Building Event-Based Services for Awareness in P2P Groupware Systems

Alex Poulovassilis, Fatos Xhafa, Tom O'Hagan

3PGCIC2013
October 2013

Motivation

- Monitoring and awareness are essential in collaborative work environments, in order to support team members' interaction and coordination processes
- There has been much work in implementing awareness in web-based applications, but less so for P2P collaborative systems.
- P2P systems can potentially foster more support for collaboration than centralized approaches as group members can interact directly with their peers and can share their knowledge, skills and resources in order to provide mutual support in the accomplishment of group tasks
- ***Awareness*** is a vital aspect of collaborative systems, and refers to knowledge provided by the system to group members about what other group members are doing at the same time and what they did in the past

Motivation

- However, several challenges arise in developing groupware and awareness mechanisms in decentralized P2P systems:
  - achieving a consistent view of the groupware global state from the local states of the group members
  - issues inherent to P2P systems such as their dynamicity and heterogeneity
  - P2P systems are pervasive and ubiquitous in nature, thus requiring contextualized awareness, e.g. event transformation and enrichment may be necessary
  - awareness mechanisms in P2P systems have generally been implemented as part of applications, thus having the limitation of being application-dependent

Our work

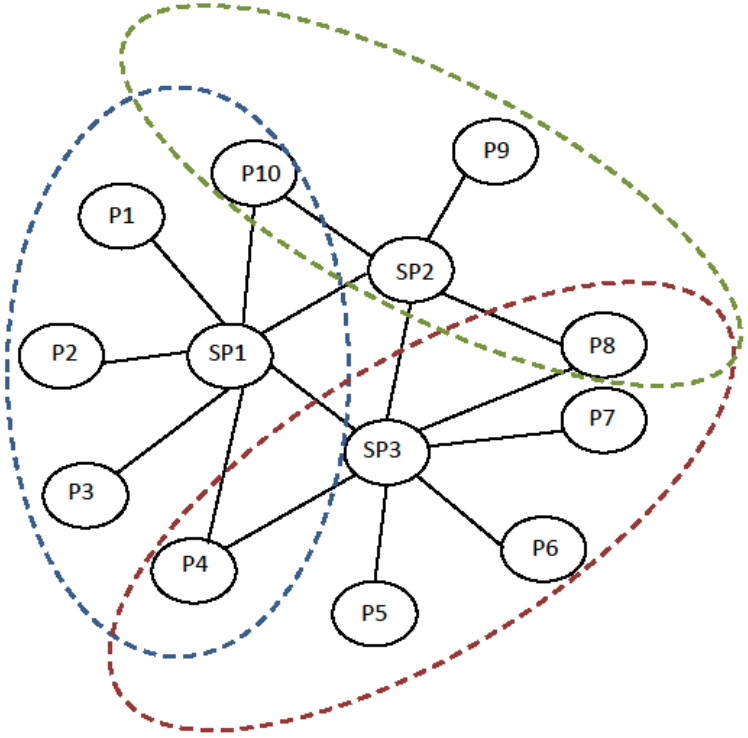
- Provision of ***service-based group awareness capabilities in P2P middleware,*** on top of which groupware applications can be developed
- We identify a set of low-level awareness services and show how these can interoperate over the P2P network in order to provide awareness as part of the P2P middleware infrastructure
- We envision the use of service composition to provide more complex awareness services
- We envision the implementation of a superpeer P2P network model on a Cloud platform and the provision of reliable awareness services from the Cloud
- Our motivating P2P collaboration setting is that of Project-Based Learning (PBL), although our approach is intended to apply more generally to project-based collaboration in other sectors e.g. business, science, healthcare

Major types of Group awareness requirements

- Activity awareness
- Process awareness
- Communication awareness
- Context awareness
- Availability awareness

Computational Model

- We propose an RDF/S-based superpeer network for supporting group awareness in P2P groupware systems:



Computational Model

- Peers in a peergroup can communicate directly with their superpeer and with each other
- Concretely, the P2P network will typically be an overlay network
- Each project is coordinated by one superpeer – though superpeers may coordinate multiple projects.
- Peers contact the relevant superpeer in order to join a project — we term the set of peers working on a project a *project group*, or just a *group*
- Peers may join or leave a group at any time
- Services hosted at superpeers control and coordinate the peergroup's activities towards accomplishing one or more projects
- Such services include consistency control, distribution and replication of information, maintaining the project workflow constraints, building reports and summaries, and handling requests for resources from peergroup members

Computational Model

- Information about the group's processes can be distributed and replicated at peers of the group
- Peers' operations and control information is forwarded to their superpeer which manages the replication and consistency of information within the group
- This enables efficiency due to local access to data and support of failures e.g. when peers leave or are temporarily disconnected
- Information at peers is stored in local repositories; we assume it is encoded in RDF/S
- The data in these RDF/S repositories will include descriptions of objects, workspaces, resources, tasks, sessions, possible actions etc., as relating to the groupware application
- Our approach aims to support both stand-alone mobile peers and mobile peers that are attached to other fixed peers through lightweight mechanisms and summary services

Computational Model

- Each superpeer has *Event-Condition-Action (ECA)* rule processing capabilities. ECA rules hosted at the superpeer can be used to:
  - achieve replication and consistency of distributed group data and processes – the replication and consistency policies are encoded using ECA rules
  - automate the generation and propagation of global overviews and summaries from detailed information and local summaries received from individual peers
  - automate the receipt of awareness information by peers, according to their current status, status of the project they are participating in, their preferences, and their context
- Updates initiated at a peer site are executed locally and are then notified to the superpeer. This determiness whether the update triggers an ECA rule, or rules, causing the stated rule actions to be propagated to other peers in the peergroup

Awareness services

- *Core Services,* necessary to support the group awareness services:
  1) Peer services: Event Notification, Repository Connection, Messaging, Peer Resource Information, Object Sharing, Synchronous forum
  2) Superpeer services: Routing, ECA Rule Processing, Rule Base Management, Synchronous forum management

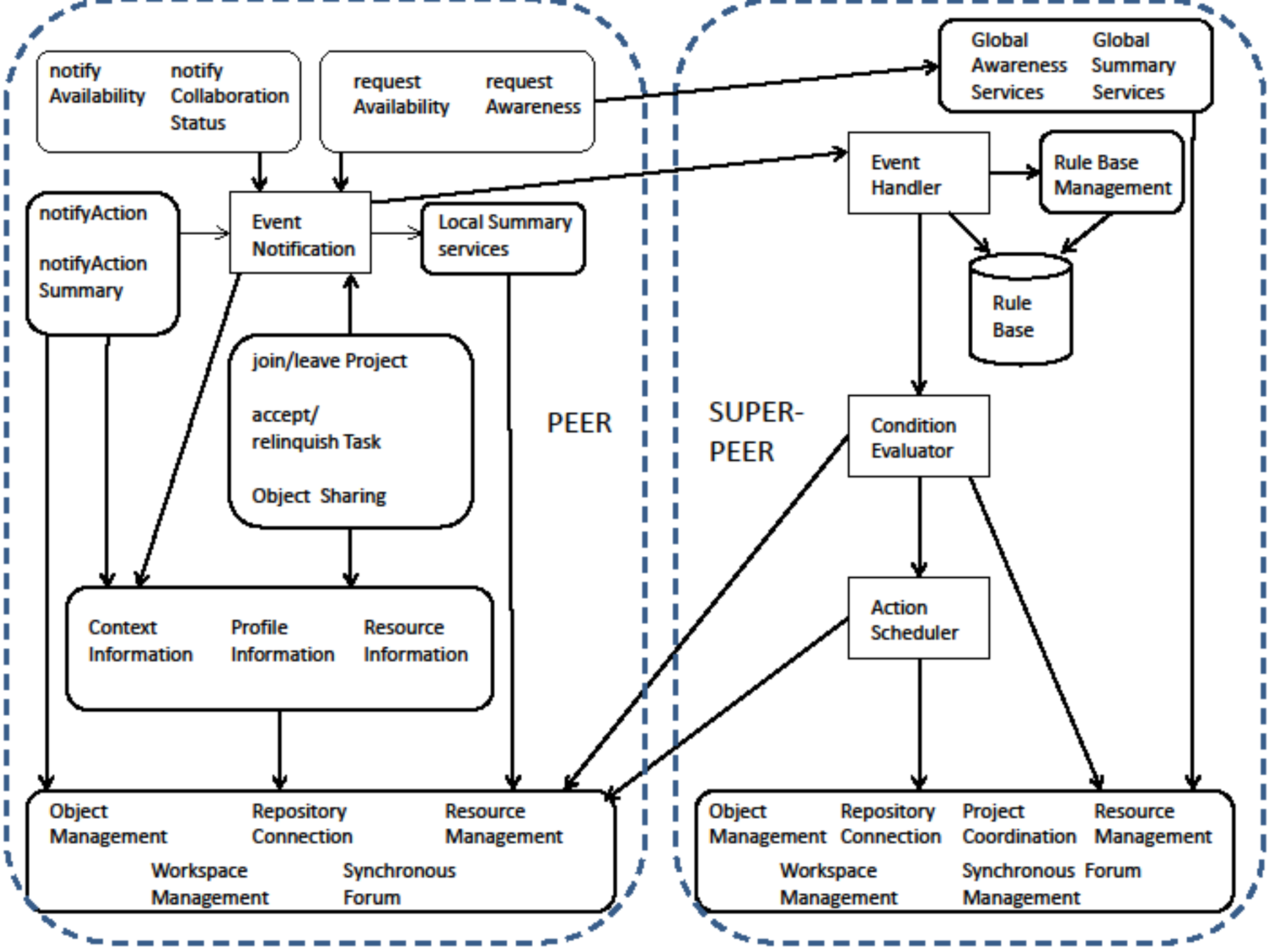
- *Group Awareness Services:*

  1) Peer services:

  - joinProject, leaveProject, acceptTask, relinquishTask – support activity awareness, process awareness
  - notifyAction, notifyActionSummary – support activity, process, context and communication awareness
  - notifyCollaborationStatus, notifyAvailability, requestAvailability – support availability awareness
  - requestProjectSummary, requestTaskSummary, requestPeerSummary – support activity awareness and process awareness
  - requestAwarenessSummary – supports all types of awareness

  2) Superpeer services:

  - assignProject
  - provideProjectSummary, provideTaskSummary, providePeerSummary – support activity awareness and process awareness
  - provideAwarenessSummary – supports all types of awareness


notify Availability
notify Collaboration Status
request Availability
request Awareness
Global Awareness Services
Global Summary Services
notifyAction
notifyAction Summary
Event Notification
Local Summary services
Event Handler
Rule Base Management
Rule Base
join/leave Project
accept/ relinquish Task
Object Sharing
PEER
SUPER-PEER
Condition Evaluator
Action Scheduler
Context Information
Profile Information
Resource Information
Object Management
Repository Connection
Resource Management
Workspace Management
Synchronous Forum
Object Management
Repository Connection
Project Coordination
Resource Management
Workspace Management
Synchronous Forum Management

Implementation

- We have implemented a prototype providing the core peer and superpeer functionality detailed above
- This prototype has been used in a preliminary performance evaluation, detailed below
- Our intention is to allow application programmers to use the system as middleware for building other P2P applications
- Application programmers can flexibly provision awareness services using the system
- The prototype system is implemented in Java
- We use the Apache Jena API for RDF/S processing
- For data storage we use the Jena Triplestore (TDB); each peer and super-peer hosts a local TDB repository

Implementation

- Event notifications (sent from peers to superpeers) are RDF graphs encoded in RDF/XML
- These notifications trigger ECA rules at the superpeer
- ECA rules are encoded in XML and supplied by the application programmer to the superpeer.
- These rules are templates for XML-RPCs that are used to trigger actions at peer nodes
- The parameters used in each XML-RPC and the destination of each XML-RPC is resolved at runtime by the superpeer through its ECA rule-processing capability

Performance evaluation

- A preliminary performance evaluation has investigated the scalability of the system
- The test scenario we used was *document synchronisation*, as this likely to be the most resource-intensive functionality required
- In this test scenario, collaborating members of a peergroup hold local copies of a shared document; and when a peer makes a modification to the shared document, we measure the average time it takes for details of this modification to be received by all the other owners of the shared document (so that they can make the requisite updates to their own copies of the document)
- Variables under consideration in the performance evaluation are:
  - The number of peers in a peergroup
  - The fraction of these peers that have a local copy of the shared document (0..1)
  - The size of the update made to the document

Communication Protocols

We studied the performance of three different communication protocols for propagating the updates within the peergroup:

Protocol 1:

- A peer (*Pinit*) updates a local copy of a shared document (*doc*)
- *Pinit* sends an event notification (*ev*) plus the document *'patch'* to its superpeer
- The superpeer executes the ECA rules that are triggered by this event notification and sends *ev* plus *'patch'* to all peers who hold a copy of *doc* (*Ptarget*). These peers update their copies.

Protocol 2:

- Peer *Pinit* updates *doc* and sends *ev* plus *'patch'* to the superpeer
- The superpeer does not execute any ECA-rules. It sends *ev* plus *'patch'* to all peers in the peergroup
- If a peer holds a copy of *doc*, it updates it accordingly.

Communication Protocols

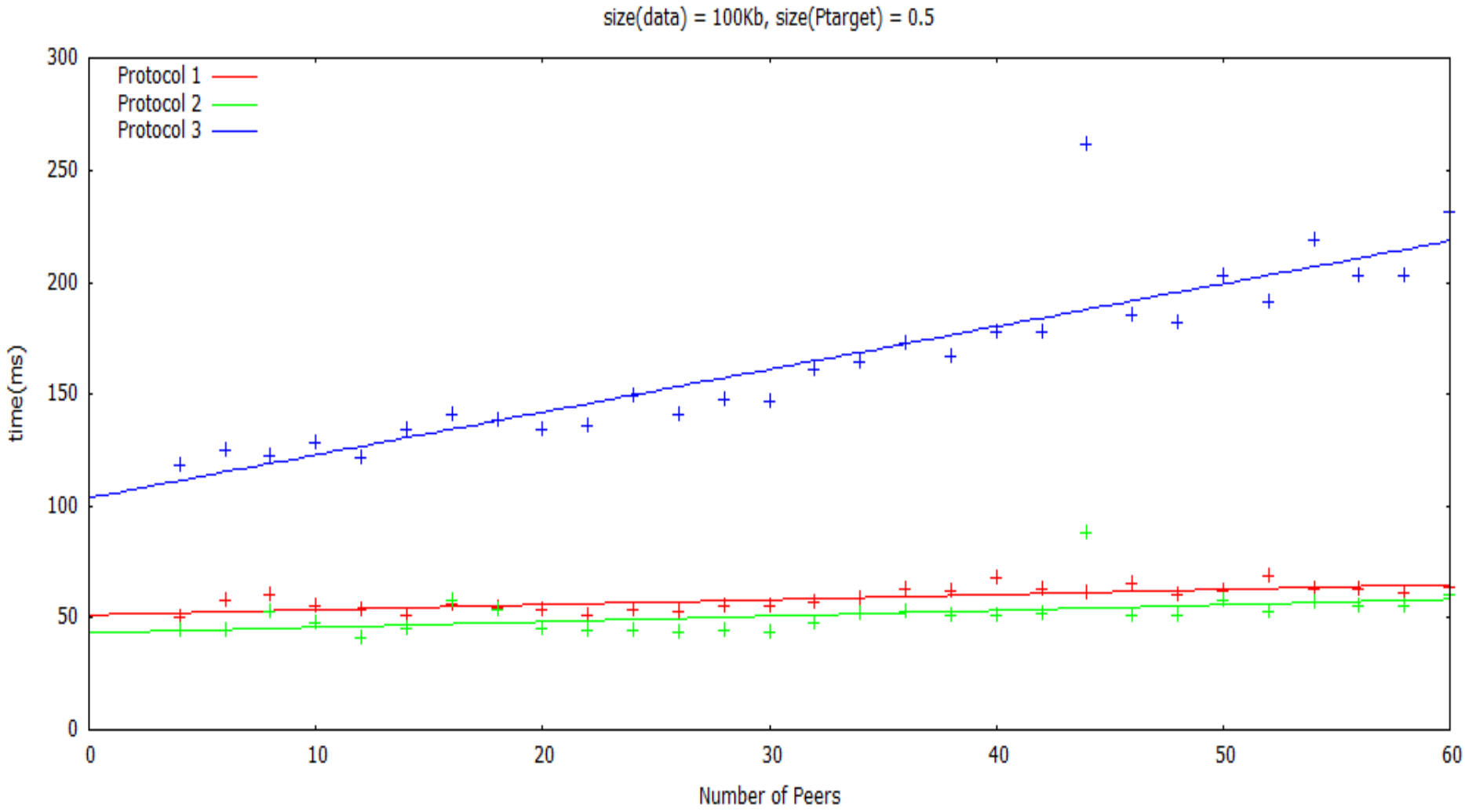
Protocol 3:

- *Pinit* updates its local copy of *doc*
- *Pinit* sends *ev* to its super-peer
- The superpeer executes ECA rules that are triggered by *ev*. This process identifies which peers hold a local copy of *doc* (*Ptarget*)
- The superpeer sends details of the peers in *Ptarget* back to *Pinit*
- *Pinit* sends the *'patch'* plus accompanying metadata to all peers in *Ptarget*
- These peers update their local copies of *doc*

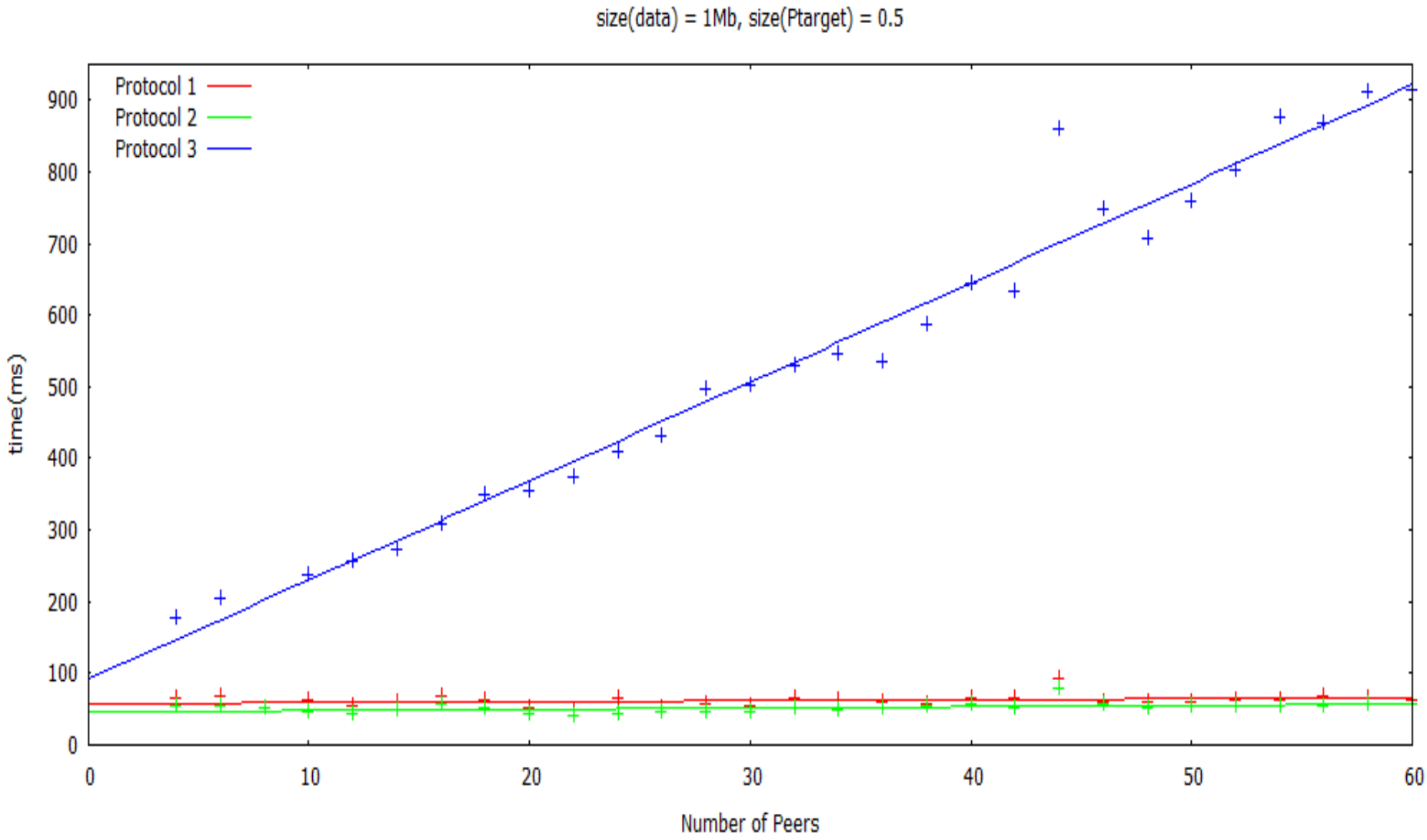
Cloud Platform

- We used the Amazon Web Services (AWS) Cloud Computing platform for the performance evaluation
- This allowed us to flexibly and cheaply access the computing resources required to simulate the P2P network
- Each peer and superpeer node was run on a dedicated Elastic Compute Cloud (EC2) virtual machine ('instance').
- Peers were run on low-spec t1.micro instances
- Superpeers were run on higher-spec m1.large instances
- 64-bit Linux AMIs were used for both types of instance
- For the experiments, peer nodes did not maintain continuous connections to other peers, apart from the superpeer. Rather, when a peer node needs to send a message/data to another peer node, it creates a new connection to that node, transfers the message/data and then closes that connection.

Results: update size 100Kb, Ptarget size 0.5



Results: update size 1Mb, Ptarget size 0.5



Conclusions

- Provision of awareness in project-based group work is a crucial factor for the successful accomplishment of the project
- Mature proposals have been reported in the literature for web-based groupware systems, but there has been little work for P2P groupware systems
- We aim to address the awareness requirements of P2P groupware systems
- We have identified awareness mechanisms that are targeted at the middleware layer on top of which P2P groupware systems can be built
- Superpeers implement core services in order to support peergroup activity and awareness provisioning
- Peers implement lower-level services and also support direct communication among group members, reducing the communication burden on superpeers during the group's activities

Conclusions

- We envision the event-based services of superpeers and peers being composed in order to build more complex services, at varying levels of abstraction, in order to provide the required awareness functionality for a particular P2P groupware application
- We have implemented a prototype system providing the core peer and superpeer functionality, using Java, Apache Jena API, and Jena Triplestore
- We have undertaken a preliminary performance evaluation to investigate the scalability of our system, using the Amazon Web Services (AWS) Cloud Computing platform
- This evaluation points to the scalability of our approach, but more experimentation is under way