Adaptive Workflow Scheduling Under Resource Allocation Constraints and Network Dynamics

Artin Avanes
supervised by Johann-Christoph Freytag
Humboldt-Universität zu Berlin
Unter den Linden 6, 10099 Berlin
avanes@informatik.hu-berlin.de

ABSTRACT

Workflow concepts are well suited for scenarios where many distributed entities work collaboratively together to achieve a common goal. Today, workflows are mostly used as computerized model for business processes executed in instances in commercial Workflow Management Systems. However, there are many other application domains where computer-supported cooperative work can be captured and organized by workflows. In this paper, we investigate the task of scheduling workflows in self-organizing wireless networks for disaster scenarios. Most research work in the field of workflow scheduling has been driven by temporal and causality constraints. We present an adaptive scheduling algorithm that finds a suitable execution sequence for workflow activities by additionally considering resource allocation constraints and dynamic topology changes. Our approach utilizes a multi-stage distribution algorithm which we extend with techniques to cope with network dynamics.

1. INTRODUCTION

Systems that support collaborative work among many distributed entities, such as people, components, software services, machines are referred to as Groupware. In last years, Workflow Management Systems (WfMSs) as one specification of Groupware systems are increasingly used to coordinate and execute business processes. These business processes are described by workflow models constituting the temporal and causal sequence (order) of individual process tasks, the data flow between these tasks, and the user responsibility for a certain task [19]. The execution of business processes is facilitated by standardized and machine-processable business process languages, e.g. BPEL [23], that map single tasks to web services. The usage of WfMSs aims at standardizing processes, masking the heterogeneity of involved resources and improving the process quality thus reducing the processing-time and costs for the whole process. Traditionally, WfMSs have been designed for well-known, structured and primarily static processes executed in a stable wired infrastructure. While large backend servers for workflow execution oversee the whole workflow and delegate tasks to workflow clients (e.g. terminals), users can execute the tasks by invoking software services at these remote clients.

PVLDB '08, August 23-28, 2008, Auckland, New Zealand
Copyright 2008 VLDB Endowment, ACM 978-1-60558-306-8/08/08

However, workflows can be applied in other, more dynamic application scenarios beyond business computing. In recent years, the computing and resource capacities of small, embedded and wireless networking devices have matured sufficiently to enable their integration into novel application domains. Example for new application scenarios are wilderness exploration, military operations or disaster management that require strong collaboration among heterogeneous devices, such as sensor networks and hand-held devices. In this paper, we introduce an adaptive scheduling algorithm that considers the resource constraints and the dynamics of potential workflow participants during a disaster event.

Scheduling of workflows is defined as a problem of finding a correct execution sequence for workflow activities. A correct execution sequence obeys the constraints inherent in the workflow model, i.e. temporal and causality constraints. Considering workflow scheduling in more dynamic applications, e.g. during a disaster event, we have to cope with new challenges and more constraint classes: First, the topology of the underlying network is more dynamic. On the one hand, devices may fail caused by a disaster event. On the other hand, new devices must enter the network dynamically. Consider for example a scenario where many police and fire men equipped with hand-held devices enter the disaster area and have to be coordinated. Second, we must expect an unreliable communication infrastructure. IP-based or cellular networks will be temporarily unavailable due to broken cables, transmitters or overloading. Additionally, more message packets may be lost using wireless communication. Finally, small embedded devices, e.g. PDAs and sensor devices, have a finite energy capacity. The execution of a high number of workflow (disaster battling) tasks will result in shorter life cycle of these devices. Hence, workflow scheduling must consider resource and location constraints as further constraint classes.

Most research work in the area of workflow scheduling has been focused on the temporal and causality constraints [4, 14, 28, 9]. There are only few approaches that consider scheduling under resource allocation constraints [27, 3]. However, these algorithms assume a stable infrastructure and do not provide robustness against topology changes. To the best knowledge, this paper is the first that describes

a procedure that produces a workflow schedule considering resource allocation constraints and network dynamics. For this propose, our algorithm incorporates a multi-stage distribution procedure consisting of a logical partition and physical allocation step with self-adaptive recovery techniques.

The remainder of this paper is organized as follows. In Section 2 we discuss related work in the context of workflow scheduling. In Section 3 we present a concrete example within a disaster scenario to illustrate the challenges. Section 4 introduces the overall framework including the workflow model and the adaptive scheduling algorithm addressing the identified challenges. We summarize and conclude our paper with open research questions in Section 5.

2. STATE OF THE ART

Workflow scheduling belongs to the class of application scheduling algorithms where the requirements of an application primarily determine the scheduling behavior. Our work completes a number of research efforts in the area of workflow scheduling. To the best knowledge, there is no approach that combines both, resource constraints and possible network dynamics applicable for the use in a disaster scenario. In the following, we discuss related work and thereby, we distinguish between two areas: (1) workflow scheduling and (2) (application) scheduling algorithms in other fields of computer science.

Scheduling in Centralized WfMSs: In the business area, there are commercial and open-source WfMSs [17, 1]. All of them are based on a centralized workflow engine that maps tasks to corresponding web services. This mapping procedure, also referred to as *Orchestration* and usually enabled by BPEL process files, assumes a stable communication and strong workflow participants, e.g. a server with sufficient resource capabilities. Contrary, we focus on participants with limited resource capabilities and on dynamic topology changes due to failure, i.e. broken communication links.

Scheduling in Distributed WfMSs: In the past, a series of WfMSs with several, distributed workflow server have been developed. Some [19] address mobility of users and support disconnected clients during workflow execution. Others [7] distribute workflow tasks to users by minimizing the communication costs of the subnets. Again, even if these systems consider disconnections and communication costs, they still rely on a fixed infrastructure and are not applicable for our scenario. Since disconnections are temporary, rescheduling of workflow tasks is not supported.

Scheduling in Agent-based WfMSs: In agent-based WfMSs, execution of workflow tasks is controlled by agents that react on certain events and conditions [20, 18]. Even if [20] support preventive as well as reactive workflow adaptation, they pay little attention to workflow scheduling under resource allocation constraints. Additionally, they do not consider dynamic topology changes while the running workflow instance is still the same.

Scheduling in MANETs: Scheduling of workflows in **m**obile **a**d-hoc **net**works (MANET) are another related research area. Here, nodes of a MANET cooperate with each other without relying on a fixed infrastructure. In [6], workflow applications are modeled by task graphs which are embedded onto a MANET in order to discover appropriate devices in the network. If disconnections between mobile devices are detected, a re-instantiation of the task-graph is conducted. Our work can be seen as extension to [6] since we also consider device capabilities that change with time (e.g. power consumption). The consideration of these dynamic device attributes is crucial with increasing number of workflow instances that may slow down the execution rate.

WfMSs using Constraint-Programming: An interesting approach can be found in [27] which deals with scheduling under resource allocation constraints. It uses constraint logic programming (CLP) integrating with Concurrent Transaction Logic (CTR) thus introducing a new logical formalism. Our work can be seen as an extension to [27] where a stable and reliable communication infrastructure is assumed. In addition, we propose strategies to cope with network dynamics and integrate them into the constraint solving process. We will also show how to schedule several concurrent workflow partitions of one workflow instance.

There are further scheduling methods used in other fields of computer science. In the area of **grid computing**, there are several adaptive scheduling algorithms [25, 8]. However, the communication effort for these scheduling algorithms is very high and therefore not applicable for our disaster scenario. Another related field to workflow scheduling is **job-shop scheduling** [10, 30, 15] where Operation Research (OR) techniques are used and incorporated with CLP. However, workflows are more complex than a job-shop and in addition, the algorithms fail in dealing with dynamic topology changes. Finally, **planning in AI** [21, 22] includes strategies that can be used for workflow scheduling. Again most research work in this area does not focus on workflow scheduling under resource constraints and do only support dynamic changes at the workflow level ignoring possible topology changes.

3. A MOTIVATING EXAMPLE

3.1 Workflows in Disaster Events

Disaster is a broad term including varying temporal and spatial dimensions as well as social and economic consequences. In this paper, we refer to a definition given in [11] where a *disaster* is defined as a

> "'serious disruption of the functioning of a community or a society causing widespread human, material, economic or environmental losses which exceed the ability of the affected community or society to cope using its own resources."

Following this definition, we subsume all activities connected with a disaster as so-called *disaster management activities*. In the area of disaster management we distinguish between three major time periods: (a) *preparation*, (b) *response*, and (c) *recovery*. In each of these periods a set of disaster management activities is executed by selected authorities in a specific order. Consequently, we distinguish between *preparation*-, *response*-, and *recovery* workflows (processes) deployed and executed within the introduced disaster life cycle phases.

The preparation period contains preventive measures including planning, mitigation and preparation activities. During the planning phase, possible disaster scenarios and their social and economic impacts are evaluated. Results of this phase are utilized to develop individual tasks and emergency plans thus reducing the impact of a disaster event

and saving human lives. The response period aims to provide assistance for victims shortly after the disaster occurs. Rescue teams first analyze the situation at the disaster location and decide then which assistance has to be conducted in which order. In particular, they aim for stabilization thus avoiding secondary damages, i.e. fire caused by broken gas pipes or collapsing of bridges. First speed recovery measures are enforced by the rescue teams to provide a minimal operating standard for medicine treatment, food supply and other essential saving tasks. Finally, long-term

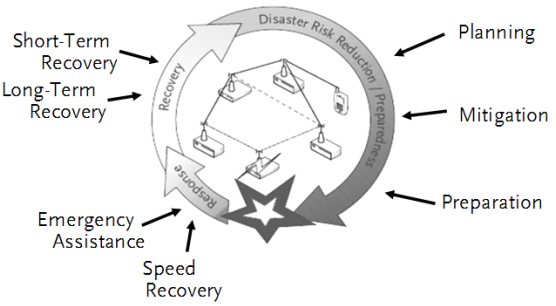

Figure 1: The Life Cycle of Disaster Management

recovery measures during the recovery phase complete the disaster management activities. In contrast to the speed recovery activities, decisions are not taken in real-time; they focus on re-installing the infrastructure (buildings, cellular-networks etc.). In this paper, we focus on scheduling of assistance and speed-recovery activities which are part of response workflows as defined above. Hence, many of the response activities cannot be foreseen and therefore planned in advance. They are dynamically composed after the disaster event when the rescue teams know more details about the magnitude and impact of a disaster. However, the results of our research may be also leveraged in the preparation phase of a disaster cycle to enable a more efficient coordination and a robust execution of emergency activities among different rescue organizations.

3.2 A Scenario

For the following example we assume that each rescue team member is equipped with particular devices offering wireless communication, sensing, image processing and further computational capabilities. In our scenario we focus on scheduling of disaster management activities after the disaster occurs. We will use this example throughout the paper.

Consider a city shortly after an earthquake or tsunami when major parts of a city's infrastructure is destroyed. A crisis squad decides to send out the rescue teams equipped with laptops and PDAs to analyze and to document the damages at the disaster location. Based on the information gathered, the groups start with first assistance and speed recovery activities. There can be several sub-workflows within the whole response workflow. One group may be in charge of collecting more up-to-date map information about the location to refresh possible outdated map data or to construct a map of an unknown area. Within another group some team member may take photos or measure temperature and the dissemination of a fire in a building. They may send this information to other members of the team who are ready to enter the building. Such information prepares them for appropriate reactions. An overview about one possible (response) workflow is shown in Figure 2. Resource allocation constraints in such response workflow can be:

1. Critical activities should be scheduled to team members with sufficient capacity and capabilities.
2. Two consecutive and interdependent activities should be scheduled to team members who are within (wireless) communication range.
3. The number of activities assigned to one rescue team should not exceed a certain threshold.

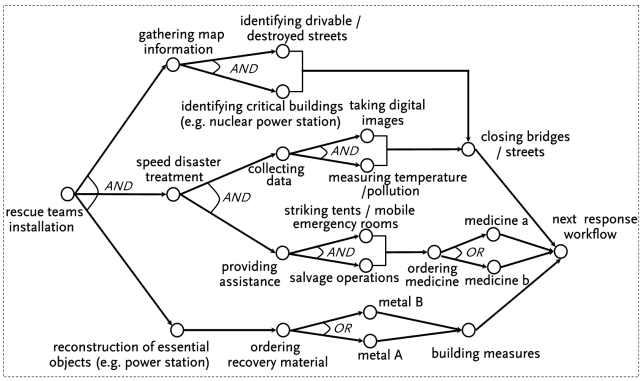

Figure 2: Example of a Disaster Workflow

Dynamic topology changes may appear during the scheduling and execution of workflow activities. In a disaster scenario, it is very likely that communication between team members may be interrupted temporally or team members may fail during the operation leaving activities uncompleted. Hence, *re-scheduling* of workflow activities at run-time is essential to allow the continuation of workflows even in a failure case.

4. OUR CONTRIBUTION

We mainly focus on following three research challenges: (a) designing a suitable workflow and coordination model for disaster events, (b) designing a model for distributing workflow activities and (c) designing algorithms to perform recovery for workflow activities.

4.1 The Workflow and Network Model

Workflows are conventionally seen as a collection of activities executed in a specific temporal/causal sequence order. In this context, we distinguish between a *global workflow set* S and *local workflow schedules* L_i. While the workflow set contains several workflows P_i, the local schedules determine the concurrent execution of partial workflows from the set S. Thereby, one local schedule may contain activities from different workflows $P_i \in S$ (see Figure 3). Activities are atomic, by definition. Each activity is performed either in an automated manner by a computer system or by humans, e.g. rescue team members. We assume that each activity returns at least a value indicating whether it succeeded or failed. We distinguish between *sequential* and *parallel* activities. *AND-branches* indicate workflow activities which are to be executed in parallel. All parallel activities within an AND-branch must be finished before the next activity can be

executed. For instance, bridges can be only closed after map information (where and how many affected bridges exist in an area) and a first estimation about the possible damages of these bridges are collected. Contrary, *OR-branches* present paths with alternative activities where only one path will be selected for the execution. For instance, different recovery material can be used depending on the availability of resources (e.g. transport machines), the emerging costs, or on the individual skills of the rescue members.

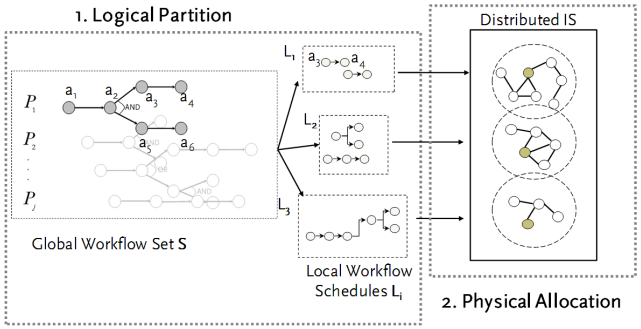

Figure 3: Overview about Scheduling Procedure

So far, our workflow model is not fundamentally different from other graph-based workflow models, e.g. as in [26, 29]. However, we will extend this workflow model by adding resource allocation constraints and considering network dynamics.

Network Model. During a disaster, there is always a hierarchical execution of emergency processes. Local headquarters determine the individual activities that are executed by the rescue forces afterwards. Thus, we assume a distributed and heterogeneous execution environment where workflow activities are performed either by fixed, static stations, e.g. measuring temperature by sensor devices or by mobile workflow participants (e.g. rescue teams). In general, workflow participants are able to accomplish the activities by accessing software services, files, databases running on these devices. Mobile workflow participants are clustered into groups and cooperate within these groups using wireless communication. Each group has a group leader responsible for coordinating the activities within its group. The selection process of a group leader is not in our scope; its existence is assumed.

4.2 Scheduling of Workflow Activities

The distribution of emergency processes short after a disaster is crucial for a successful disaster management (see Katrina in 2005). Hence, we are interested in an efficient distribution of corresponding workflow activities that enhances the life span of the system (including people and devices) on the one hand and that supports inter-process concurrency on the other hand. Based on the workflow and network model as given above, we first define the scheduling procedure:

Definition 1. (Workflow Scheduling). Workflow scheduling is a mapping function that assigns a given workflow set $S = \{P_1, P_2, ..., P_n\}$ to a set of distributed workflow participants. A valid mapping solution must satisfy the resource allocation constraints and consider possible network dynamics.

Our idea is to divide the scheduling process into 2 phases: (a) *a logical partition step* and (b) a *physical allocation step*. The key motivation behind this multi-stage procedure is to reduce the complexity of our scheduling problem. Apart from multiple resource constraints that must be considered, workflow activities often interdepend by control and data flows. In a large-scale disaster, e.g. an earthquake, we expect thousands of rescue members and workflow activities respectively to be involved as well as a similar number of heterogeneous devices. Existing scheduling procedures as mentioned above are not appropriate for such a large-scale scenario.

1.Step: Logical Partition. In a first step, the workflow set S is divided into a set of partitions $L = \{L_1, L_2, ..., L_i\}$ which will be distributed as *local schedules* to the network groups afterwards. During the response phase of a disaster event, it is very likely that there are alternatives for an execution of an activity, e.g. either by a rescue member X or Y. Thus, by assuming the presence of alternative executions, the goal of this first step is to find "good" partitions or local schedules respectively. To determine such partitions, we suggest to use *affinity matrices* to calculate the similarity of workflow activities. The key idea is to group "similar" activities together that may be executed by the same/similar workflow participants. For this purpose, we create *workflow attribute tables* which contain *qualitative* (database and application information) and *quantitative* information (network and computer system information) for each activity. While the former is used during the logical partition, the latter is needed during the physical allocation step. Database information may contain the data flow between different workflow activities, whereas application information may determine the location (where) or the person (who) that executes the activity. Contrary, quantitative information provides information about the physical requirements of an activity, e.g. the minimum required energy capacity, network bandwidth or the needed operating platform. An example of a workflow attribute table can be found in Figure 4.

	Loc	Authority	Pre	Post	Cons	Dur
a_1	‚Build A, B'	‚Tom',Jerry'	Null	a_2	23	30
a_2	‚Build B'	‚John',Jerry'	a_1	a_3	2	10
a_3	‚HQ F'	‚Server Y'	a_2	a_4	5.5	21
	...	...	...	...	...	...
a_n	‚Area C'	‚Sensor Device'	a_{n-1}	Null	0.8	0.4

$$\begin{array}{c} \\ a_1 \\ a_2 \\ a_3 \\ \\ a_n \end{array} \begin{array}{c} \begin{array}{ccccc} a_1 & a_2 & a_3 & & a_n \end{array} \\ \begin{pmatrix} 1 & 0.4 & 0.8 & \dots & 0.1 \\ 0.4 & 1 & 0.7 & \dots & \dots \\ 0.8 & 0.7 & 1 & \dots & \dots \\ \dots & \dots & \dots & 1 & \dots \\ 0.1 & \dots & \dots & \dots & 1 \end{pmatrix} \end{array}$$

Figure 4: Translating Workflow Attribute Table into Affinity Matrix

In the following, we rather present a general algorithm template to determine an affinity matrix for one workflow $P_i \in S$ than a detailed elaboration of each step since this will be part of our future thesis work. So far, we have identified three important steps:

(a) Similarity Calculation: We calculate a similarity value for each activity pair (a_i, a_k) of a workflow P_j based on attribute vales (of a qualitative attribute) contained in the workflow attribute table.

Definition 2. (Multiple Values). Given a workflow attribute table $T_P(A_1, A_2, ..., A_n)$ where A_i is an attribute defined over a domain D_i. Then an attribute A_i may take multiple values $v_1, v_2, ..., v_j \in D_i$ for one specific activity tuple $a_i \in T$, i.e. value v_1 *OR* v_2 *OR...OR* v_j.

Consider the workflow attribute table given in Figure 4: For the activity a_1, the location attribute may either take the value $a_1.loc =$ "$BuildingA$" or $a_1.loc =$ "$BuildingB$". In such a case, the selection of an eligible value may influence the partition affiliation of the corresponding activity.

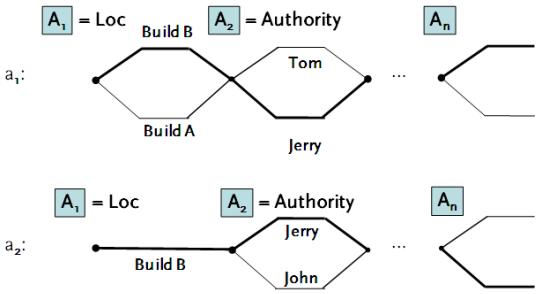

Figure 5: Comparing Paths of Attribute Values

In general, we purpose two different heuristics to deal with multiple values for one attribute (in one activity tuple). The first strategy is to choose for one attribute A_i of one activity tuple a_i the value v_k from the set of possible values $\{v_1, v_2, ..., v_j\}$ that leads to a higher similarity with other activity tuples a_j. Hence, for a given tuple pair $(a_i, a_j) \in T_P$, the algorithm must compare possible *paths of attribute values* for a_i and a_j and select the two paths that have the most similarity. In the second strategy, the opposed procedure is pursued where the value v_l from the set of possible values $\{v_1, v_2, ..., v_j\}$ is chosen that leads to the most dissimilarity with other activity tuples. While the former behavior will result in a fewer number of partitions with many partition elements, the latter supports inter-process concurrency by generating more partitions with fewer partition elements.

(b) Derived Partition: Based on the similarity values calculated for all activities of one workflow P_i, we propose an algorithm that derives partitions from each column of the affinity matrix. The goal of this step is to incrementally traverse the columns to first identify neighbors of the corresponding activities. A neighbor activity a_j is an activity that has a high similarity to the compared activity a_i. Based on the neighborhood of activities we will group similar activities to a partition., i.e. activities that have a common neighborhood [16]. However, there may be still alternatives into which partition an activity may be inserted. In such cases, we will develop several strategies that also consider other dependencies between activities, such as the control and data flow.

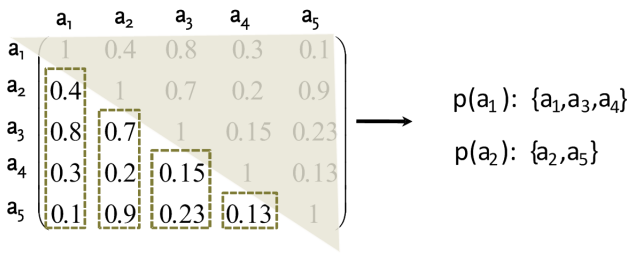

Figure 6: Derived Partition

(c) Merging of Partitions: Once we have derived the partitions for one workflow, we repeat steps (a) and (b) for all given workflows in the global workflow set S. Finally, we merge partitions derived from different workflows. Again, application information, e.g. the location, may be utilized to merge "similar" partitions to local schedules together.

2.Step: Physical Allocation: The physical assignment of partitions to the distributed workflow participants is the task of the physical allocation step. Even if the logical partition step leverages knowledge about the given physical attributes, e.g. $a_n.Authority =$ "$SensorDevice$" indicates that this activity must be executed by a sensor device, a physical assignment is required to select suitable devices or workflow participants respectively from the set of possible candidates. To continue the example, sensor device SD_1 or SD_2 may be candidate devices to execute the corresponding activity.

The physical allocation consists of two sub-steps: (a) assigning partitions (aka local schedules) to groups and (b) assigning activities of the local schedules within the selected groups. While the former can be determined by using application and network information, we suggest to use *constraint-programming* (CP) techniques for the latter. In general, any suitable allocation model is possible to assign atomic activities to workflow participants. However, there are two major reason for using CP: First, CP has proved its feasibility in solving of several real-world scheduling and planning problems [13, 12]. Second, we assume a highly distributed environment where multiple groups of rescue forces execute different emergency activities. CP allows us to model each group G_i as an individual constraint system with corresponding constraints that need to be considered during the allocation step.

Definition 3. (Constraint System ζ_{G_i}). A constraint system, ζ_{G_i}, is defined as a system that gets as input **group variables** $V_{G_i} = \{v_1, ..., v_m\}$ as the set of all workflow participants within G_i, the **group domain** as the set of all assigned partitions (local schedule of G_i) and finally a set of resource allocation **constraints** C_r.

The key aspect of CP [5] is to use constraints to remove infeasible values from the domains of the given variables thus pruning the search space. However, the calculation of a constraint solution often remains a complex and time-consuming task. During a disaster, many activities must be assigned to many workflow participants within strict time constraints.

By using distributed constraint systems that calculate their solutions locally, our approach reduces the complexity of CP and introduces a distributed way of solving a complex CP problem. Further on, we are able to define an individ-

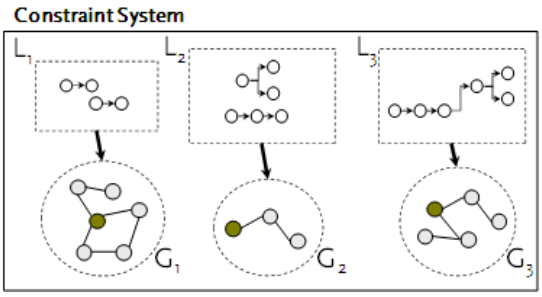

Figure 7: Distributed Constraint Solving

ual *objective function* for each group separately extending CP to a *constraint satisfaction problem* (CSP). Our goal is to approximate a (global) optimal assignment solution by solving individual CSPs for each corresponding group.

Future work will include the determination of suitable objective functions that can be used in our application scenario. In addition, it will be important to design an allocation model dependent from physical requirements of the workflow activities, e.g. the consumption or the duration, on the one hand and current network information, e.g. current energy capacity or network bandwidth, on the other hand.

4.3 Extension for Handling Changes

Changes can dynamically appear on the workflow level as well as on the network level. Focusing on the latter case, we distinguish between different dynamic events that can trigger re-scheduling. We classify failure events into *communication failures* and *(network) node failures*. Communication links may be broken preventing the communication between team members or resulting in a network partitioning in the worst case. Additionally, rescue members may fail during the execution leaving activities uncompleted. Topology changes may also appear due to new entering rescue members. For each of these events, we aim to provide strategies to re-schedule activities.

For re-scheduling, we distinguish between so-called *retriable* and *compensation activities*. An activity is retriable if it has a successfully execution even in the case of temporary (network) failures. Thus, an alternative execution will always exist for each retriable activity. In our future research work, we will pursue a strategy that first re-schedule retriable activities locally. We will retrieve an alternative execution path within the local schedules, whereas the global schedule will not be significantly affected. However, there will be cases where a local re-organisation is not possible within a specific local schedule. Hence, as a second we need an adaptive algorithm that allows a balanced re-scheduling among local schedules thus considering the limited resource capabilities of each local group on the one hand and the given control and data flow on the other hand.

In summary, our goal is to develop different levels of *recoverability* for workflow (process) schedules that are concurrently executed in a distributed environment. Thereby, the level of recoverability is determined by the possibility of a local re-scheduling. The major difference to the initial scheduling is the additional states which activities may have. In contrast to the initial scheduling, *committed* as well as *running* activities may exist at the time of re-scheduling thus exacerbating the selection of alternative execution paths.

If there is no alternative given for a failed activity, compensation of this activity is required. In the worst case, previous executed activities may also be involved in the compensation and need to be rollbacked. In such a case, our re-scheduling algorithm has to decide how far compensation must be conducted among the distributed local schedules. Here, our goal is to develop different levels of *compensation* for workflow schedules. In contrast to the recoverability issue, the level of compensation is determined by the number of activities that need to be rollbacked.

5. CONCLUSION AND OUTLOOK

This paper presents ideas about scheduling for workflows in disaster scenarios under consideration of two major research challenges: resource allocation constraints and possible network dynamics. Based on an initial workflow and network model, we propose an multi-stage algorithm that consists of two steps to assign activities to nodes: a logical partition step that identifies partitions of workflow activities based on similar attribute values and a physical allocation step that assigns these partitions by using constraint-programming. The consideration of network changes completes our research work and contributes to a robust and adaptive scheduling procedure.

The evaluation of our research work will be based on real-world emergency processes. Currently, we are working together with domain experts from the GeoForschungsZentrum Potsdam (GFZ), the national research centre for Geosciences in Germany. Future collaboration with fire and police departments will give a deeper understanding of emergency processes. Based on these concrete processes, we will first evaluate our distribution algorithm regarding an "optimal" assignment solution. Thereby, the assignment solutions locally calculated within the groups will be compared with a global assignment solution. As a second, we will simulate several failure events to test the robustness of our re-scheduling algorithm. We will also evaluate our approach regarding the generality and applicability for other domains beyond the specific disaster scenario.

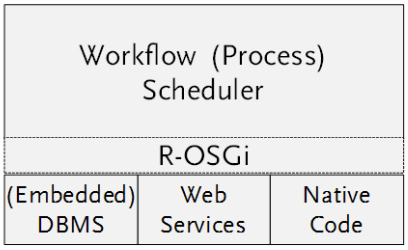

Figure 8: Technical Overview

Technically, we focus on the usage of an OSGi-based prototype to implement our workflow scheduler that may run on both, server and group leader node. The OSGi-Alliance has specified a Java-based middleware platform that provides a service-oriented, component-based environment for application development [2]. Recently, the distributed variant called *R-OSGi* has been developed [24] that can be used to enable the distributed deployment and coordination of our emergency processes among distributed network groups. Thereby, our workflow scheduler may act on top of embedded databases or as a stand-alone execution component of software services.

Acknowledgments.

This research is supported by the German Research Society (DFG grant no. NA 1324). Thanks to my thesis advisor Johann-Christoph Freytag for helpful discussions.

6. REFERENCES

[1] ActiveBPEL Engine. http://www.activebpel.org/.
[2] The OSGi.Alliance. http://www.osgi.org/.
[3] G. Alonso, D. Agrawal, A. E. Abbadi, M. Kamath, R. Günthör, and C. Mohan. Advanced Transaction Models in Workflow Contexts. In *ICDE*, pages 574–581, 1996.
[4] P. C. Attie, M. P. Singh, A. P. Sheth, and M. Rusinkiewicz. Specifying and Enforcing Intertask Dependencies. In *VLDB*, pages 134–145, 1993.
[5] R. Bartak. Constraint Programming: In Pursuit of the Holy Grail. In WDS, 1999.
[6] P. Basu, W. Ke, and T. D. C. Little. Dynamic Task-Based Anycasting in Mobile Ad Hoc Networks. *Mob. Netw. Appl.*, 8(5):593–612, 2003.
[7] T. Bauer and P. Dadam. Efficient Distributed Workflow Management Based on Variable Server Assignments. In *CAiSE*, pages 94–109, 2000.
[8] F. Berman, R. Wolski, H. Casanova, and et al. Adaptive Computing on the Grid Using AppLeS. *IEEE Transactions on Parallel and Distributed Systems*, 14(4), 2003.
[9] A. J. Bonner. Workflow, Transactions and Datalog. In *PODS*, pages 294–305, 1999.
[10] Y. Caseau and F. Laburthe. Improved CLP Scheduling with Task Intervals. In *ICLP*, pages 369–383, 1994.
[11] T. Catarci, S.Dustdar, and et al. Workpad: 2-Layered Peer-to-Peer for Emergency Management through Adaptive Processes. *ColCom*, 0:43, 2006.
[12] H. K. W. G. Chan, P. Nurse Scheduling with Global Constraints in Chip: GYMNASTE. In PACT, 1998.
[13] P. M. Chow, K.P. Airport Counter Allocation using Constraint Logic Programming. In PACT, 1997.
[14] H. Davulcu, M. Kifer, C. R. Ramakrishnan, and I. V. Ramakrishnan. Logic Based Modeling and Analysis of Workflows. In *PODS*, pages 25–33, 1998.
[15] H. Goltz and U. John. Methods for Solving Practical Problems of Job-Shop Scheduling Modelled in CLP. In PACT, 1996.
[16] S. Guha, R. Rastogi, and K. Shim. Rock: A Robust Clustering Algorithm for Categorical Attributes. In *ICDE*, pages 512–521, 1999.
[17] IBM. Websphere Process Server. http://www-306.ibm.com/software/integration/wps/.
[18] N. R. Jennings, T. J. Norman, and P. Faratin. Adept: An Agent-Based Approach to Business Process Management. *SIGMOD Record*, 27(4):32–39, 1998.
[19] C. Mohan, G. Alonso, R. Gunthor, and M. Kamath. Exotica: A Research Perspective on Workflow Management Systems. *Data Engineering Bulletin*, 18(1):19–26, 1995.
[20] R. Müller and E. Rahm. Dealing with Logical Failures for Collaborating Workflows. In *CoopIS*, pages 210–223, 2000.
[21] K. Myers and P. Berry. At the Boundary of Workflow and Ai. In AAAI, 1999.
[22] A. Nareyek. Applying Local Search to Structural Constraint Satisfaction. In Workshop on Intelligent Workflow and Process Management: The New Frontier for AI in Business, 1999.
[23] OASIS. Web Services Business Process Execution Language. http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf.
[24] J. S. Rellermeyer, G. Alonso, and T. Roscoe. R-OSGi: Distributed Applications Through Software Modularization. In *Middleware*, pages 1–20, 2007.
[25] T. Röblitz and A. Reinefeld. Co-Reservation with the Concept of Virtual Resources. In *CCGRID*, pages 398–406, 2005.
[26] H. Schuldt, G. Alonso, C. Beeri, and H.-J. Schek. Atomicity and Isolation for Transactional Processes. *ACM Trans. Database Syst.*, 27(1):63–116, 2002.
[27] P. Senkul, M. Kifer, and I. H. Toroslu. A Logical Framework for Scheduling Workflows Under Resource Allocation Constraints. In *VLDB*, pages 694–705, 2002.
[28] M. P. Singh. Semantical Considerations on Workflows: An Algebra for Intertask Dependencies. In *Workshop on Database Programming Languages*, page 5, 1995.
[29] C. Türker, K. Haller, C. Schuler, and H.-J. Schek. How can we support Grid Transactions? Towards Peer-to-Peer Transaction Processing. In *CIDR*, pages 174–185, 2005.
[30] J. Wuertz. Oz Scheduler: A Workbench for Scheduling Problems. In *ICTAI*, pages 149–156, 1996.