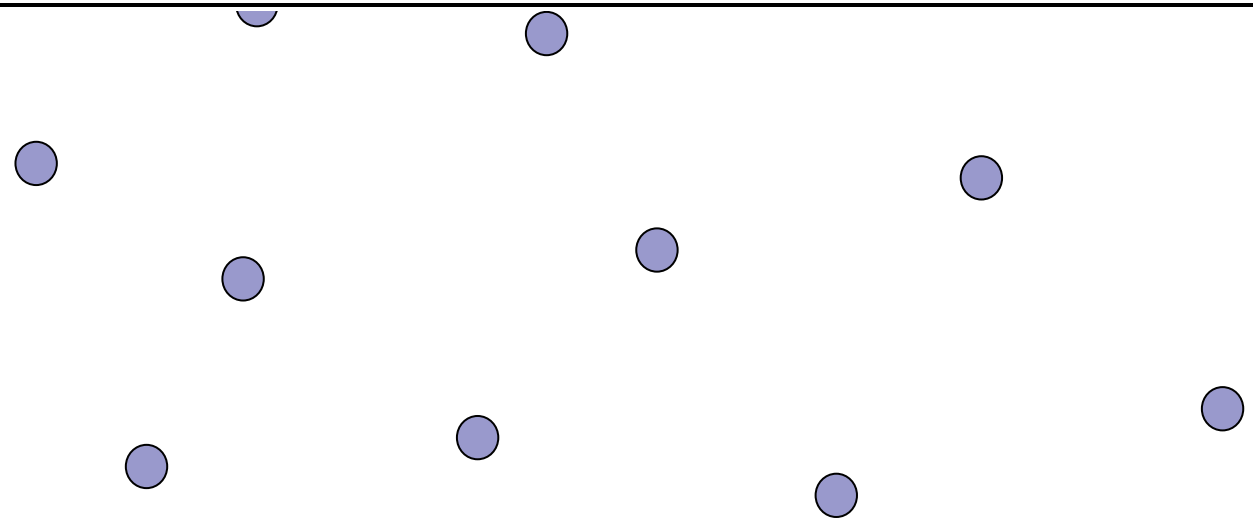
Phase : Leak Check

Inquiry: Flood (high rate)

Integration: Additive

Introspection: Additive Data Rate Change (δ = 3)

```
private void adaptQuery(){
  ...
  myInquiry = new Inquiry(new Flooding(), 500);
  myIntegration = new CumulativeIntegration();
  myIntrospection =
    new SemanticAdditiveIntrospection(new Integer(thresh));
  PersistentQuery initialQuery =
    new PersistentQuery(myInquiry, myIntegration,
                        myIntrospection, leakAdaptation);
}
```



Application Phase III

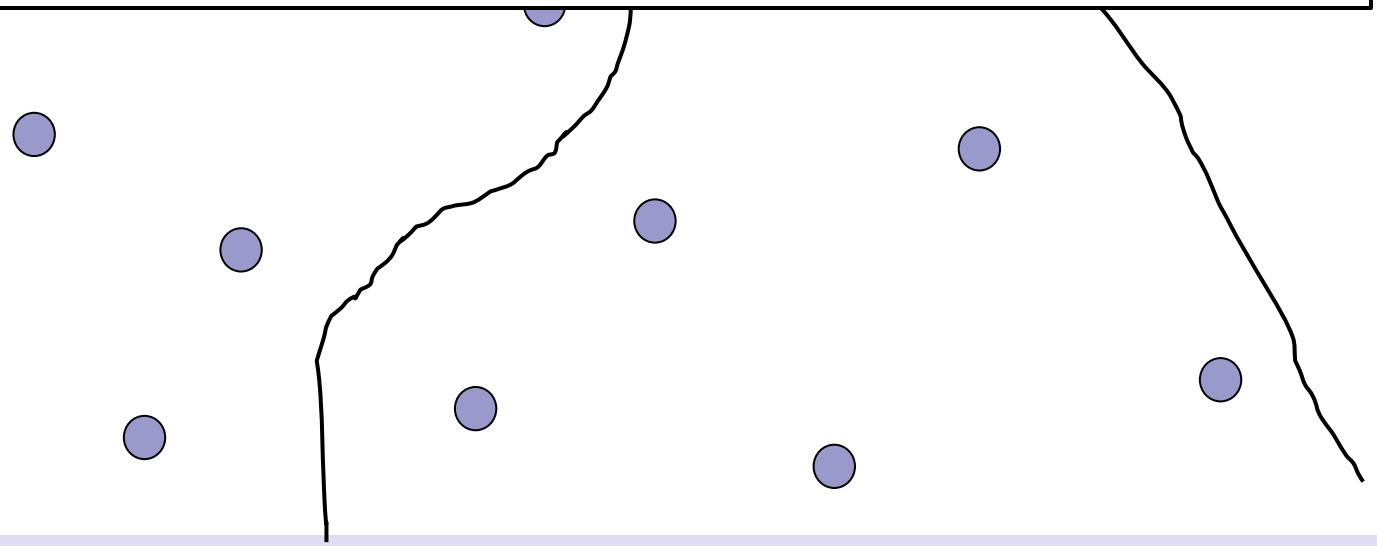
Phase: Targeted Collection

Inquiry: Location-Base (high rate)

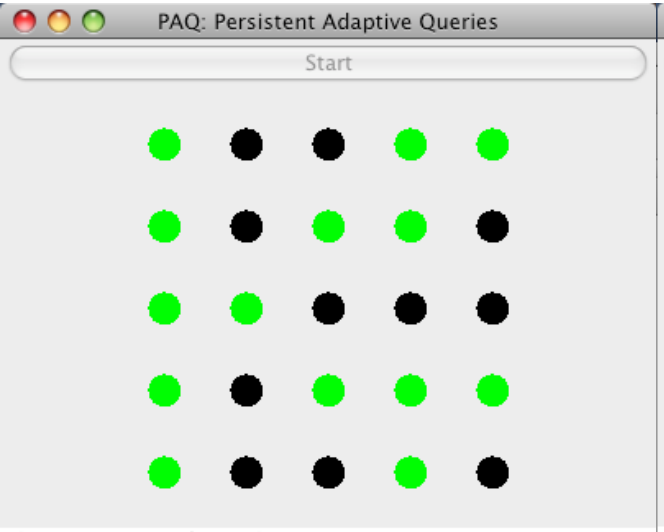
Integration: Cumulative

Introspection: Spatial Coverage Change (δ = 2.5)

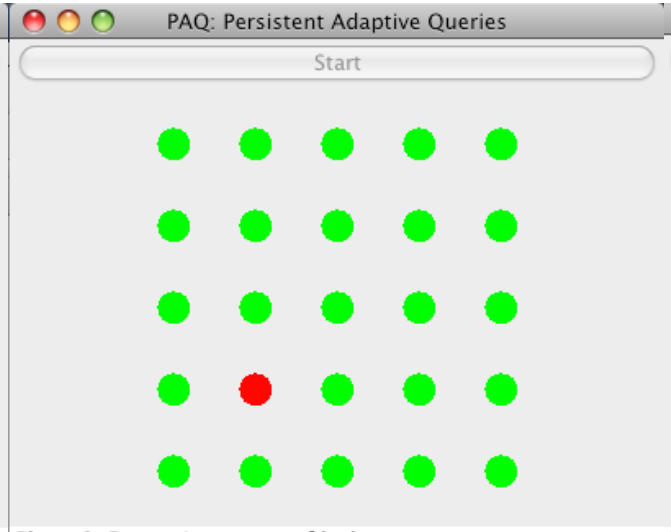
```
private void adaptQuery(){
  ...
  myInquiry = new Inquiry(new LocBased(mid.x, mid.y, radius), 500);
  myIntegration = new WindowedCumulativeIntegration(6);
  myIntrospection =
    new SpatialCoverageIntrospection(mid.x, mid.y, radius);
  PersistentQuery initialQuery =
    new PersistentQuery(myInquiry, myIntegration,
                        myIntrospection, locationAdaptation);
}
```



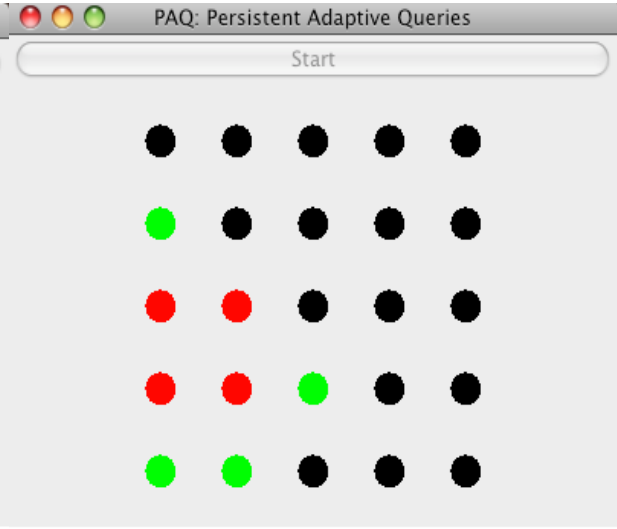
Integrated Response



Phase 1: Monitor for Leak
Inquiry: Random sampling, probability=0.5, freq.=5s
Integration: Windowed cumulative, window=20s
Introspection: Semantic discovery, detecting leak



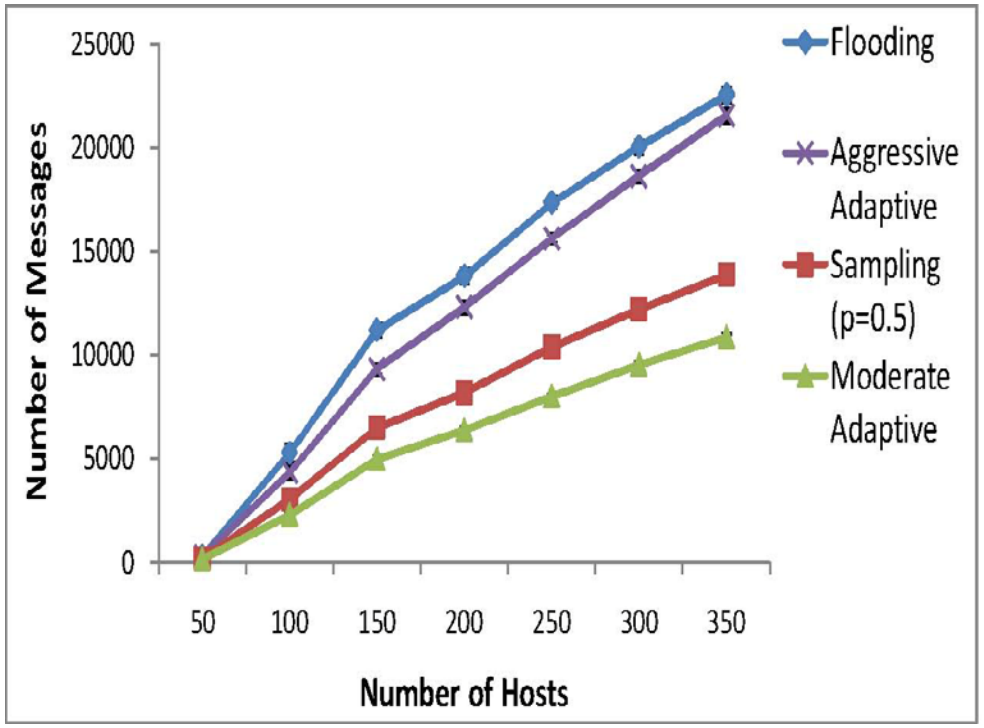
Phase 2: Determine extent of leak
Inquiry: Flooding, freq.=1s
Integration: Windowed cumulative, window=20s
Introspection: Semantic additive change rate, finding additiona



Phase 3: Localizing around the leak
Inquiry: Location Based, area around detected leak
Integration: Windowed cumulative, window=10s
Introspection: Semantic discovery, detecting absence of leak

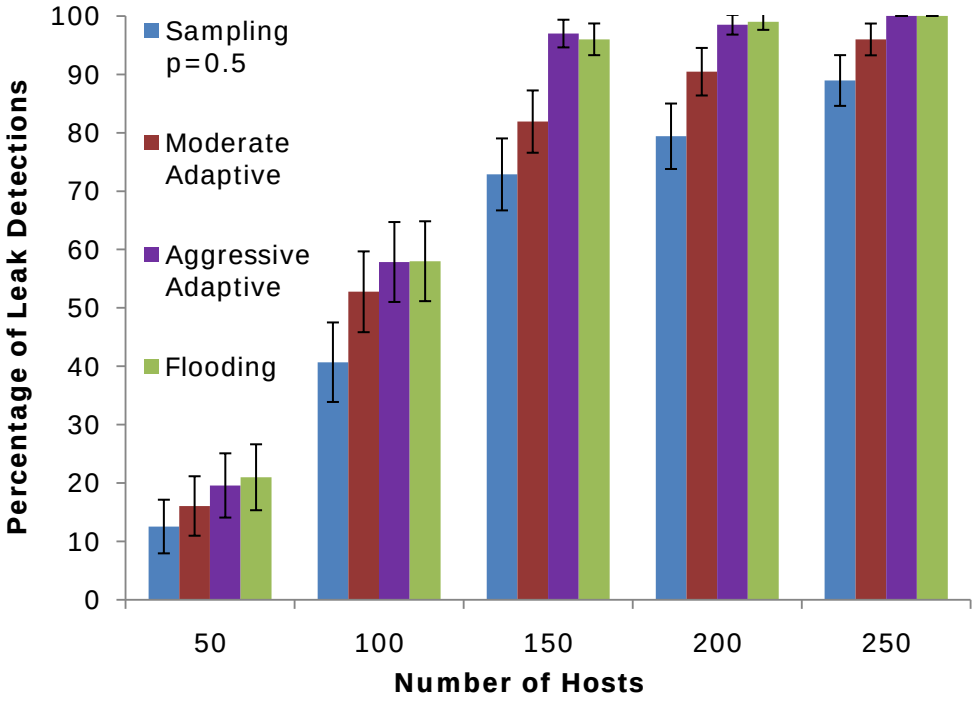
Performance Evaluation

- OMNeT++ for robust results
- Compared adaptive and non-adaptive approaches
- Two different policies of adaptation
- Adaptive approach: No worse than flooding even when more queries are issued



Evaluation II

- Adaptive approaches flag more leaks than just sampling
- Aggressive version does as well as flooding



Summary

- Sense and respond applications are becoming increasingly important
- PAQ allows application developers to easily construct and deploy adaptive applications
- Presented the main abstractions
  - Inquiry, Introspection, Integration and Adaptation
- Performed both qualitative and quantitative evaluation to evaluate effectiveness of PAQ

Mobile & Pervasive Computing Group

EXCELLENCE IN DISTRIBUTED GLOBAL ENVIRONMENTS

Thanks! Questions?

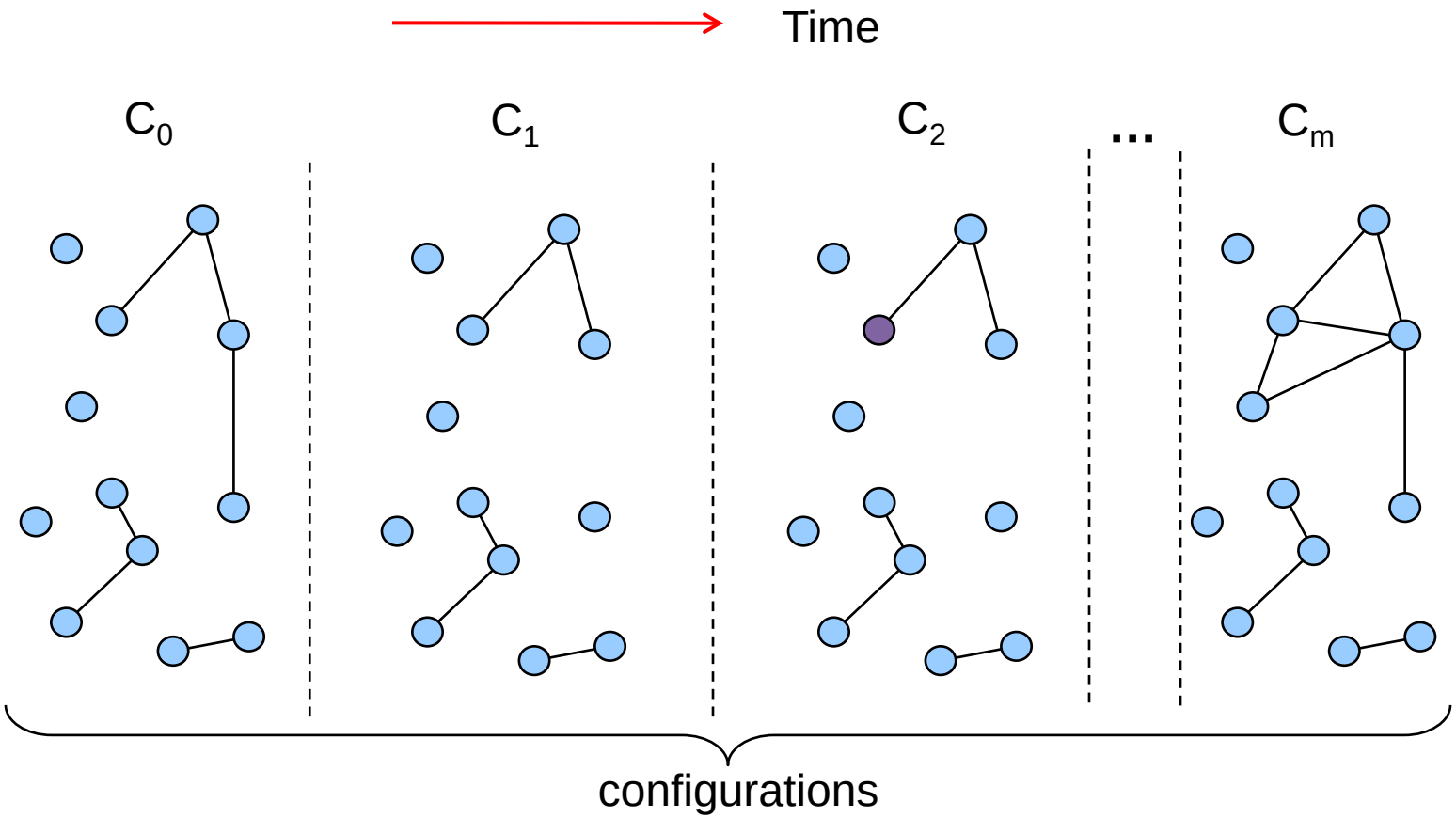
Vasanth Rajamani

The University of Texas at Austin

vasanthrajamani@mail.utexas.edu

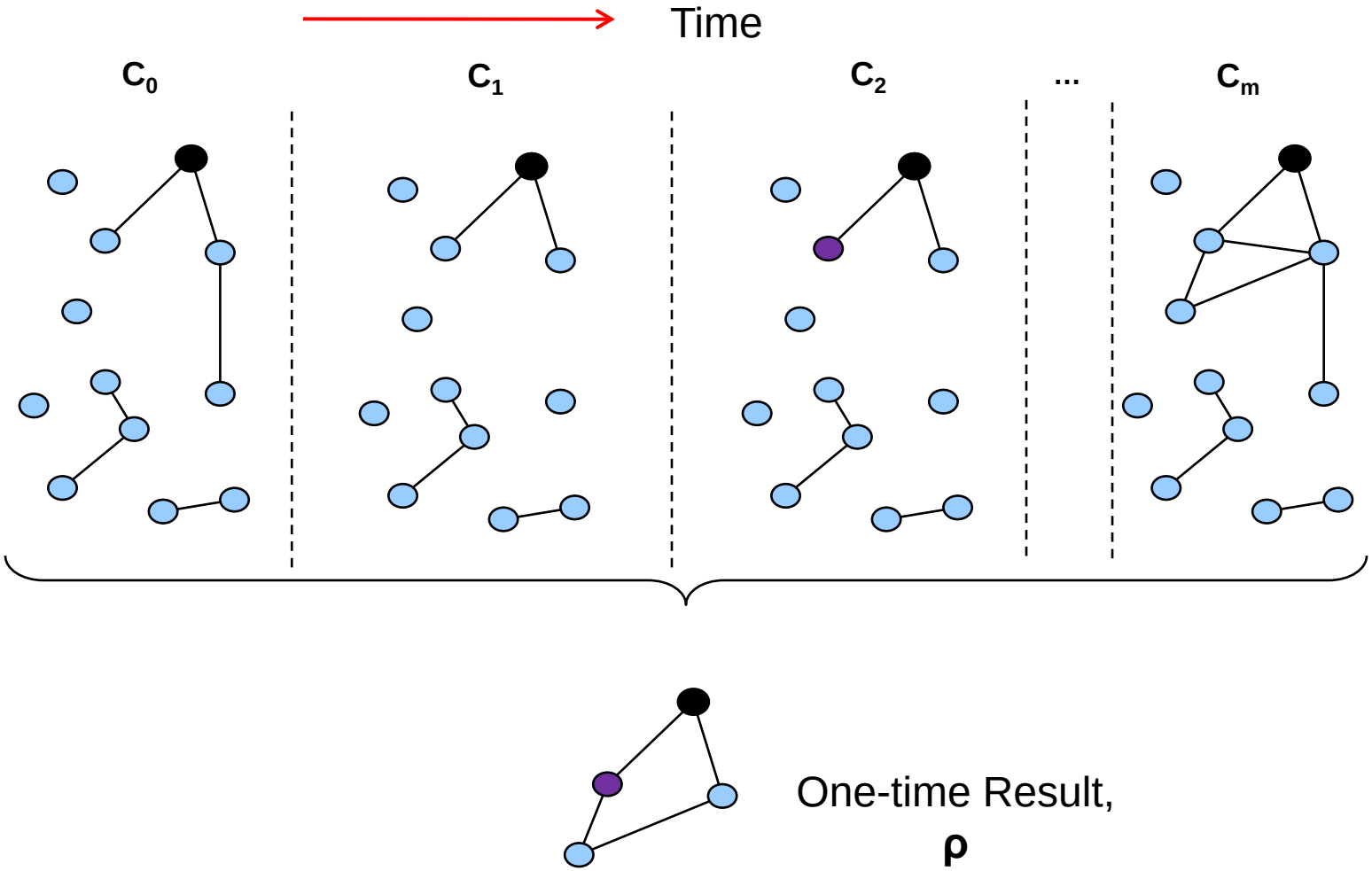
Backup

Formal Model of Dynamic Network



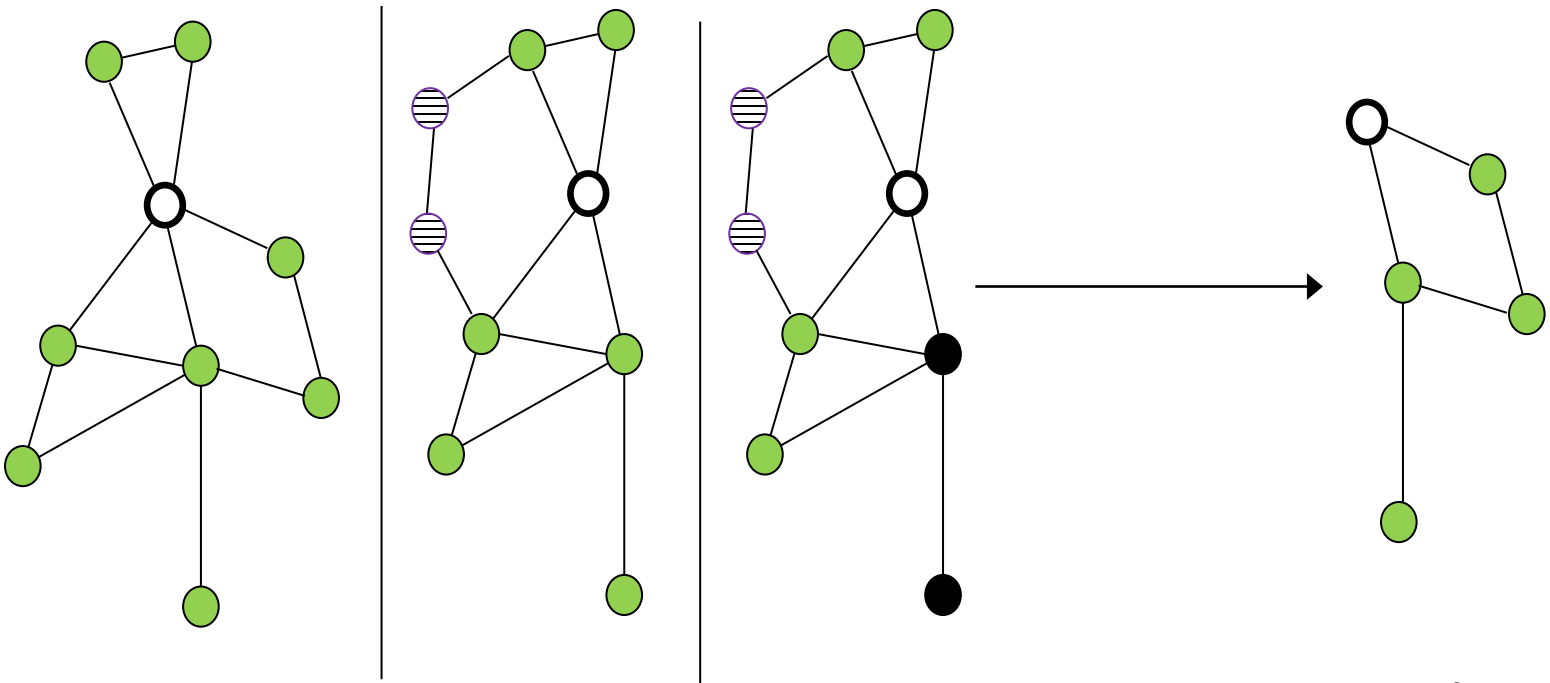
$$\Delta C = value\,change \oplus neighbor\,change \oplus message\,exchange$$

One-Time Queries



Departure Integration

$$\Pi_i = \rho_o - \rho_i$$



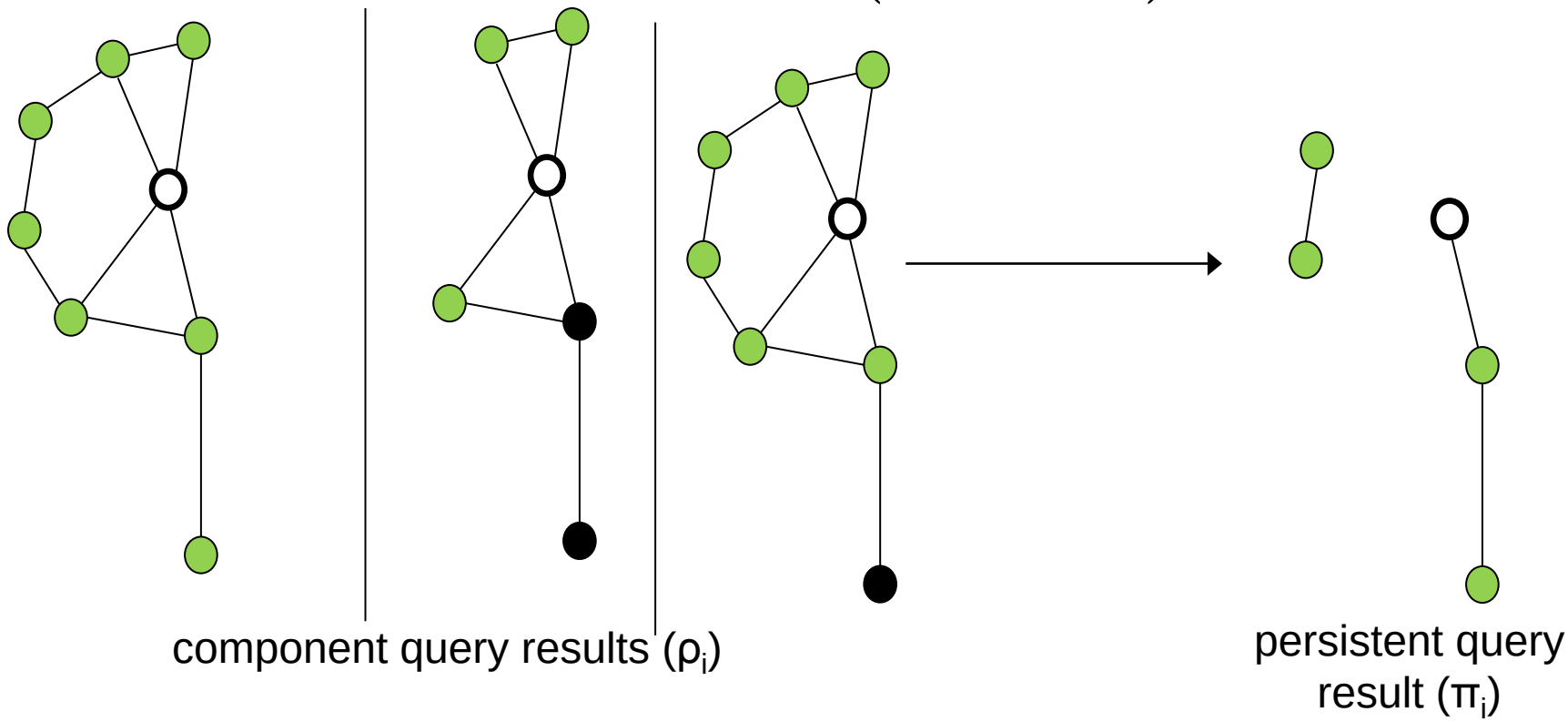
component query results (ρ_i)

persistent query result (π_i)

Added Hosts Data Value Change

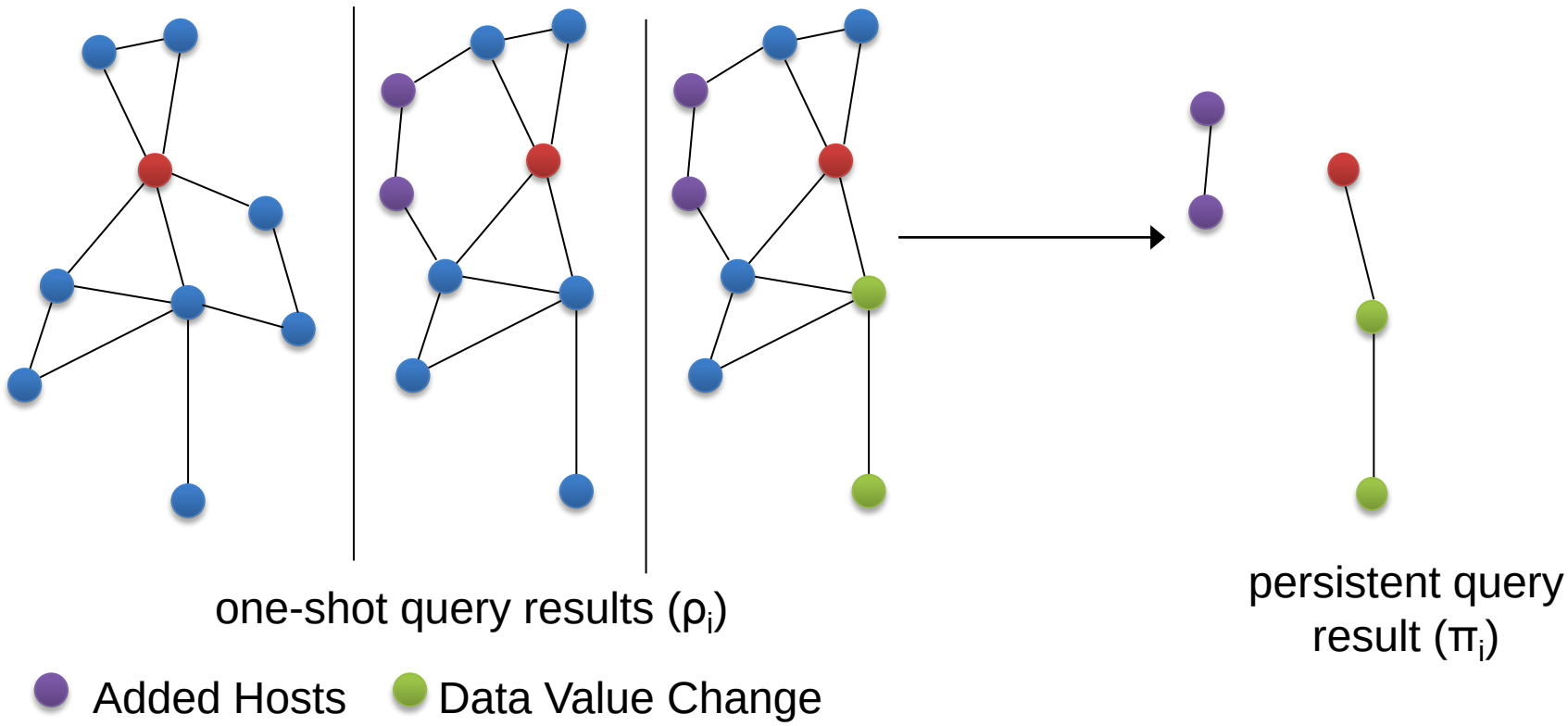
Transient Departure Integration

$$\Pi_i = \Pi_{i-1} \cup \left(\rho_o \cap \left(\rho_{i-1} - \rho_i \right) \right)$$



Additive Integration

Result contains any data value that is available now but was not available initially



Transient Additive Integration

Result contains any data values added, even if they subsequently disappeared