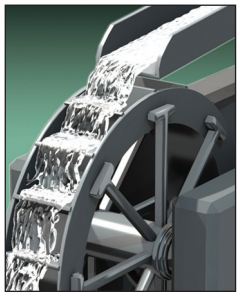
Editor: Schahram Dustdar • dustdar@infosys.tuwien.ac.at



The Social Routing Principle

Schahram Dustdar • *Technical University of Vienna*

Martin Gaedke • *Technical University of Chemnitz*

This article discusses how modern Web-scale workflows can be built using a novel building block to be integrated into social networks as well as apps. This social routing principle fosters the integration of social networking systems and mobile apps as well as Web systems.

Social networks, apps (smart phone apps, widgets, and Web applications[1]), and collective intelligence have recently gained momentum. Social networks and their tools are increasingly changing how people work as individuals as well as in teams and communities. The notion of collaborative problem-solving[2] is becoming an increasingly adopted work style. We face many challenges and opportunities in our approach to problem-solving using these technologies as we move forward to a socially connected global place for knowledge work. The challenge in this age of social software is thus to move beyond applying social computing by individuals and to understand how an entire society can organize or, respectively, program itself to unleash its full potential to solve societies' problems.

Here, we look at one fundamental principle that we suggest as a core building block for addressing the design of future social Web-scale workflow systems: the *social routing principle*.

Integrating Apps and Social Networks

Today, we use various technical app implementations mainly as a personalized way to work on individual tasks or activities (in isolation), such as checking flights, reading news, taking pictures, and sending messages in various forms. Communities use social networks and their predominant representatives, such as Facebook and Twitter, for communication among members. However, these networks provide little, if any, support for actually working on tasks in a problem-solving context. Moreover, except for authentication purposes, most apps we use today are fully decoupled from social networks. Some increasingly widespread apps, on the other hand — such as FourSquare (www.foursquare.com) and Quora (www.quora.com) — are in fact supporting their own built-in and domain-specific social networks by routing and connecting traffic from well-known networks to their own. Hence, today we can say that apps are mainly perceived as *vertical*, or domain-specific, solutions for individuals. There's nothing wrong with this per se because it leads to economies of scale — but only within a domain-specific (vertical) focus.

Our fundamental assumption is that, as social networks increasingly become an infrastructure in their own right, we must develop and compose more apps *horizontally* to fully support Web-scale workflows for individuals and businesses alike. In other words, mechanisms that include parts of a social network or the crowd must become integral to an addressing mechanism for task delegations from domain-specific apps.

Having said so, we can delegate tasks to three different types of networks: personal communities (such as Facebook friends or Twitter followers), context-based communities (FourSquare, Local Minds, Quora, and so on), and the crowd (for example, the Amazon Mechanical Turk). A personal community is composed of the owner's

Published by the IEEE Computer Society
1089-7801/11/$26.00 © 2011 IEEE

friends or contacts. A context-based community is defined by all parties interested in a certain topic — that is, who share context — or who, for example, share a common location. A crowd by definition comprises all existing (and anonymous future) platform members.

Social Computing Environment Evolution

We can view Twitter and Facebook as very successful generic social computing environments because they provide the infrastructure for dealing with many different kinds of problem–solution scenarios using the power of human language.

In Twitter, "please RT" is a short phrase for "please re-tweet," which means that other people who read a given message (or *tweet*) should post it to their own followers, usually via Twitter's automatic re-tweet feature or using the older approach of re-posting:

RT Peter: Need help w/ #ProjectWorkplanStructure. Please, RT.

Usually, only those who are also interested in the problem will propagate the tweet. These people delegate the problem to their follower networks and, due to tweets' nature, again to the channel #ProjectWorkplanStructure. Tweets and terms that are often re-tweeted can therefore become a trending topic creating even more visibility and, as such, increasing the probability that an expert will finally solve the problem. We can also observe this well-known phenomenon in other social network contexts, such as community message boards, where users commit others' problems by sending comments such as "I suffer from the same problem, please send me a solution, too. Thanks!"

The major challenge for problem-solving in social computing environments is obviously creating an appropriate reach. In the context of Twitter, you might achieve this by enlarging your social network — that is, by having thousands of followers or becoming a person people talk about (in the sense of re-tweeting messages). Several applications, such as klout.com, focus on this social reach and provide different measures that might be relevant to predicting the chances that someone's social network will successfully solve a certain problem.

Twitter, like Facebook, provides a generic social computing environment that people can use to address the problems we discuss here. Such generic social computing environments are available for all sorts of problem-solving; however, we believe that task delegation mechanisms must improve if we're to build Web-scale workflows using such social networks. Initial attempts to include workflow features using Twitter are available elsewhere.[3,4]

The problem-solving capabilities we find in Twitter and Facebook induce new ideas for further generations of dedicated social computing environments. Examples include FourSquare, which focuses on a user's location, and Quora, which aims at answering users' questions. By providing a context, users can solve problems more efficiently; additionally, the dedicated problem domain better enables the environment or its users to create or provide incentives.

For instance, FourSquare creates different incentives urging users to contribute and behave with regard to its social rules — that is, to create or categorize new locations and provide information about users' whereabouts by honestly checking in at given locations. All these activities are rated with points and the chance to collect badges for certain events — for example, checking in at several airports will get you the "jet-setter badge." The social network maintains user loyalty by using game practices. While Twitter and Facebook present users' messages, Quora presents messages (usually answers) for questions raised. As in FourSquare, additional psychological aspects come into play, allowing for better-suited incentives than in generic environments; in Quora, the incentive is to reach "expert" status based on how other users rate the answers you've provided.

Problem-Solving Using a Social Router

So, how can we enrich the vertical (domain-specific) problem-solving approach with the power of horizontal problem-solving capabilities? As Figure 1 shows, app users have a certain amount of problem-solving capability and capacity they can use to solve the task. Either task owners can solve a task, or they delegate it to their personal community, a community in their social network, a context-based community, or a crowd. Comfortable task delegation utilizing social networks or the crowd isn't foreseen or currently possible with most apps. We propose introducing mechanisms for task delegation in apps, thereby scaling individual work as well as collaborative work on the Web.

Illustrative Example

Joe is a senior manager for a software company, SmartCorp. His calendar for today contains three entries:

- 10:00 a.m. to 12:00 p.m., write Q3 report
- 1:00 p.m. to 2:00 p.m., lunch with Jack (MarketingCorp)
- 5:00 p.m. to 7:00 p.m., find pictures of SmartCorp at CeBIT exhibition on the Net

In the morning, Joe receives an urgent call asking him to see a mission-critical customer from 10:00 a.m. onward. This customer's priority

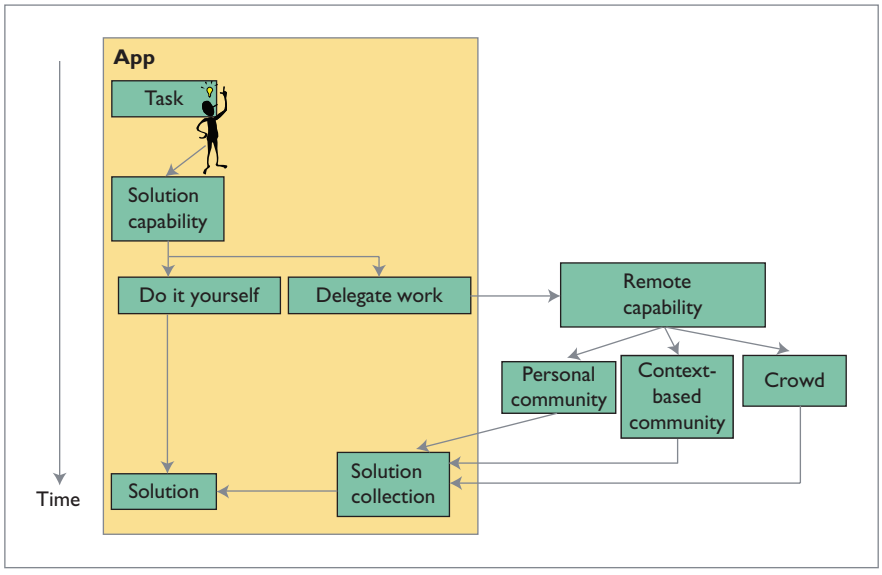


Figure 1. Problem-solving approach with apps and social networks. A task at hand might be solved in two ways: either in a "do it youself" manner or by delegating it directly from the app. This delegation of work could target some or all of the three remote capabilities: personal social networks, specialized communities, or the crowd.

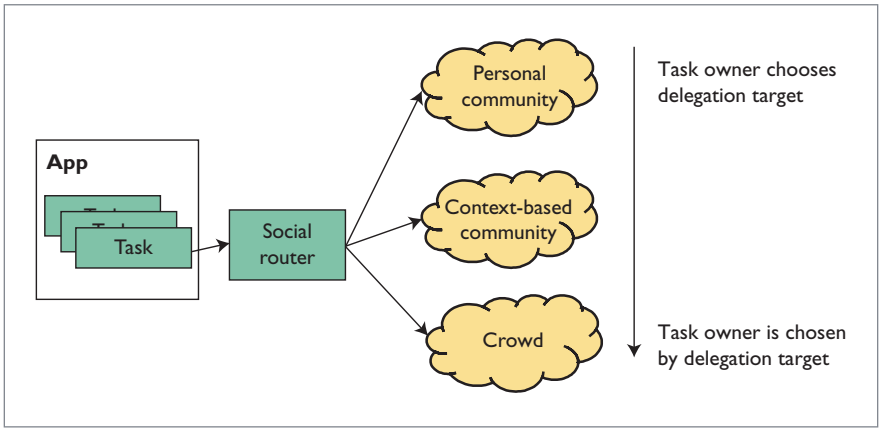


Figure 2. The social routing principle. A task is delegated either to a specific personal community — that is, the task owner knows about and chooses the solution capacity — or to a context-based community or crowd. In this context, the task might be solved when the targeted community or crowd's solution capacity chooses it.

demands that Joe move all his calendar entries for that day. He opens his calendar app, which is social-router enabled, selects the first entry, and clicks the delegate button. A pop-up menu appears showing three types of communities. Based on this task's nature, he selects the personal community, which is composed of senior members in his software unit. Then he selects the meeting entry and delegates it to a context-based community – that is, to members of his software company who happen to be near the restaurant. Finally, he selects the "find pictures" task and delegates it to the crowd – that is, the social router creates a *human intelligence task* (HIT) for Amazon Mechanical Turk automatically containing the task description (find pictures of SmartCorp at CeBIT exhibition on the Net). Because Joe delegated this task to a paid crowd, the app asks Joe to enter the amount he's willing to pay.

Social Router Features

In computer networks, a router is a device that forwards data packets to its connected networks by taking their different capabilities and requirements into account. It draws its routing decisions based on a local routing table. When connecting technologically different networks, the router translates the data packets and follows the rules of the communication protocols in place.

A social router requires a similar approach (for instance, implemented as a software service) that can forward (delegate) tasks across different social networks (personal communities, context-based communities, or the crowd). As Figure 2 depicts, the social router can forward its tasks to these three types of networks. To do so, the router's conceptual routing table must provide a means for how and to where these can be forwarded. As such, this conceptual routing table must know how to bind to the different target networks – that is, how to apply address schemes and API contracts based on the user's personal preferences.

Challenges and Opportunities

As the Web's reach and its social networks evolve, the number of specialized social networks increases. Generic problem-sharing platforms such as Amazon Mechanical Turk become more attractive and host ever larger audiences of remote solution capabilities – in the same way, the probability increases that this mass of people might include individuals who are willing to solve a routed task. Given this necessary core requirement for applying the social router principle in

all kinds of apps, we face several challenges leading to exciting new opportunities, as we mentioned previously.

The following three topics address some of these challenges and are a starting point for demonstrating new areas of research.

Workflow-based delegation models. Delegating tasks requires the social router to translate a given task into representations of the corresponding target networks' tasks. In the illustrative example we described, the context of the tasks to be delegated was given in the calendar entries and by Joe himself. More complex delegation mechanisms must incorporate mechanisms such as trust and privacy, incentives, and the more sophisticated addressing schemes of the communities addressed.

Managing context. The following simple example of a task received via email expands on the mechanisms that a social router must manage:

Dear Jim,

Please check the attached demonstration picture of our new project to determine if it's compliant with our partner rules. I need your reply by tomorrow 3 p.m.

Best regards, Joe

Humans might easily understand this task, but it's already very complex for the social router. The router must deal with certain questions: What is the new project? The social router must understand context: Who are the partners? The social router must know about the related parties and potentially about legal aspects, which forbid forwarding a task to every community. Besides addressing and contract issues, a social router must take privacy and trust into account when sending task descriptions to other communities.[5]

Humanistic socio-technical issues. Finally, a social router must take care of another parameter — that is, the human factor. Based on task complexity, time frame, and other aspects, the target network's delagatee expects incentives for solving the task. Several possibilities exist for implementing such incentives — providing money, virtual currencies (such as points in frequent flyer programs or Facebook credits), virtual points, goods and badges, or other means — to increase the chance of receiving a solution. The data for describing a task is often available online and machine-usable. To make the social router a reality, the apps must support the user in formulating tasks that the social router can then translate.

We believe that the social routing principle fosters a novel and more efficient approach to utilizing personal and context-based communities as well as the crowd in general. Applying this principle accelerates individual as well as collaborative work by combining apps, the Web, and social networks into a collaborative ecosystem for Web-scale problem-solving.

References

1. E. Wilde and M. Gaedke, "Web Engineering Revisited," *Proc. Visions of Computer Science — BCS Int'l Academic Conf.*, British Computer Society, 2008, pp. 41–49.
2. S. Dustdar, "Caramba — A Process-Aware Collaboration System Supporting Ad Hoc and Collaborative Processes in Virtual Teams," *Distributed and Parallel Databases*, special issue on teamware technologies, vol. 15, no. 1, 2004, pp. 45–66.
3. M. Treiber et al., "Tweetflows — Flexible Workflows with Twitter," *Proc. 3rd Int'l Workshop on Principles of Eng. Service-Oriented Systems, 33rd ACM/IEEE Int'l Conf. Software Eng.* (ICSE 11), to appear, 2011.
4. M. Böhringer and M. Gaedke, "Ubiquitous Microblogging: A Flow-Based Front-End for Information Logistics," *Lecture Notes in Business Information Processing* (LNBIP 57), Springer, 2010, pp. 158–167.
5. F. Skopik, D. Schall, and S. Dustdar, "Modeling and Mining of Dynamic Trust in Complex Service-Oriented Systems," *Information Systems*, vol. 35, 2010, pp. 735–757.

Schahram Dustdar is a full professor of computer science (informatics) with a focus on Internet technologies and heads the Distributed Systems Group, Institute of Information Systems, at the Vienna University of Technology (TU Wien). Dustdar is an ACM Distinguished Scientist. Contact him at dustdar@infosys.tuwien.ac.at; www.infosys.tuwien.ac.at/.

Martin Gaedke is a full professor of computer science at the Chemnitz University of Technology, head of the professorship for distributed and self-organizing systems, and a guest professor at TU Wien. His research focuses on applying the Web for massively distributed collaboration. Contact him at gaedke@informatik.tu-chemnitz.de; http://vsr.informatik.tu-chemnitz.de/people/gaedke/.