SOFTWARE SECURITY RESEARCH GROUP
In Collaboration With IBM

A Scalable P2P RIA Crawling System with Partial Knowledge

Khaled Ben Hafaiedh, Gregor v. Bochmann, Guy-Vincent Jourdan, Iosif Viorel Onut

School of Electrical Engineering and Computer Science - University of Ottawa

Introduction – Traditional vs. Rich Internet Applications

Traditional Web Applications

- The typical interaction between the client and the server in a traditional web application consists of sending a request for a URL from the client to the server so that the corresponding web page is downloaded in response for each URL request.
- Each web page is identified by its URL and has only a single state.

Rich Internet Applications

- Modern web technologies gave birth to interactive and more responsive applications, referred to as RIAs.
- RIAs combine the client-side scripting with new features such as AJAX (Asynchronous JavaScript and XML).
- JavaScript functions allow the client to modify the currently displayed page, by communicating with the server asynchronously.

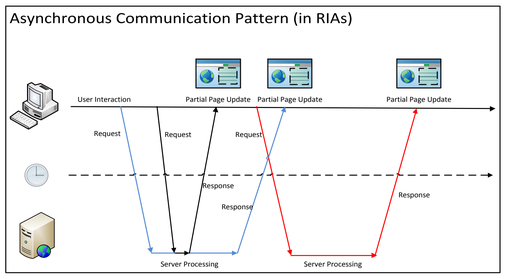


Figure 1. Asynchronous Communication Pattern in RIAs

RIA Crawling

The purpose of a RIA crawler is to automatically exploring states of a rich internet application.

Goal

- Content indexing
- Testing for security
- Building application models

Distributed RIA Crawling

Aim

- Distributed centralized crawling has been introduced to reduce the amount of time required to crawl RIAs.
- It consists of running multiple crawlers simultaneously and sharing the searching space in a single storage unit, called the controller.

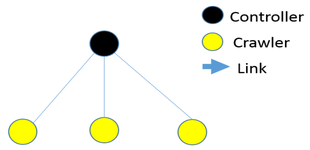


Figure 2. A Distributed Centralized Crawling System.

Challenges and Motivation

- **Scalability:** The controller may become a bottleneck when it is accessed simultaneously by a high number of crawlers.
- **Fault tolerance:** A failure occurring within the controller may result in the entire loss of the graph under exploration.

Contributions

- **Scalability:** The distribution of responsibilities for the states among multiple controllers in the underlying P2P network, where:
  - Each controller maintains a partial model of the application
  - A high number of crawlers are associated with each controller, which allows for scalability.
- **Knowledge sharing:** Defining and comparing different knowledge sharing schemes for efficiently crawling RIAs in the P2P network.

Architecture

- A peer-to-peer crawling system composed of multiple controllers in the form of a chordal ring.
- States are partitioned into disjoint sets, each of which is handled by a distinct controller.
- Each controller is associated with a certain number of crawlers.
- Crawlers and controllers are independent processes running on different computers.
- Each crawler gets access to all controllers through a single controller it is associated with.

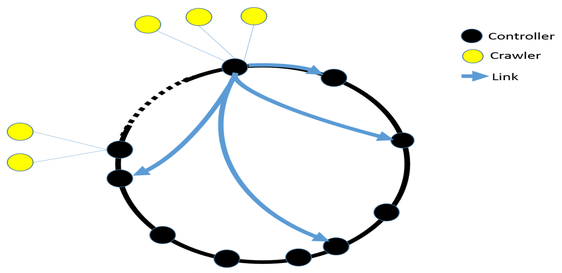


Figure 3. Distribution of states and crawlers among controllers.

Methodology

The greedy strategy

- **With one single crawler**
  - Exploring an event from the current state if there is any unexplored event.
  - Otherwise, the crawler may execute an unexplored event from a different state, until all transitions are traversed.
- **Distributed centralized system**
  - Each crawler may retrieve the required graph information by communicating with the single controller.
  - It then executes a single unexecuted event from its current state if such an event exists, or may move to another state with some unexecuted events based on the information available on the single controller.
- **In the P2P environment**
  - States are partitioned among the controllers.
  - The controller responsible for storing the information about a state is contacted when a crawler reaches a new state.
  - For each request, the controller returns in response a single event to execute on this state.
  - If no event can be executed on the current state of the crawler, the crawler may communicate with other controllers to execute events from another state.

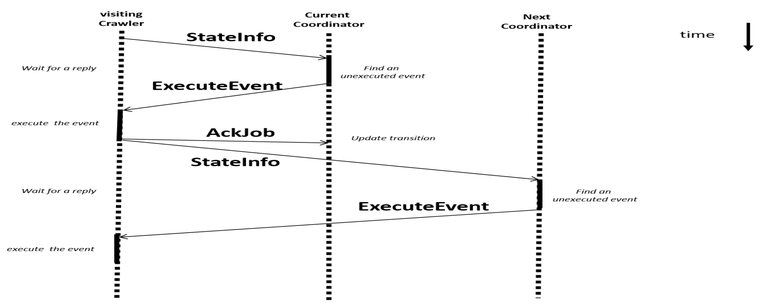


Figure 4. Exchanged messages during the exploration phase.

Challenges

- **Efficiency:** Crawlers must efficiently execute the graph transitions by only communicating with as few controllers as possible.
- **Termination detection:** An idle crawler cannot know a priori if all transitions on all states that are maintained by different controllers have been already executed or not.

Choosing the next event to explore from a different state

Four approaches for finding and executing events on a state other that the current state of the crawler in the P2P crawl system:

Reset-Only

- A crawler can only move from a state to another by performing a Reset.
- Reset-Only is the simplest way for distributively crawling RIAs.
- However, this approach results in a high number of resets performed, which may increase the time required to crawl a given application (cost).

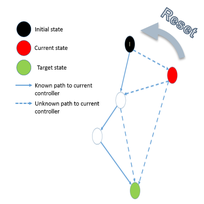


Figure 5. Possible path in the Reset-Only approach

Shortest Path based on local knowledge

- A controller may use its local transitions knowledge to find the shortest path from the crawler current state leading to the closest unexecuted event.
- Since the knowledge is partial, the number of known transitions to a controller remain relatively low.

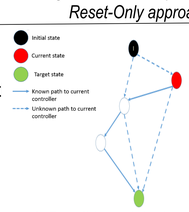


Figure 6. Possible path in the shortest path based on local knowledge approach

Shortest Path based on shared knowledge

- The executed transitions are locally stored by intermediate controllers when a message is forwarded through the DHT.
- The transitions knowledge is significantly increased among controllers with no message overhead.

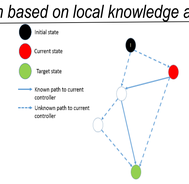


Figure 7. Possible path in the shortest path based on large knowledge approach

Forward-Exploration

- It consist of globally finding the optimal choice based on the breadth-first search, by sequentially moving to the neighboring states from the current state of a visiting crawler.
- Preventing different controllers from visiting states that have already been visited by other controllers and have no unexecuted events:
  - Controllers may share during the forward exploration their knowledge about all executed transitions on these states, with other controllers.
  - A controller can only jump over a visited state if and only if all transitions have been executed on that state.

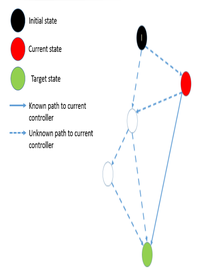
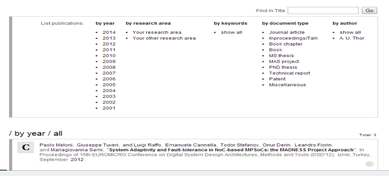


Figure 8. Optimal path with the Forward-Exploration approach

Results

Tested large-scale application: Bebop RIA

- 5,082 states
- 468,971 transitions
- Direct Link: http://ssrg.eecs.uottawa.ca/bebop/



Cost in time of decentralizing the crawling system

- Based on our preliminary analysis of experimental results, a controller can support up to 20 crawlers without becoming overloaded.
- We plot the simulated time (in seconds) for an increasing number of controllers from 1 to 20, with steps of 5, while the number of crawlers is constant and set to 20 crawlers.
- We compare our results to the ideal performance (Global Knowledge scheme) where all controllers have instant access to a globally shared information about the state of knowledge at each controller.

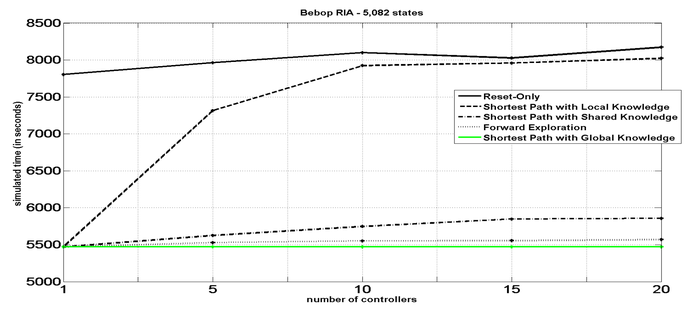


Figure 9. Comparing different sharing schemes for crawling the Bebop RIA.

- The worst performance is obtained with the Reset-Only strategy, followed by the Shortest Path with Local Knowledge strategy (Due to Resets performed and partial knowledge).
- The Shortest Path based on shared Knowledge strategy comes in the second position and significantly outperformed both the Reset-Only and the Shortest Path based on Local Knowledge strategies as controllers has more knowledge about the application graph.
- The best performance is obtained with the Forward Exploration strategy by finding globally the optimal choice based on the distributed breadth-first search.

Scalability of our approach

- We consider the strategy with the best performance (Forward Exploration) and we plot the simulated time (in seconds) for an increasing number of controllers from 1 to 5, with 20 crawlers for each controller.

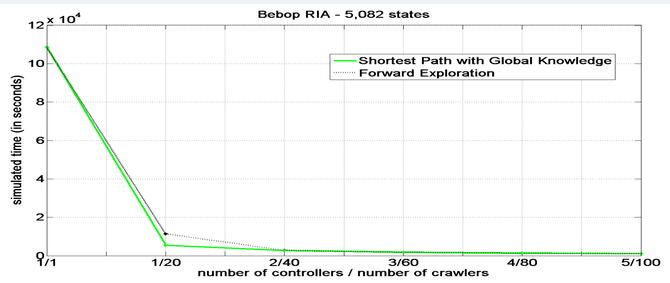


Figure 10. System scalability for crawling the Bebop RIA.

- Our simulation results show that the crawling time decreases near optimally as we increase the number of crawlers.
- We conclude that our system scale with the number of crawlers when the controllers are not overloaded.

Conclusion & Future Work

- Simulation results show that the Forward Exploration strategy outperforms the Reset-Only, the Shortest Path based on Local Knowledge and the Shortest Path based on Shared Knowledge strategies.
- This is due to its ability to globally find a shortest path, compared to all other strategies that are based on partial knowledge.
- This makes Forward Exploration a good choice for general purpose crawling in a decentralized P2P environment.
- Some of the areas that we are currently working on are:
  - Fault-tolerance.
  - Applying other crawling strategies such as the menu model, the component-based model and the probabilistic strategy.

Acknowledgments

This work is supported in part by IBM and the Natural Science and Engineering Research Council of Canada.

DISCLAIMER
The views expressed in this poster are the sole responsibility of the authors and do not necessarily reflect those of the Center for Advanced Studies of IBM.