Messages with Implicit Destinations as Mobile Agents

Ahmad Ahmad-Kassem

INRIA - Université de Lyon

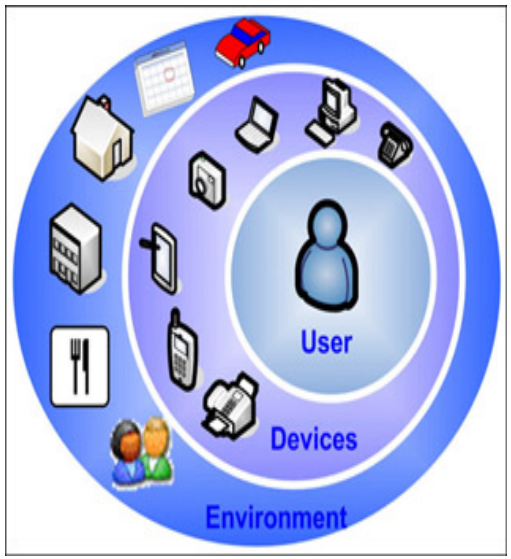
Joint work with:

Stéphane Grumbach [INRIA], Stéphane Ubéda [INRIA]

AGERE'12, Tucson, Arisona, USA

Context

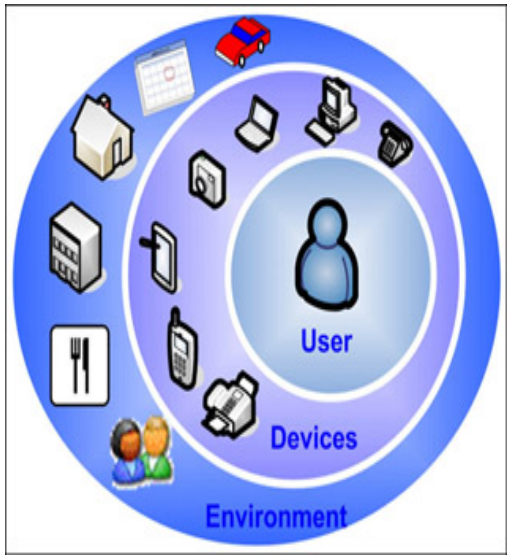
- Ubiquitous environment
  - heterogeneous devices
  - data intensive applications
- Complex network infrastructure
  - dynamic of some networks
  - various types of failures



Requirements

Such applications require distributed algorithms

- are difficult to program
- require skilled programmers
- offer limited warrantee on their behavior



Problematic

Traditionally

- messages with **explicit** destination

Topology

- include failures
- join/leave the network at any time

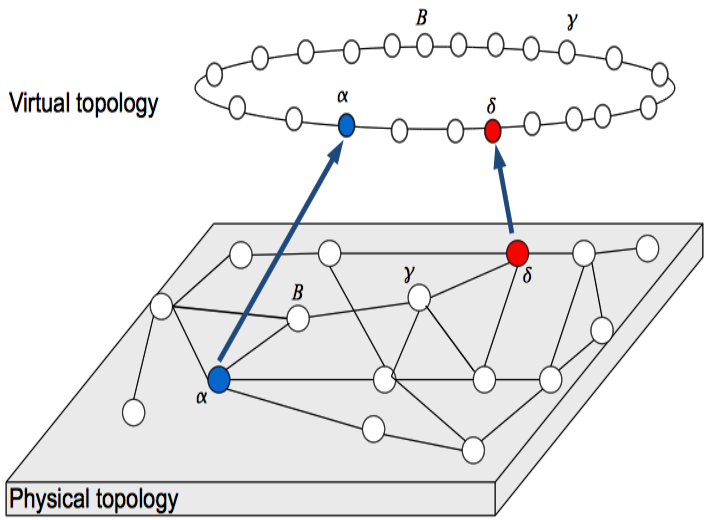
Problematic

Traditionally

- messages with **explicit** destination

Topology

- include failures
- join/leave the network at any time



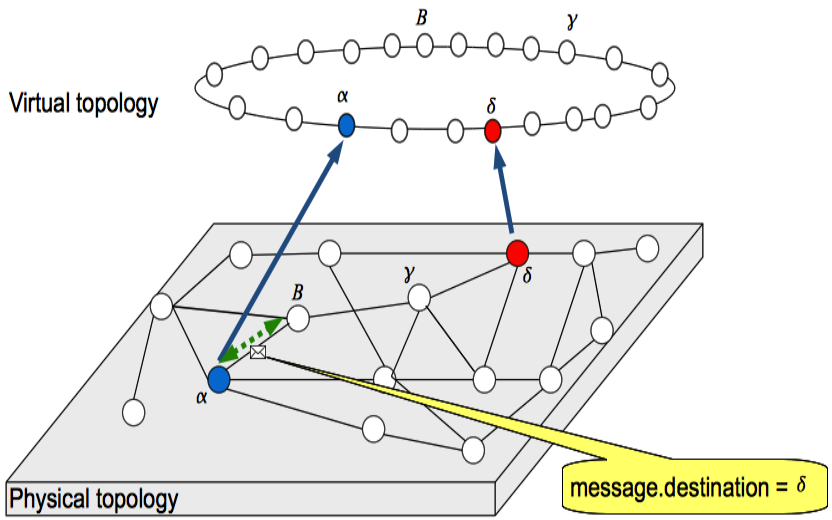
Problematic

Traditionally

- messages with **explicit** destination

Topology

- include failures
- join/leave the network at any time



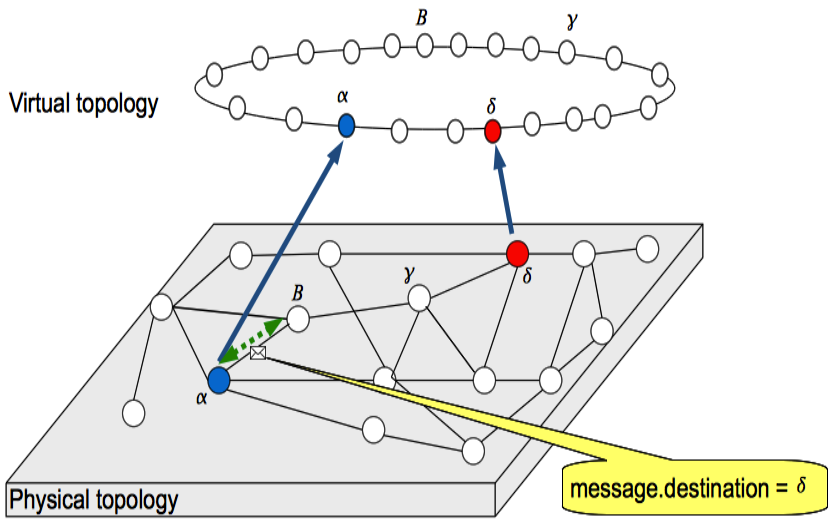
Problematic

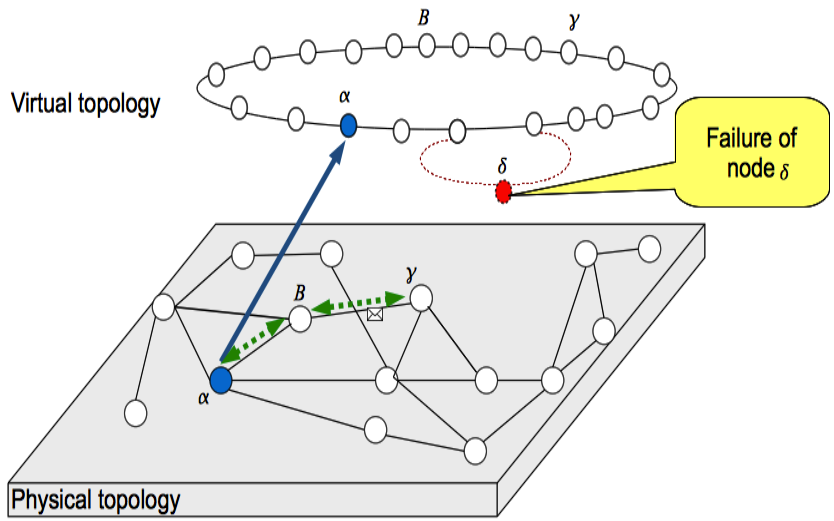
Traditionally

- messages with **explicit** destination

Topology

- include failures
- join/leave the network at any time





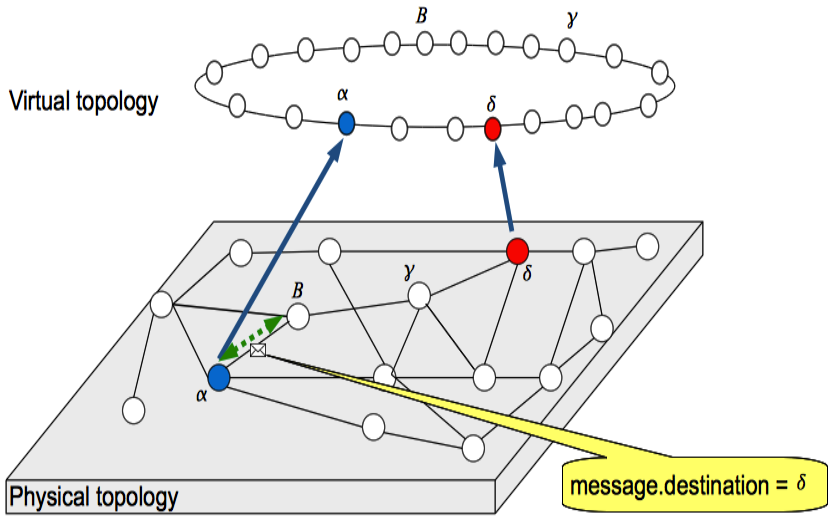
Problematic

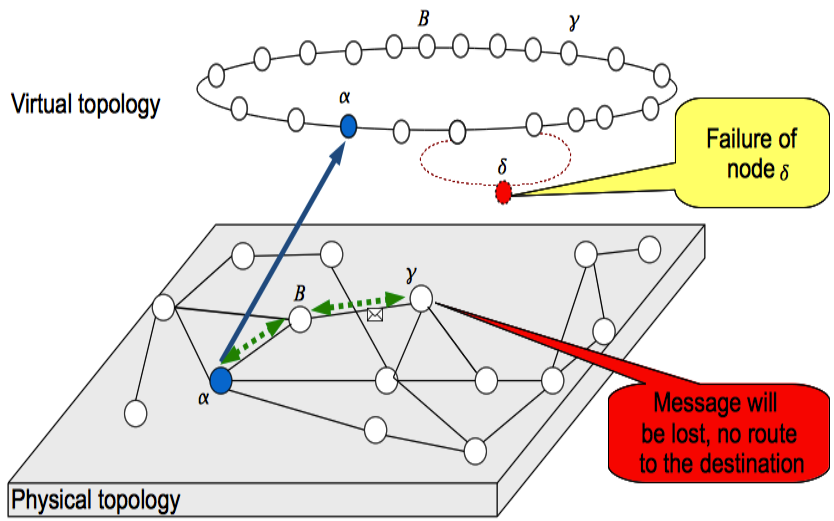
Traditionally

- messages with **explicit** destination

Topology

- include failures
- join/leave the network at any time





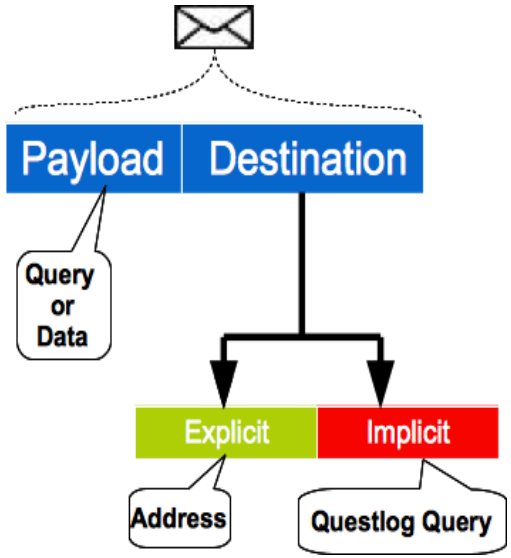
Proposition

- A framework that allows:

  - messages as mobile agents with **explicit/implicit** destinations

- A rule-based language **Questlog**

  - implicit destination is specified by a **Questlog** query



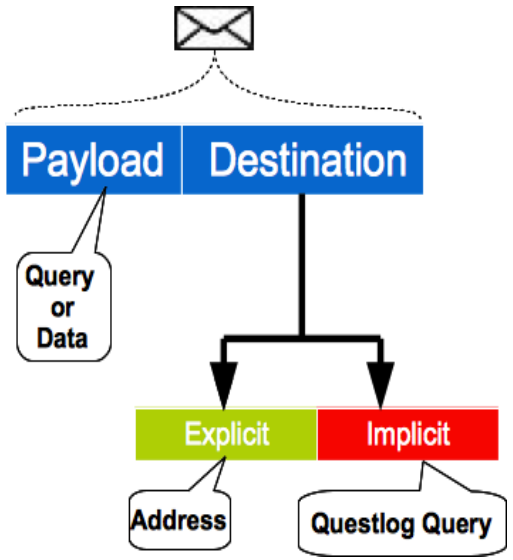
Proposition

- A framework that allows:

  - messages as mobile agents with **explicit/implicit** destinations

- A rule-based language **Questlog**

  - implicit destination is specified by a **Questlog** query



→ Messages are solved while traveling in the network

Outline

Outline

Questlog query

- Query in Questlog
  - Based on relational data model
  - Predicate prepended by: ?
  - $?R(x_1, ..., x_\ell)$
    - R is a relation symbol
    - $x_1, \cdots, x_\ell$ are variables or constants

On-demand routing protocol

Find the nexthop "*y*" and the number of hop "*n*" for a route from source "*s*" to destination "*d*"

On-demand routing protocol

Find the nexthop "y" and the number of hop "n" for a route from source "s" to destination "d"

$$?Route(s, d, y, n)$$

On-demand routing protocol

Find the nexthop "y" and the number of hop "n" for a route from source "s" to destination "d"

$$?Route(s, d, y, n)$$

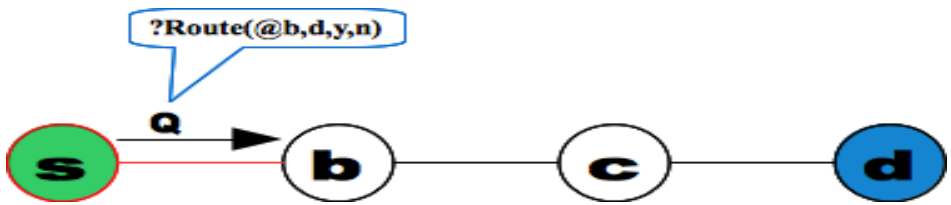
$$\updownarrow Route(x, w, w, 1) : -Link(x, w).$$

$$\updownarrow Route(x, w, z, n+1) : -\neg Link(x, w), Link(x, z), ?Route(@z, w, y, n).$$

On-demand routing protocol

Find the nexthop "y" and the number of hop "n" for a route from source "s" to destination "d"

$$?Route(s, d, y, n)$$



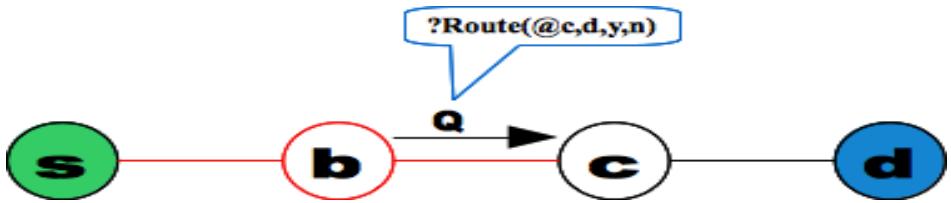
$$\updownarrow Route(x, w, w, 1) : -Link(x, w).$$

$$\updownarrow Route(x, w, z, n+1) : -\neg Link(x, w), Link(x, z), ?\mathbf{Route}(@\mathbf{z}, \mathbf{w}, \mathbf{y}, \mathbf{n}).$$

On-demand routing protocol

Find the nexthop "*y*" and the number of hop "*n*" for a route from source "*s*" to destination "*d*"

$$?Route(s, d, y, n)$$



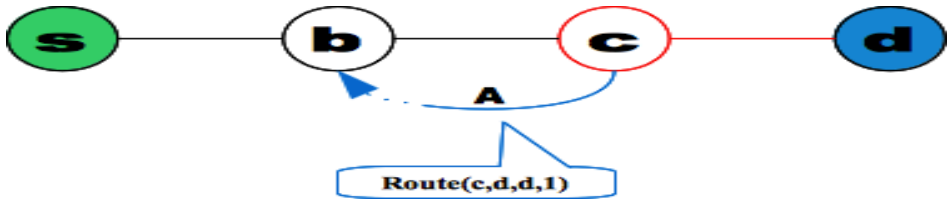
$$\updownarrow Route(x, w, w, 1) : -Link(x, w).$$

$$\updownarrow Route(x, w, z, n+1) : -\neg Link(x, w), Link(x, z), ?\mathbf{Route}(@\mathbf{z}, \mathbf{w}, \mathbf{y}, \mathbf{n}).$$

On-demand routing protocol

Find the nexthop "y" and the number of hop "n" for a route from source "s" to destination "d"

$$?Route(s, d, y, n)$$



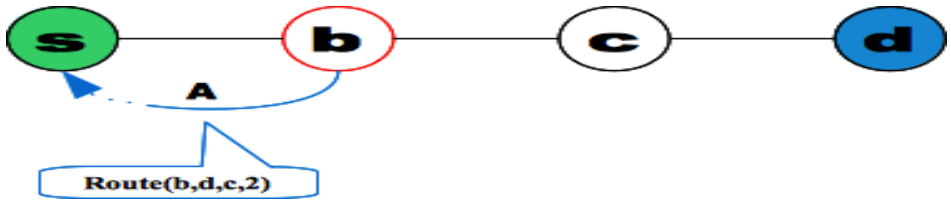
$$\updownarrow \mathbf{Route}(\mathbf{x}, \mathbf{w}, \mathbf{w}, \mathbf{1}) : -Link(x, w).$$

$$\updownarrow Route(x, w, z, n+1) : -\neg Link(x, w), Link(x, z), ?Route(@z, w, y, n).$$

On-demand routing protocol

Find the nexthop "y" and the number of hop "n" for a route from source "s" to destination "d"

$$?Route(s, d, y, n)$$



$$\updownarrow Route(x, w, w, 1) : -Link(x, w).$$

$$\updownarrow \mathbf{Route}(\mathbf{x}, \mathbf{w}, \mathbf{z}, \mathbf{n} + \mathbf{1}) : -\neg Link(x, w),$$
$$Link(x, z), ?Route(@z, w, y, n).$$

WSN application

Select the positions of nodes which have, together with their neighbors to avoid individual measurement errors, a temperature higher than a threshold T

WSN application

Select the positions of nodes which have, together with their neighbors to avoid individual measurement errors, a temperature higher than a threshold T

$$?WarnPos(v, x, y)$$

WSN application

Select the positions of nodes which have, together with their neighbors to avoid individual measurement errors, a temperature higher than a threshold T

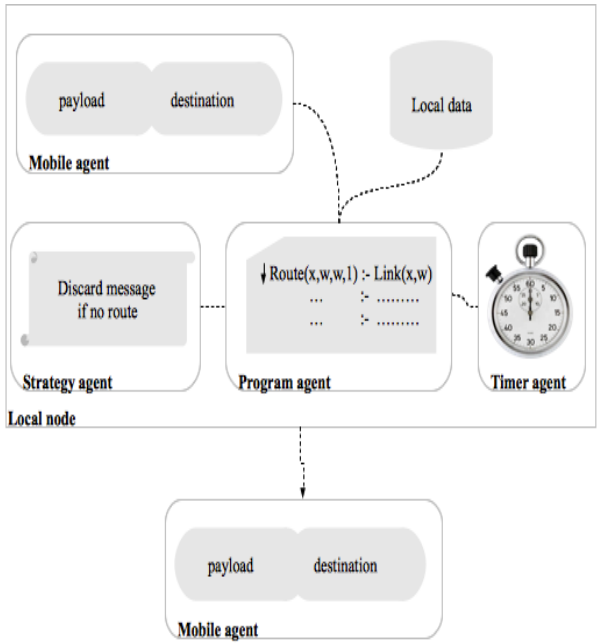
$$?WarnPos(v, x, y)$$

$$\begin{aligned} \uparrow WarnPos(v, x, y) :- & \; Pos(v, x, y), Tmp(v, t), t > T, \\ & \forall w \; Link(v, w), ?HighTmp(@w). \\ \uparrow HighTmp(v) :- & \; Tmp(v, t), t > T. \end{aligned}$$

Outline

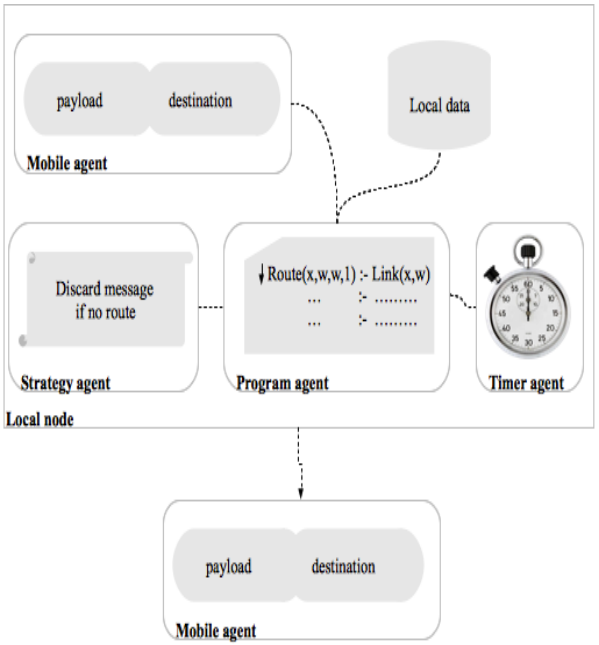
Questlog programs

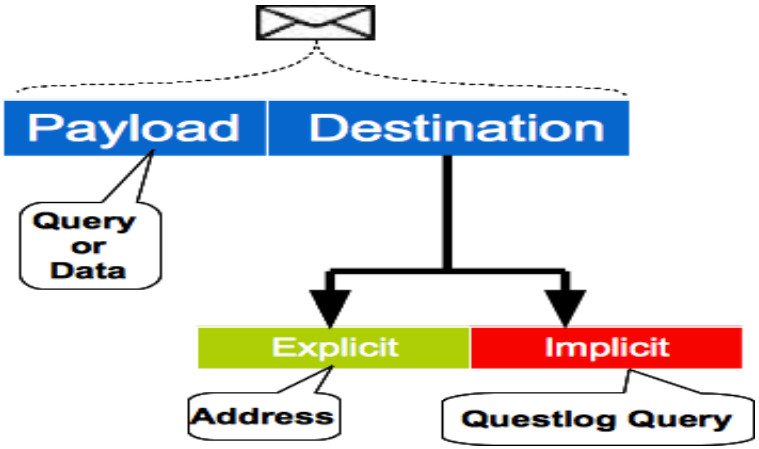
- The Questlog programs are agents
  - installed on each node
  - run concurrently
  - interact to solve tasks
- The computation is distributed
  - nodes exchange asynchronously messages of the form:



Questlog programs

- The Questlog programs are agents
  - installed on each node
  - run concurrently
  - interact to solve tasks
- The computation is distributed
  - nodes exchange asynchronously messages of the form:

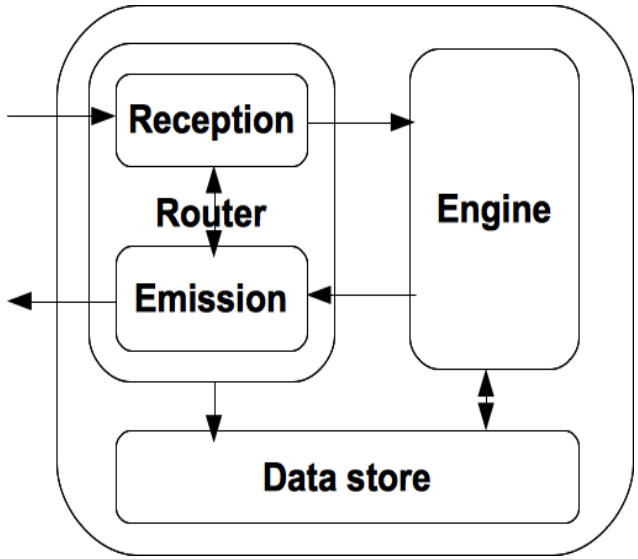




Execution

Each node is equipped with an embedded machine:

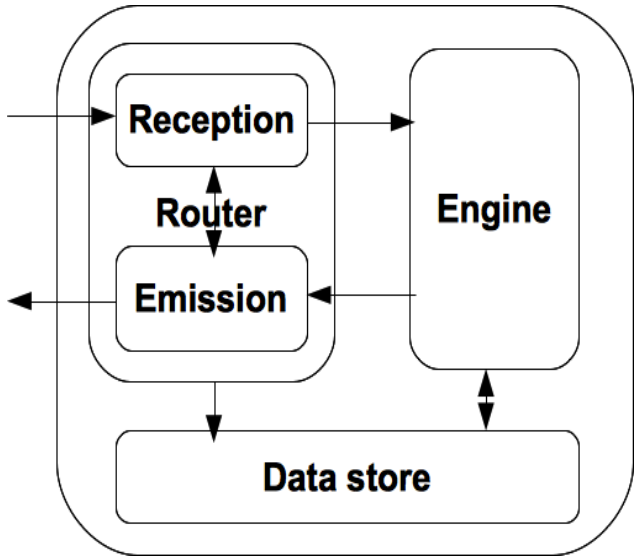
- **router**: to handle the communication with the network
- **engine**: to evaluate the Questlog queries
- **local datastore**: to manage the information (data and programs) local to the node



Execution

Each node is equipped with an embedded machine:

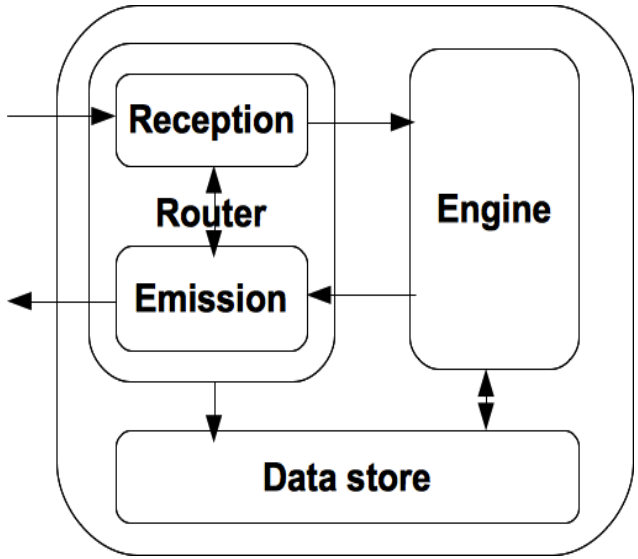
- **router**: to handle the communication with the network
- **engine**: to evaluate the Questlog queries
- **local datastore**: to manage the information (data and programs) local to the node



Execution

Each node is equipped with an embedded machine:

- **router**: to handle the communication with the network
- **engine**: to evaluate the Questlog queries
- **local datastore**: to manage the information (data and programs) local to the node



Outline

QuestMonitor simulation tool

QuestMonitor:

- create groups of nodes and install protocols
- allow to interact with a network at run time
- visualize the behavior of protocols

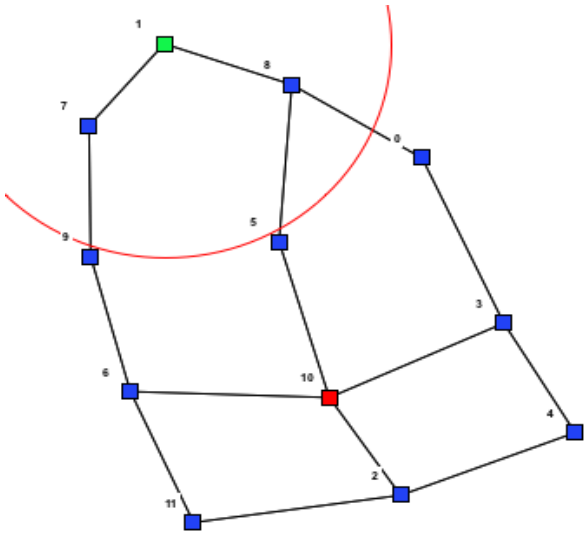
QuestMonitor simulation tool

QuestMonitor:

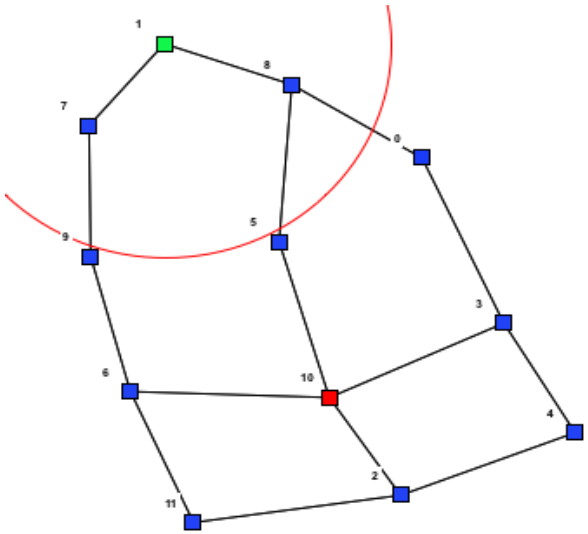
- create groups of nodes and install protocols
- allow to interact with a network at run time
- visualize the behavior of protocols

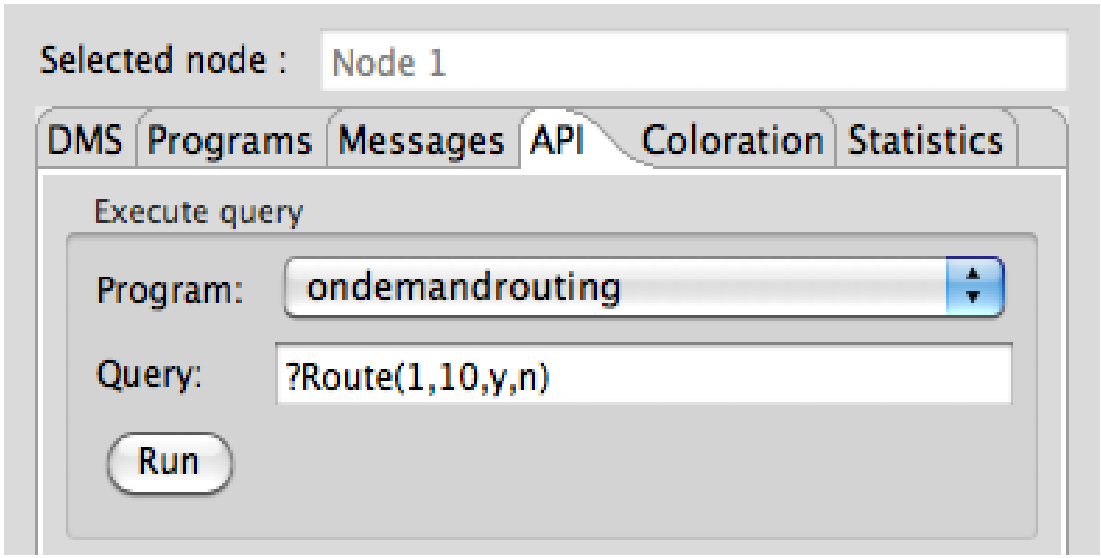
→ Extend it with an API to send Questlog queries at run time

Simulation results

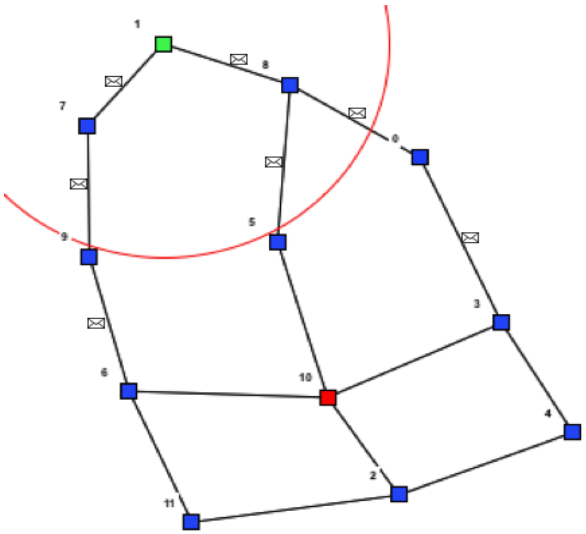


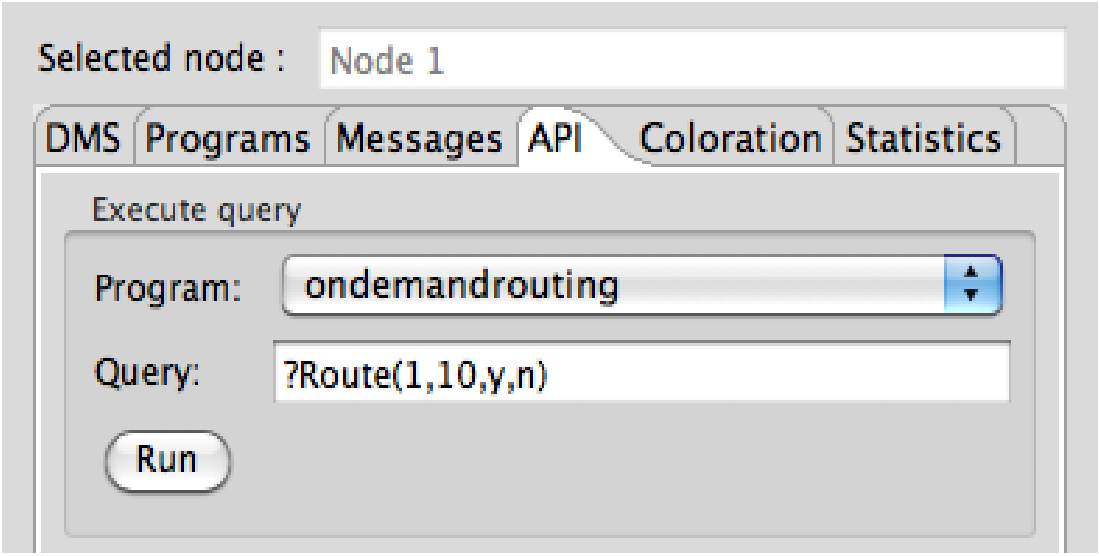
Simulation results



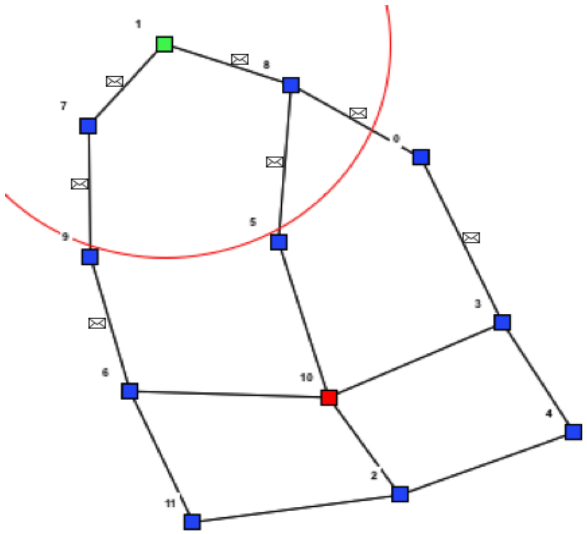


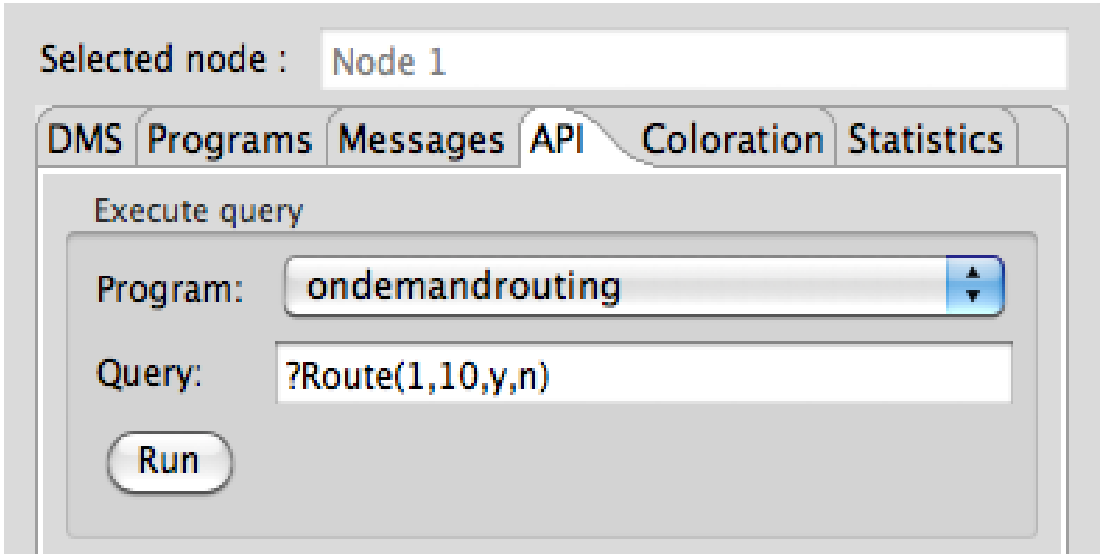
Simulation results

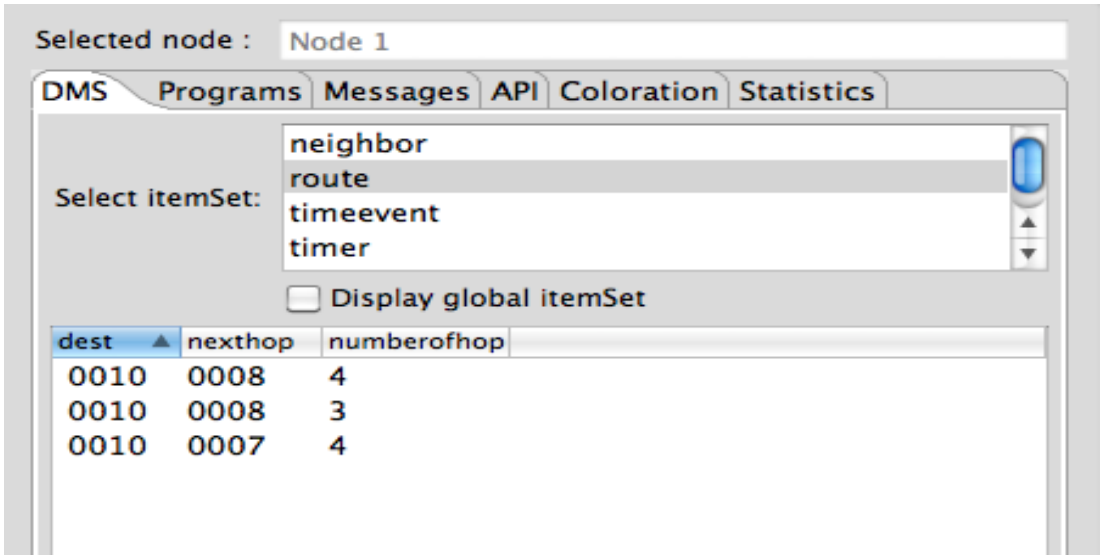




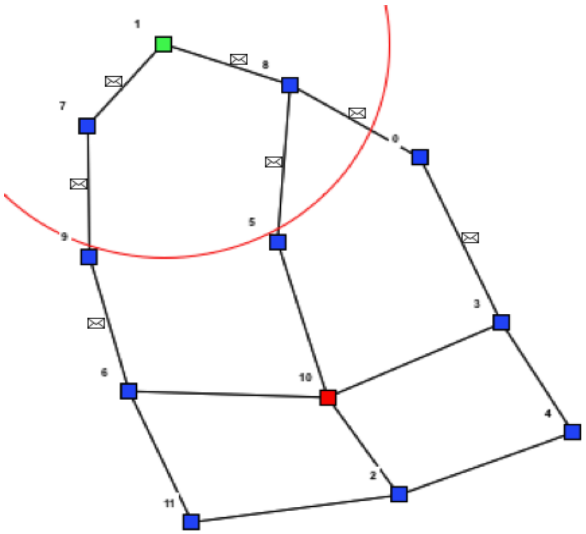
Simulation results

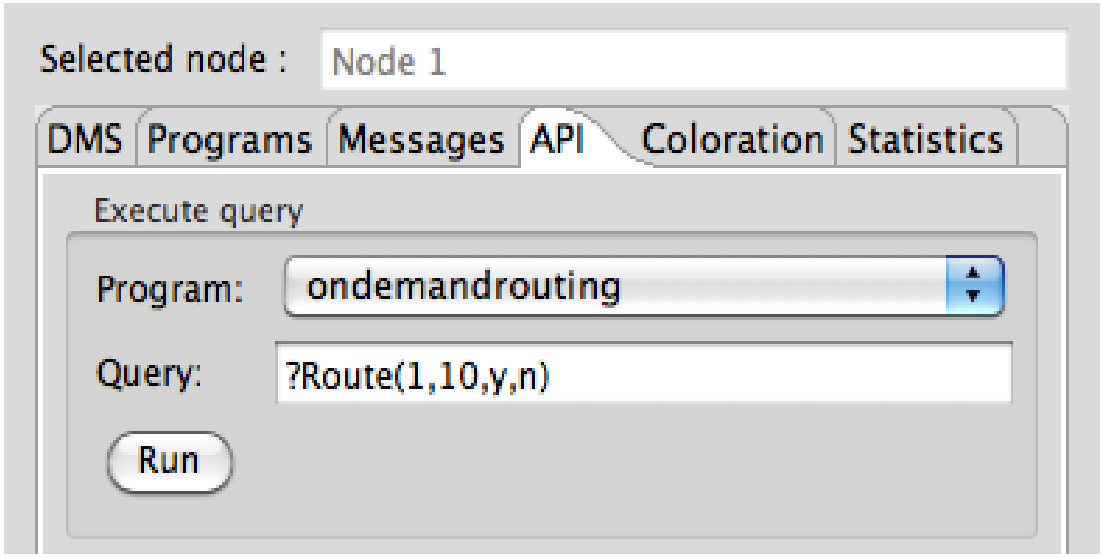


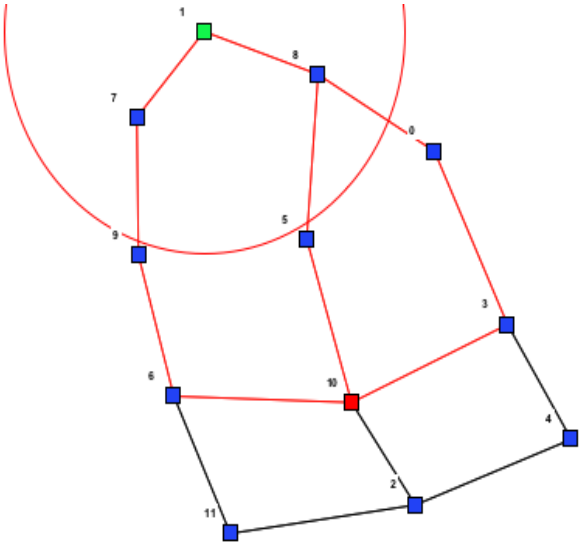


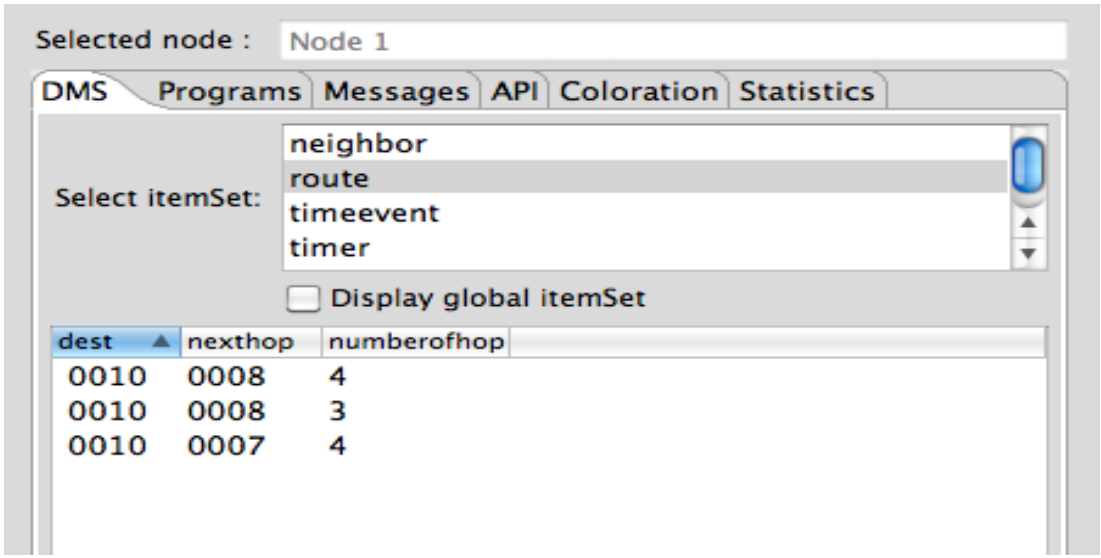


Simulation results









Outline

Conclusion

Proposition of a framework

- program applications in a message oriented manner
- provide abstraction of the destination of message
  - expressed declaratively by a query
  - use of the Questlog language

Conclusion

Proposition of a framework

- program applications in a message oriented manner
- provide abstraction of the destination of message
  - expressed declaratively by a query
  - use of the Questlog language

Current work

- experimenting the different strategies of programming (balance the load between payload and destination), and studying the resulting overhead

Conclusion

Proposition of a framework

- program applications in a message oriented manner
- provide abstraction of the destination of message
  - expressed declaratively by a query
  - use of the Questlog language

Current work

- experimenting the different strategies of programming (balance the load between payload and destination), and studying the resulting overhead

Future work

- address social network applications using this framework