Modelling and Analyzing the Authorization and Execution of Video Workflows

Ligang He[1], Chenlin Huang[2], Kenli Li[3], Hao Chen[3], Jianhua Sun[3], Bo Gao[1], Kewei Duan[4] and Stephen A. Jarvis[1]
1. Department of Computer Science, University of Warwick, Coventry, UK
2. Institute of Software, School of Computer Science, National University of Defense Technology, Changsha, China
3. School of Computer and Communication, Hunan University, Changsha, China
4. Department of Computer Science, University of Bath, Bath, UK
liganghe@dcs.warwick.ac.uk

Abstract— **It is becoming common practice to migrate signal-based video workflows to IT-based Video workflows. Video workflows have some inherent features, including: 1) necessary human involvements in video workflows introduce security and authorization concerns; 2) the frequent change of video workflow contexts requires a flexible approach to acquiring performance data; 3) the content-centric nature of video workflows, which is in contrast to the business-centric of business workflows, requires the support of scheduled activities. This paper takes the above issues into account, proposing a novel mechanism for modeling video workflow executions in cluster-based resource pools under Role-Based Authorization Control (RBAC) schemes. The Color Timed Petri-Net (CTPN) formalism is applied to construct the models. Various types of authorization constraint are modeled in this paper, and scheduled activities are also supported in the model. There is a clear interface between workflow execution and workflow authorization modules. The constructed models are then simulated and analyzed to obtain performance data, including authorization overhead, system- and application-oriented performance. Based on the model analysis, this paper further proposes the methods to improve performance in the presence of authorization policies. This work can be used to plan system capacity subject to the authorization control, and can also be used to tune performance by changing the scheduling strategy and resource capacity when it is not possible to adjust the authorization policies.**

Keywords: Modeling, Authorization, Workflow, RBAC

I. INTRODUCTION

A video production workflow refers to all of the steps required to create, acquire, modify and deploy video media and supporting metadata [16]. Over the past few years, the video industry has been gradually migrating the supporting platforms of video workflows from signal-based infrastructure (i.e., using tape and coaxial cable as the video medium) to IT-based infrastructure (i.e., using computer servers, file and Ethernet networks as the medium). The transformation is enabled by the fact that many video processing software (e.g., to perform up-conversion, live graphics insertion, etc) has been developed and gradually replaced their specialized hardware equivalents.

Typically, video companies invest in purchasing supporting platforms. The advent of Cloud computing offers the companies the option of leasing the resources from Cloud service providers. No matter which option is adopted, it is important to investigate performance so that platform capacity can be planned and the desired production efficiency provided without purchasing or renting excessive resources.

The following inherent features of video workflows complicate the issue of performance investigation.

First, while business and scientific workflow management tries to automate the workflow execution, IT-based video workflows are not intended to completely replace humans in the video lifecycle, since human interactions are still required for decision making and artistic choices (e.g, video editing decisions) [13]. Human involvement in IT-based video workflows introduces security and authorization concerns. Role-based Authorization control, in which the users are assigned to certain roles, and the roles are associated with permissions, is a popular scheme in such a scenario [14][15]. For example, a video company employs a cluster to host the applications required in the process of video production. A particular application (in a video workflow) hosted in the cluster can only be invoked by an employee (user) with certain job positions (roles) in the company. The following authorization constraints are often encountered in such scenarios [14]: 1) Role constraints: An application may only be run by assuming a particular set of roles; 2) Temporal constraints: A role or a user is only activated during certain time intervals (e.g., an video editor's working hours); 3) Cardinality constraints: The maximum number of applications running at the same time under a role is *N*; 4) Separation of Duty constraints: If Task *A* is run by a role (or a user), then Task *B* must not be run by the same role (or user); 5) Binding of Duty constraints: If Task *A* is run by a role (or user), then Task *B* must be run by the same role (or

978-1-4577-1950-9/11/$26.00 ©2011 IEEE

user). All these authorization constraints may affect the execution behaviors of applications and have impact on both application and system performance (e.g. mean response time of workflows, utilization of the resource pool, etc).

Second, video workflows tend to change frequently due to various reasons, such as new advertising customer requests, new distribution channels, changing human interaction patterns and so on. Therefore, a flexible approach is desired so that when video workflow settings are changed, new performance data and consequently appropriate platform capacity can be conveniently recomputed. This feature of flexibility is especially important if the video company is leasing resources from the elastic Cloud services.

Finally, video workflows are typically *content-centric* processes, which mean that the processes start execution when content becomes available. This is in contrast with *business-centric* processes in business or scientific workflows, in which the workflow execution is typically triggered by a user's request for business or scientific processing. The content-centric feature requires the support of managing scheduled video operations (i.e., video operations are performed at scheduled time points, such as scheduled capture and playback).

This paper addresses these issues and develops a mechanism to model the execution of video workflows under Role-based Authorization Control in Cluster-based Resource pools, where the video processing applications are deployed and accessible to the employees of a video company. By analyzing and simulating the constructed models, we will be able to investigate system- and application-oriented performance (such as throughput, response time) when running a workflow of video processing software subject to authorization policies on computing platforms.

The Color Timed Petri-Net (CTPN) formalism [5][9] is applied to construct the models, which can then be analyzed and/or simulated to obtain system and/or application performance. Various types of authorization constraints are modeled in this paper, including role constraints, temporal constraints, cardinality constraints, Binding of Duty (BoD) and Separation of Duty (SoD) constraints. Various performance metrics can be analysed and obtained from the constructed models, including system-oriented performance (e.g., utilisation and throughput) and application-oriented performance (e.g., response time of workflows), which can be used as the basis to make decisions for platform capacity. In this paper, the model is constructed in a modular fashion, and individual authorization models can be assembled to form the authorization model for the entire system. Further, there is a clear interface between workflow execution and workflow authorization models. Therefore, if there are changes in workflow execution and authorization settings, the model can be quickly adjusted to recompute performance data. In the model, a task in a workflow is associated with a timestamp, which represents the earliest time when the task can start. We set the timestamp properly to support scheduled activities.

The work presented in this paper can be used to provide insight into how to tune performance by adjusting authorization policies on human interactions and/or the scheduling strategy so as to achieve balance between performance and security overhead. When it is not possible to adjust the authorization policies, this work can be used to plan system capacity or design the scheduling strategy so as to achieve the desired performance.

The remainder of this paper is organized as follows: Section II discusses related work; Section III introduces the Color Timed Petri-Net formalism applied in this paper; workflow authorization and execution is modeled in Section IV; model simulations and analysis are provided in Section V and, Section VI presents the simulation results for representative workflows. Section VII concludes the paper.

II. RELATED WORK

In [14], a security and authorization framework is proposed to protect the acessing of video images and applications. The framework adopts the RBAC authorization and formalises various authorization constraints such as temporal constraints and duty constraints. However, the work only focused on formalising and implementing authorization policies. It did not investigate the impact of authorization constraints on application performance.

Workflow management has been extensively studied and as a result is well documented in related literature [1][3][7][8]. Some of these researchers have also utilized Petri-nets to model workflow execution, however we note that their work does not formally investigate the performance of workflow execution under authorization constraints.

The work in [2] presents a formal model of human-aided workflows using CSP process algebra. The work modeled cardinality constraints, separation of duty and binding of duty constraints. However, its objective is to capture human behaviors and consequent authorization considerations in workflow formation, and cannot be used to investigate their impact on workflow executions.

Research has also been conducted on the topic of security and authorization constraints in workflow execution [1][4][6][10]. Some of that research also uses Petri-nets to model authorization constraints. The work presented in this paper differs from that research in the following respects: First, those models were geared towards business or scientific workflows. The model construction process is not automated, and the models do not support scheduled activities. Second, the work found in [1][10] does not capture the temporal constraints of the roles' availability. Third, it is assumed in [1][4] that a task can only be run under one role. This assumption simplifies the modeling process. However, the assumption is not always true. It may well be the case that a task is allowed to run under a range

of roles. The relaxation of this assumption is especially necessary when temporal constraints of roles' availability are taken into account. In so doing, when one role is not available, another activated role may be assigned to run the task, so that the workflow execution can still progress. In this work, the task-role assignments and the task-user assignments are modeled in a flexible fashion, which allows a task to be run under a selection of roles/users. Finally, the work in [2][11] does not capture the interactions between workflow authorization and workflow execution. The work presented in this paper models resource competition and interactions between the authorization module and the execution module.

III. COLOR TIMED PETRI-NETS

The formal definition of a Color Petri-Net (CPN) differs depending on the source literature [1][12]. The CPN formalism applied in this paper can be formally defined as Eq.(1) [12],

$$CPN = (P, T, I, O, CS, CF, A, G) \quad (1)$$

where P is a finite set of places; T is a finite set of transitions; I: $T \rightarrow P^{\infty}$ is the input function mapping from a transition to a multiset of places, which are termed *input places* of the transition; O: $T \rightarrow P^{\infty}$ is the output function mapping from a transition to a multiset of places, which are termed the transition's *output places*; CS is a set of colors[1] (a color in the CPN in [12] is represented as a data type, which can be a primitive data type or derived data type); CF: $P \rightarrow 2^{CP}$ is the color function mapping from places to a subset of the color set[2]; A: $(T \times P) \cup (P \times T) \rightarrow f(CF(P))$ is the arc function mapping a directed arc (from a place to a transition or from a transition to a place) to a function of the colors in the place P; G: $T \rightarrow f(\bigcup_{P \in (IN(T) \cup OUT(T))} CF(P))$ is a guard function mapping a transition to a function of the colors in all places associated with T.

The CPN model determines whether a token can be fired through the arc functions associated with the arcs (defined in A) and/or guard functions associated to transitions (defined in G). An arc function evaluates to a set of tokens, which determine the type and number of tokens that can pass through the arc. An arc function or a guard function can contain a number of variables as well as the operations (e.g. comparison) and logical operators (e.g. if-else branch) on these variables. Therefore, an arc function or a guard function may evaluate to different values for different tokens.

A Color *Timed* Petri-Net is an extension of a CPN. In a CTPN, a token can be associated with both a color and a timestamp. Furthermore, a CPN model has a global timer representing the model time. The timestamp attached to a token indicates the earliest time when the token can be processed. In addition to the arc and guard function, whether a transition can fire or not in a CTPN model, is also controlled by the model's global timer and the tokens' time stamps. The rule is that besides satisfying the arc and the guard functions associated to a transition, the tokens must have timestamps which are not later than the value of the global timer. The model remains at a given model time as long as there are enabled transitions. If there is not such an enabled transition, the global timer is advanced to the earliest model time at which at least one transition is enabled.

[1] A color in the CPN defined in [12] is represented as a data type, which can be a primitive data type (e.g., an integer) or a derived data type (e.g., a record derived from other data types).

[2] 2^{CP} is the power set of the color set.

IV. MODELLING WORKFLOW AUTHORIZATION AND EXECUTION

In this section, various types of workflow authorization constraints are modeled using CTPNs. These include: 1) Role constraints; 2) Temporal constraints; 3) Role and user assignment; 4) Binding-of-duty constraints; 5) Separation-of-duty constraints; 6) Cardinality constraints; This section also presents the method to automatically assemble individual authorization modulars, and models workflow executions under authorization control.

A. Role and user assignment subject to temporal constraints

A constituent task of a video workflow involves an application being launched by an employee (user) with a job position (role) in the video company. Generally speaking, a role and a user need to be assigned to invoke a particular application in the workflow. The model of assigning a role and a user to a task is illustrated in Figure 1. In this model, the transition T_{rt} assigns a role in the P_r place to a task in the P_t place, and deposits a token to the P_{rt} place, indicating the task-role assignment has been established. When a token is deposited into the place P_{urt}, it means that a user has been assigned to the task. We call the model of assigning a role and a user to a task as a *role and user assignment module*. It will be shown in Subsection 4.3 that these modules can be assembled to form the authorization control module for the entire workflow.

A Petri-net model can be formally defined using Eq.1. The remainder of this subsection is dedicated to determining the attributes CS, CF, A and G (ie., token colors, arc and Guard functions) for the role and user assignment module in Figure 1. These attributes together enforce the temporal constraints and role constraints. It is straightforward to determine the attributes P, T, I and O in the model. Therefore, they are omitted for brevity.

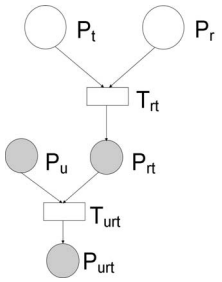


Figure 1. Role and user assignment module

1) Token colors

- A token in the P_r place, denoted as r, represents a role. The color of the token r is defined as $r = (rid, D)$, where rid is the role identifier and D is a set of durations in which the role is activated and can be assigned to tasks, i.e.,

$$D = \{[ld_i, ud_i] \mid i \in \aleph\}. \quad (2)$$

- A token in the P_t place, denoted as t, represents a task. The token color is defined as $t = (tid, wid, e)$, where tid is the task identifier, wid is the identifier of the workflow that the task belongs to, and e is the task's execution time. A timestamp ts is attached to the token t, which is symbolized as $t@ts$. The timestamp represents the earliest time at which the task can be processed, which is initialised as the task's earliest start time given the precedence constraints among tasks in the workflow. The timestamp can also be initialized as the time point scheduled to perform the task (subject to other authorization contraints and precedence constraints between). By doing so, the *scheduled activities* are supported.
- The color of a token in the P_{rt} place is defined as follows, which is the combination of the color attributes of token t and r.

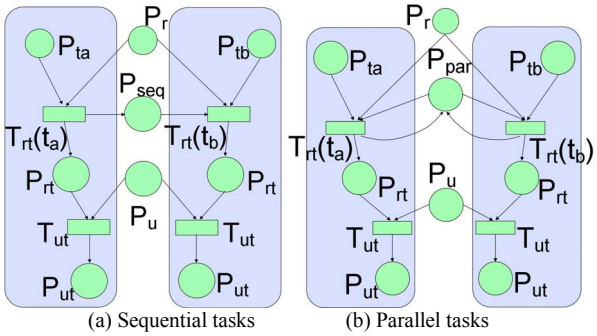
$$rt = (tid, wid, e, rid, D)$$

- A token in place P_u represents a user, whose color is defined as $u = (uid, rid)$, where rid is the role that the user belongs to.
- The color of a token in the place P_{urt} is defined as

$$urt = (uid, rid, tid, wid, e, D)$$

2) arc functions

- The arc function $A(P_r, T_{rt})$ can be defined as "If $gt \sqsubseteq r.D$ then r"; The expression means a role is allowed to be assigned to a task only when the value of the global timer is within one of the role's activation durations (denoted by the symbol "$\sqsubseteq$"), i.e., the role is available. It is assumed in this paper that the users have the same availability period as their associated roles. But it is straightforward to extend the model to allow individual users to have different and more restricted availability (e.g., in the case of different employees rotating to cover different shifts of the same official position).
- Other arc functions in the model are defined as $A(P_t, T_{rt})$ = "t", $A(P_u, T_{ut})$ = "u", $A(P_{rt}, T_{ut})$ = "rt" and $A(T_{ut}, P_{urt})$ = "urt"



(a) Sequential tasks (b) Parallel tasks

Figure 2. Duty constraints modules

3) Guard functions

The guard G associated to transition T_{rt}, denoted as $G(T_{rti})$, is used to enforce role assignment restrictions. There are two types of restriction. One is the role constraint which specifies the set of roles that are allowed to run a task. This restriction is expressed as "$r.rid \in R(t_i)$", where $R(t_i)$ denotes the set of roles which can run t_i. The other type of restrictions specifies that a role is assigned to a task only when the task can run to completion within one of the role's activation durations specified in Eq.2. This restriction is formulated as "$[gt, gt+t.e] \sqsubseteq r.D$". Therefore,

$$G(T_{rti}) = r.rid \in R(t_i) \;\&\&\; [gt, gt+t.e] \sqsubseteq r.D. \quad (3)$$

B. Binding of duty and separation of duty constraints

The duty constraints impose restrictions on the role assignments of two tasks in a workflow. Although Binding of Duty (BoD) and Separation of Duty (SoD) represent opposite authorization behaviors, these two types of duty constraints are modeled in a uniform fashion so that they have the same model structure. We call these the SoD module and the BoD module. The differences between these are essentially in some arc and guard functions, as well as the color set of some places. The benefit of doing this is that the models for individual duty constraints can be easily assembled to form the authorization model for the entire workflow.

There are two types of relationship between two tasks in a workflow in terms of precedence constraints: sequential tasks and parallel tasks. Assume task t_a and t_b. If $t_b \in$ Pred(t_a) or $t_b \in$ Succ(t_a) (Pred(t_i) and Succ(t_i) denote the set of tasks which are task t_i's predecessors and successors, respectively), then t_a and t_b are sequential tasks and have to be run in the required order. If $t_b \notin$ Pred(t_a) and $t_b \notin$ Succ(t_a), t_a and t_b can be executed in parallel. Duty constraints are modeled in a different way for these two types of tasks. Their model structures are shown in Figure 2a and Figure 2b, respectively.

1) Sequential tasks

As shown in Figure 2a, the duty constraints module consists of the role and user assignment modules for t_a and t_b (encompassed in two round-cornered rectangles), connected by place P_{seq}. P_{seq} is one of the output places of transition $T_{rt}(t_a)$ and one of the input places of transition $T_{rt}(t_b)$. The attributes of the role and user assignment modules have been discussed in Subsection A. The attributes related to the new place, P_{seq}, are as follows.

- **Token colors**: The color of a token in place P_{seq} is defined as $seq = (tid, wid, rid)$, which carries the information of which role has been assigned to task tid.
- **Arc functions**: $A(T_{rt}(t_a), P_{seq})$ and $A(P_{seq}, T_{rt}(t_b))$ are both defined as "seq".
- **Guard functions**: The guard functions associated to $T_{rt}(t_b)$ are different for SoD and BoD constraints. If it is a SoD constraint, $G(T_{rt}(t_b))$ is expressed as Eq.4. If it is a BoD

constraint, $G(T_{rt}(t_b))$ is formulated as Eq.5. t in Eq.4 and Eq.5 is a token in place P_{tb}. The condition *seq.wid=t.wid* is used to guarantee that the same workflow instance is referred to, since this model allows multiple instances of the same workflow to be processed simultaneously. Different instances of a workflow will have different values of *wid*.

$$seq.wid=t.wid \;\&\&\; seq.rid \neq r.rid. \quad (4)$$

$$seq.wid=t.wid \;\&\&\; seq.rid = r.rid. \quad (5)$$

2) Parallel tasks

Similar to Figure 2a, a new place, labeled P_{par}, is used in Figure 2b to interface between the role and user assignment module when t_a and t_b are parallel tasks. In the remainder of this subsection we first determine the attributes related to place P_{par}, and then use an example to illustrate the workings of the module.

- **Token colors**: There are two types of tokens in P_{par}: *par_init* and *par*. *par_init* is defined as *par_init* = (tid_a, tid_b, *wid*), while *par* is defined as *par* = (*tid*, *rid*, *wid*).
- **Guard functions**: If it is a SoD constraint, the guard function of the transition $T_{rt}(t_a)$ (or $T_{rt}(t_b)$) is formulated as Eq.6. If it is a BoD constraint, it is expressed as Eq.7.

$$((t.tid = par\_init.tid_a \,||\, t.tid = par\_init.tid_b) \;\&\&\; t.wid = par\_init.wid) \,||\, (t.wid = par.wid \;\&\&\; r.rid \neq par.rid) \quad (6)$$

$$((t.tid = par\_init.tid_a \,||\, t.tid = par\_init.tid_b) \;\&\&\; t.wid = par\_init.wid) \,||\, (t.wid = par.wid \;\&\&\; r.rid = par.rid) \quad (7)$$

- **Arc functions**: $A(P_{par}, T_{rt})$ and $A(T_{rt}, P_{par})$ are defined in Eq.8 and Eq.9, respectively.

$$A(P_{par}, T_{rt}) = par\_init \,||\, par \quad (8)$$

$$A(T_{rt}, P_{par}) = \text{“}\textbf{if } par\_init \textbf{ then } par\text{”} \quad (9)$$

3) Workings of the duty constraint modules

The workings of the duty constraint modules and the above expressions are illustrated as follows. When performing the role assignment for a task (e.g., t_a), the model will check whether the other task (e.g., t_b) has been assigned a role. Assume there is a BoD constraint between t_a and t_b, then the place P_{par} will contain a *par_init* token in the model's initial marking. When the T_{rt} transition performs the role assignment for t_a, it will evaluate which token in P_{par} can satisfy Eq.6, and therefore can be fired by the transition. There are two possibilities:

a) If there is a corresponding *par_init* token in P_{par}, which means that t_b has not been assigned a role, then the first part of Eq.6 (i.e., the portion before the second "||") will be evaluated as true. Consequently, the $T_{rt}(t_a)$ transition will remove the *par_init* token from P_{par} and deposit a *par* token back to P_{par} as shown in Eq.9.

b) If there is a *par* token in P_{par}, this means that t_b has been assigned to a role. Further, if there is a role in place P_r whose identifier (i.e., *r.rid*) is the same as the role assigned to t_b (i.e., *par.rid*), the second part of Eq.6 will be evaluated as true. Thus, the BoD constraint is enforced.

C. Modeling workflow execution under authorization

The authorization model for the entire workflow can be constructed by assembling a set of interacting hierarchical modules. There are clear interfaces and hierarchy structure among the modules. As shown in Figure 2, the duty constraints module consists of the role and user assignment modules for t_a and t_b, interfacing via place P_{par} or P_{seq}. In this work, it is modelled that the tokens in P_r and P_u are shared by all tasks. This is reasonable because the roles and users are the global parameters and should be applied to all tasks in the system. Cardinality constraints, which specify the maximum number of tasks that are run at the same time by a role (or a user), can be modeled by the number of tokens representing the role (or the user) in P_r (or P_u).

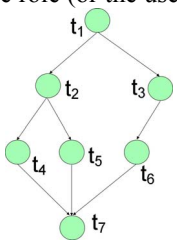


Figure 3. An exemplar workflow

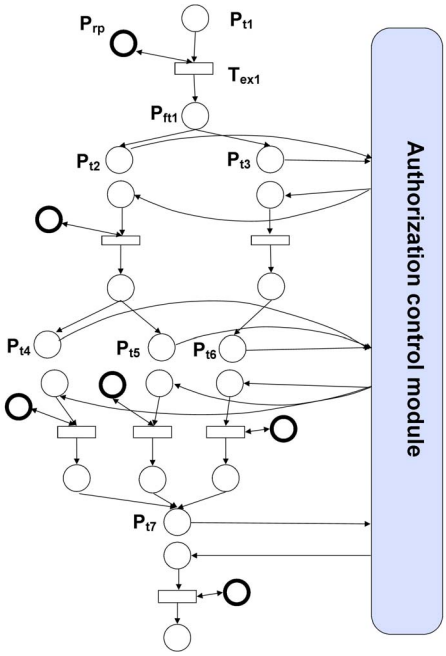


Figure 4. Modeling workflow execution under authorization control; only task t_1's CTPN components are labelled

Figure 4 models the execution of the workflow in Figure 3 under authorization control. The Authorization Control Module (ACM) is the authorization model for the entire workflow, which is assembled from individual authorization modules. It can be seen that there are clear interfaces between the actual execution of the workflow and ACM. When task t_i is ready (i.e. it has no predecessor or its

predecessors have been completed), a token t is deposited into place P_{ti}, which is the same P_{ti} place in the ACM. This starts the authorization process in the ACM for the task. After the authorization is completed, a *urt* token is deposited by the ACM into the P_{urti} place. The task is now assigned a role and a user, and can undergo the resource allocation procedure.

In Figure 4, the place P_{np}, depicted as a bold circle, represents the resource pool. In the place, a token represents a resource (e.g., a compute node), and the number of tokens represents the number of resources currently available in the system. The T_{ex} transition represents the resource allocation and task execution. There is only one resource pool (therefore, a single P_{np} place) in the model, connecting to all T_{ex} transitions. Figure 4 is drawn in the way that every T_{ex} transition is associated with a separate P_{np} place; we do this for the sake of clarity (to avoid many more arcs crossing the figure). These individual P_{np} places should be regarded as a single P_{np} place. A token in the P_{np} place is defined as $np = nid$. In this model, the execution of a task is modeled in the following way. If there are tokens in the P_{np} place (i.e., there are free resources) and there is the urt_i token in P_{urti}, the T_{exi} transition fires immediately (indicating the start of t_i's execution) and a t token is deposited into the P_{tj} place (suppose t_j is t_i's child). The timestamp of the deposited token t will be set as the current global time plus t_i's execution time, i.e., $t_j@ts = gt + t_i.e$. Therefore, the t_j token is not allowed to be fired until the time duration of $t_i.e$ has lapsed, which simulates task t_i's execution.

Moreover, after a task is completed, the role and user assigned to the task need to be returned to place P_r and P_u, so that the role and the user can be assigned to other tasks. This procedure is modelled as follows. When T_{exi} fires, an r token is deposited back to the P_r place and a u token to the P_u place. The time stamps of these two tokens are both set as $gt+t_i.e$. Similarly, when T_{exi} fires, an np token is deposited back to the P_{np} place (expressed as double-headed arrows). $np@ts$ is also set as $gt+t_i.e$, which means although the np token is back to the P_{np} place, the corresponding resource is only allowed to be allocated to other tasks when the simulated execution of t_i has been completed.

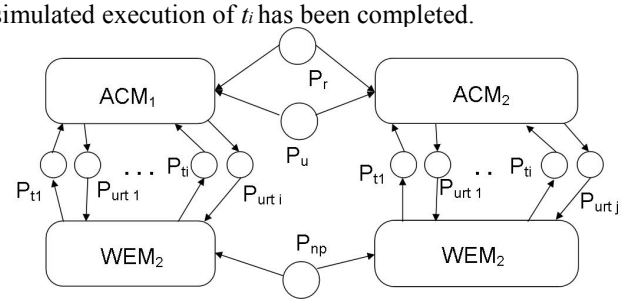


Figure 5. Authorization and executions of multiple workflows

If multiple workflows are running (and each workflow can have many instances) in the system, an Authorization Control Module (ACM) and a Workflow Execution Module (WEM) need to be constructed for each workflow. The model structure for authorising and executing multiple workflows is shown in Figure 5. As can be seen from the figure, the different workflow authorization modules only share place P_r and P_u, and the different workflow execution modules only share the P_{np} place.

V. Model Simulations and Analysis

The modeling mechanism presented in this paper has been implemented using the CPN Tools [12]. The CPN Tools are a software platform that is able to construct and simulate Petri-net models. CPN provides a flexible mechanism that allows users to monitor a set of tokens, places and/or transitions. The runtime status of these tokens, places and transitions can be automatically collected during model simulations. This functionality of CPN is utilized in this paper to analyze and evaluate a constructed Petri-net model in terms of authorization overhead and various performance metrics. In this section, we first analyze various performance data based on the constructed models, and then propose the methods to improve performance in the presence of authorization policies.

A. Analyzing overhead caused by authorization

The authorization overhead can be represented by the time a task has to wait for accomplishing role and user assignment. Analyzing authorization overhead in a constructed model can give us insight which authorization requirements are causing performance degradation under the current security deployment.

1) Overhead caused by cardinality and temporal constraints

Figure 1 shows the role and user assignment module, which is the building block of the authorization module for the entire workflow. During the model simulations, the CPN tool can extract the time stamp of a token in Place P_t (i.e., the arrival time of the corresponding task) and the time stamp of a token in Place P_{rt} when the task has been assigned a role. The difference between these two time stamps is the time the task has to wait for the role assignment. Similarly, we can calculate the waiting time regarding user assignment. In the rest of the subsection, we focus on analyzing the overhead associated with roles, and the overhead associated with users can be analyzed in a similar fashion.

r_i^e denotes the instance of role r_i which has the earliest available time (the number of instances of role r_i depends on the cardinality constraint).

$w(t_j, r_i)$ denotes the waiting time for accomplishing the role assignment. The waiting time may be caused by two factors: r_i's cardinality constraint and r_i's temporal constraint.

The waiting time caused by r_i's cardinality constraint, denoted as $w_{CD}(t_j, r_i)$, can be calculated using Eq.10, where $TS(r_i^e)$ and $TS(t_j)$ denote the time stamp of role instance r_i^e and task t_j, respectively, representing the time point when

the role instance and the task are ready for task-role assignment.

$$w_{CD}(t_j, r_i)=\max(TS(r_i^e)-TS(t_j), 0) \quad (10)$$

The waiting time caused by r_i's temporal constraint, denoted as $w_{TP}(t_j, r_i)$, can be calculated by Eq.11, where $ld^e(r_i)$ is the next earliest time when r_i becomes available, gt is the current global time of the system. $ld^e(r_i)$ can be determined by D (the activated durations of r_i) as defined in Eq.2.

$$w_{TP}(t_j, r_i) = \begin{cases} 0 & [gt, gt + t_j.e] \subseteq r_i.D \\ ld^e(r_i) - TS(t_j) & otherwise \end{cases} \quad (11)$$

2) Overhead caused by role constraints and duty constraints

Role constraints and duty constraints have the similar impact, i.e., limiting the range of roles eligible for role assignment.

The overhead caused by role and duty constraints are represented in the following way. We first calculate the overhead of assigning role r_i to task t_j when applying the associated role (or duty) constraint, which is shown in the above subsection, and then calculate the role assignment overhead disregarding the role (or duty) constraint. The difference between them is deemed as the overhead caused by the role (or duty constraint).

B. Calculating other performance metrics

The CPN tool can evaluate the overall performance of a constructed Petri-net model in terms of various metrics. For example, resource utilization can be evaluated by monitoring the number of tokens in the place corresponding to the resource pool (which is P_{rp} in our model). Assuming the initial marking of the place is n, and during model simulations the average number of tokens in the place observed by CPN is avg_n, then, the resource utilization, denoted as ru, can be calculated as:

$$ru = 1 - avg\_n/n$$

It is also straightforward to determine the response time of a workflow in CPN. The tool can extract the time stamp when a workflow arrives at the system, and the time stamp when the last task in the workflow is completed. The difference between these two time stamps is the response time of the workflow. Based on the information, we can easily calculate the mean response time of all workflows. Other performance metrics that the CPN tools are able to evaluate include throughput and deadline miss rate. In next section, the simulation results are presented in terms of mean Response Time (RT) of workflows and Resource Utilization (RU) of the computing resource pool, the performance in terms of deadline miss rate and throughput is correlated with response time and utilization, respectively.

C. Improving performance under the authorization control

Generally speaking, there are two possible approaches to improving performance in the presence of authorization control. On one hand, the company can relax the authorization constraints which are causing significant overhead. On the other hand, the company can change the system deployment that is not directly related to the authorization policies, such as the resource capacity and the scheduling strategy.

The first approach is not always feasible since the presence of the authorization constraints may be mandatory for security reasons. However, if some authorization constraints are allowed to be relaxed in certain circumstances, we can apply the modeling and analysis methods presented in this paper to calculate the overhead caused by each authorization constraint, and therefore determine which authorization constraints are significantly hampering the performance. Which authorization constraints can be relaxed mainly depends on the company's viewpoints and the nature of the businesses being processed.

In this paper, we focus on the second approach, i.e., improving the performance by 1) changing the scheduling strategy, and 2) changing the amount of resources.

1) Improving performance by changing the scheduling strategy

The scheduling strategy can have impact on the cardinality overhead. When the number of the tasks running under a role has reached the number set by the cardinality constraint, the new task requiring that role has to wait until one of those tasks is completed. The waiting time is regarded as the cardinality overhead. A good scheduling strategy could reduce the tasks' waiting time due to the cardinality constraints and therefore improve performance. The random scheduling is employed in the models presented in Section IV. In this paper, a new scheduling strategy is proposed in the rest of the subsection to enhance the performance by reducing the cardinality overhead.

Assume that r_i's cardinality constraint is c_i. In order to remove r_i's cardinality overhead, the number of the tasks running under role r_i should be less than c_i when a new task arrives requesting for running under role r_i. γ_i denotes the arrival rate of the requests for role r_i, and rt_i denotes the mean response time of the tasks assigned to run under r_i. γ_i can be obtained from the simulation results of the authorization model. rt_i can be determined as follows. According to Little's Law in Queuing theory [17], we have $c_i=\gamma_i \times rt_i$. Therefore, in order to eliminate r_i's cardinality overhead, rt_i should be less than c_i/γ_i, which is called the desired average response time of tasks assigned to r_i.

The proposed scheduling strategy is essentially the EDF (Earliest Deadline First) scheduling. Assume N types of workflow are modeled in the system. WF_i denotes workflow i, t_{ij} denotes task t_j in WF_i and e_{ij} denotes t_{ij}'s execution time. The instances of WF_i are initiated following the Poisson process with the initiation rate of λ_i^{WF}. During the simulation period, we first obtain which tasks in a workflow (i.e., t_{ij}) are assigned to role r_i. Based on the information, the average execution time of the tasks assigned to role r_i can be calculated. And then we calculate the ratio of the desired average response time (i.e., c_i/γ_i) of the tasks assigned to r_i to their average execution time. The ratio is denoted as η_i. When task t_{jj} in WF_k is assigned to role r_i, we attach a soft deadline, denoted as d_{ij}, to t_{ij}, and d_{ij} is set as t_{ij}'s ready time

plus $e_{ij} \times \eta_i$ (t_{ij}'s ready time is the time when the token corresponding to t_{ij} is deposited to the place P_{urt} corresponding to task t_j in WF_i in the authorization module). In the proposed scheduling strategy, the scheduler will schedule for execution the task with the smallest value of $d_{ij}-gt$ (gt is the current gobal time in the system). By doing so, the tasks which are more likely to cause the cardinality overhead will be put into execution first.

2) Improving performance by changing the amount of resources

The waiting time for resources can be calculated as the duration between the time when a token is deposited into the P_{urt} place and the time when the T_{ex} transition fires. Based on the model simulations, we can calculate the average resource waiting time of all tasks, denoted as w^{res}.

Assume we want to plan the resource capacity for a system such that the resource waiting time is w ($w< w^{res}$). The following method is proposed to approximate the amount of resources so that the above objective of capacity planning can be achieved.

Table 1. Role constraints of the tasks in the workflow

Task	Role constraints
t_1	r_1
t_2	r_2, r_3, r_5
t_3	r_3, r_4, r_5
t_4	r_3, r_4, r_5
t_5	r_3, r_4, r_5
t_6	r_2, r_3
t_7	r_2, r_6

From the constructed model, we know that a task's arrival time is the time when the token corresponding to the the task is deposited to the P_{urt} place. Therefore, from the model simulation, we can obtain the rate at which the tasks arrive for resources, denoted as λ_i^{res}. Also we can calculate the average execution time of these tasks, denoted as e.

According to Queuing theory [17], we have Eq.12, where n is the number of homogeneous resources. We can solve Eq.12 to obtain the number of resources such that the average waiting time is w. Note that Eq.12 is derived only by applying Little's law. Therefore, it does not require that the task arrivals follow a certain pattern (e.g., Poisson process).

$$w = \frac{\lambda^{res} e^2}{n\ (n\ - e\ \lambda^{res})} \tag{12}$$

After calculating the new resource deployment for achieving the improved performance, we can adjust the number of resources in the models and re-run the model simulation to verify the performance under the new system capacity subject to the authorization policies.

VI. SIMULATION RESULTS

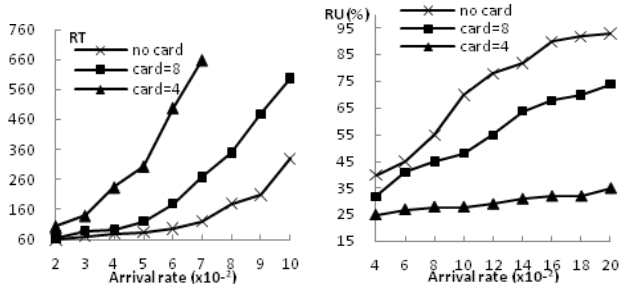
This section presents the simulation results of the CTPN model constructed for an exemplar workflow. The workflow consists of 7 tasks, whose topology is shown in Figure 5. The workflow instances arrive following the Poisson process. The execution time of a task follows an exponential distribution and the mean runtime of task t_1-t_7 is set to be 10, 15, 5, 10, 10, 20 and 25 time units, respectively. The role constraints are shown in Table 1. Assume that the following BOD and SOD constraints are imposed on the tasks, where $r(t_i)$ denotes the role assigned to task t_i.

C_1: $r(t_2) = r(t_4)$;C_2: $r(t_2) \neq r(t_5)$;C_3: $r(t_2) \neq r(t_7)$;

C_4: $r(t_6) \neq r(t_7)$;C_5: $r(t_3) = r(t_5)$;

1) Impact of Cardinality constraints

Figure 6 shows workflows' RT and RU in the existence of cardinality constraints as the workflow's arrival rate increases. In order to investigate the impact of cardinality constraints, no other constraints are imposed except the duty constraints. Each role has 4 users. There are 8 homogeneous resources in the pool.



(a) Mean response time (b) Resource utilization

Figure 6. Impact of cardinality constraints on RT and RU; "no card" means no cardinality constraints; "card=8" and "card=4" mean that the maximum number of tasks that a role can run simultaneously is 8 and 4, respectively

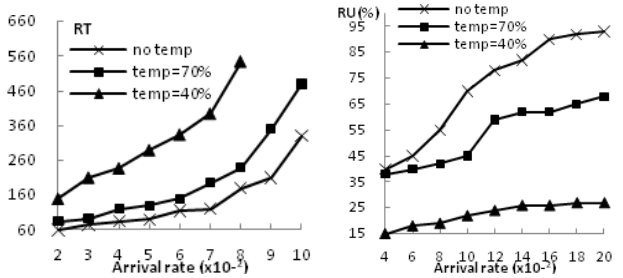
It can be observed from Figure 6a that when cardinality constraints are imposed, RT increases. This is because it is more likely that the tasks have to wait not only for resources, but also for the availability of roles. This result is to be expected and the qualitative analysis seems obvious. However, only through the modeling approach and simulation results can we acquire quantitative insight into how much impact a particular authorization constraint can have. For example, suppose the workflows' RT is desired to be no more than 300 time units. When there is no cardinality constraint, the system can accommodate a workflow stream with a mean arrival rate of up to 0.095. However, when the cardinality constraint is 8 and 4, the workload level that the system is able to handle is reduced to approximately 0.075 and 0.045, respectively.

As can be observed from Figure 6b, the cardinality constraints also have a negative impact on RU. Quantitatively, when there are no cardinality constraints, the resource utilisation is approximately 90% as the arrival rate increases. However, in the case of "card =8", the utilisation can only reach around 75%, and the performance is even worse (approximately 35%) when the cardinality parameter is 4. These results suggest that even if there are free resources in the system, the tasks cannot make use of them

because of the unavailability of roles due to the cardinality constraints. Through modeling and simulations, we are able to determine an optimized number of resources when we plan a system's capacity to support workflow executions under pre-specified authorization constraints.

2) Impact of Temporal constraints

Figure 7 demonstrates the impact of temporal constraints on performance in terms of RT and RU as the workflow instances' arrival rate increases. The temporal constraints on roles are set in the following way in the simulations in Figure 7. For each role, a time duration is selected from a period of 100 time units. The selected time duration occupies the specified percentage of the 100 time units. The starting time of the selected duration is chosen randomly. For example, to select a time duration which is 70% of 100 time units, the starting point of the duration is randomly selected from 0 to 30%×100 time units.



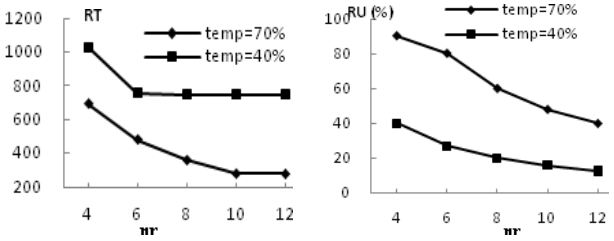
(a) Mean response time (b) Resource utilization

Figure 7. Impact of temporal constraints; "no temp" means no temporal constraints; "temp = 70%" and "temp = 40%" mean the roles are available for 70% and 40% of the time.

It can be observed from Figure 7 that temporal constraints have a negative impact on performance in terms of both RT and RU, as to be expected. Another interesting observation is that under temporal constraints, the performance seems to be less affected compared with the performance under cardinality constraints. This may be because the roles' availability period may overlap with each other. Therefore, when one role is not available for a task, the task may be able to find another available role.

3) Impact of computing resources

Figure 8 shows the impact on performance of increasing the number of resources (*nr*) when there exist cardinality and temporal constraints. As can be observed from Figure 8a, when the temporal constraint is 70%, RT decreases as *nr* increases, until *nr* reaches 10. Also, when temp=40%, RT decreases until *nr* reaches 6. This is because when *nr* is greater than a threshold, the workflow executions are mainly hampered by the authorization constraints. These results demonstrate that with the modeling and simulations, we can quantitatively investigate the impact of the deployed authorization policies, and therefore can potentially balance the authorization policy settings to remove the performance bottlenecks. The trend in Figure 8b is different from that in Figure 8a. In Figure 8b, RU keeps decreasing as *nr* increases. A closer observation shows that RU decreases linearly after *nr* is greater than 10 and 6 in the case of temp=70% and temp=40%, respectively. This is because of the same reason discussed above, i.e., an execution bottleneck is now caused by the authorization constraints and therefore, the increased resources will be largely idle. This result suggests that when planning system capacities, we should take the authorization policies into account and avoid over-provisioning unnecessary resources.



(a) Mean response time (b) Resource utilisation

Figure 8. Impact of increasing the number of resources (nr) in the existence of temporal and cardinality constraints; card=8; arrival rate is 7×10^{-2}

VII. CONCLUSIONS

This paper employs Color Timed Petri Nets to model video workflow executions under authorization control in cluster-based resource pools. Various authorization constraints are modelled in this paper, including role constraints, temporal constraints, cardinality constraints, separation of duty and binding of duty constraints. Scheduled activities are also supported in this model. In the model, invidual authorization modules can be automatically assembled to form the authorization model for the entire workflow. Therefore, it is easy to model a large collection of authorization policies. Also if there are changes in the workflow and authorization settings, the model can be quickly adjusted to recompute performance data. The model mechanism has been implemented using the CPN Tools. Various performance metrics can be computed from the constructed model, including both system- and application-oriented performance.

VIII. ACKNOWLEDGEMENT

This work is sponsored by the Research Project Grant of the Leverhulme Trust (Grant No. RPG-101), the National Significant Science and Technology Projects of China (01 Special, 2010ZX01045-001-002-5), the National Science Foundation of China (Grant No. 60803130), and the State Key Program of National Natural Science of China (Grant No. 61133005).

REFERENCES

[1] V. Atluri, "A Petri net based safety analysis of workflow authorization models", Journal of Computer Security, 8(2/3): 209-240, 2000

[2] X. Zhao, Z. Qiu, C. Cai, H. Yang, "A formal Model of Human Workflow", the 2008 IEEE International Conference on Web Services, pp. 195-202.

[3] D. Chakraborty, V. Mankar, A. Nanavati, "Enabling Runtime Adaptation of Workflows to External Events in Enterprise Environments", the 2007 IEEE International Conference on Web Services, pp.1112-1119.

[4] J. Crampton, "A reference monitor for workflow systems with constrained task execution", Proceedings of the tenth ACM symposium on Access control models and technologies, pp. 38-47, 2005

[5] Y. Jin, S. Reveliotis, "A generalized stochastic Petri net model for performance analysis and control of capacitated reentrant lines", IEEE Transactions on Robotics and Automation, 19(3): 474 - 480, 2003

[6] L. He, M. Calleja, M. Hayes, S.A. Jarvis, Performance prediction for running workflows under role-based authorization mechanisms," Proc. of the 2009 IEEE International Symposium on Parallel & Distributed Processing, IEEE Computer Society Press.

[7] L. He, S.A. Jarvis, D.P. Spooner, H. Jiang, D.N. Dillenberger, G. Nudd, "Allocating Non-real-time and Soft Real-time Jobs in Multiclusters", IEEE Transactions on Parallel and Distributed Systems, 17(2):99-112, 2006

[8] S.H. Kim, J. Kim, S.J. Hong and S. Kim, "Workflow-based Authorization Service in Grid", in 4th International Workshop on Grid Computing, Phoenix, USA, November 17, 2003, pp. 94–100.

[9] S. Manolache, "Schedulability Analysis of Real-Time Systems with Stochastic Task Execution Times", Ph.D Thesis, Department of Computer and Information Science, IDA, Linkoping University

[10] K. Tan, J. Crampton and C. Gunter, "The consistency of task-based authorization constraints in workflow systems", In Proceedings of 17th IEEE Computer Security Foundations Workshop, pp. 155–169, 2004

[11] T. Ziebermayr and S. Probst. "Web Service Authorization Framework", The 2004 International Conference on Web Services, pp.614.

[12] K. Jensen, L. Kristensen, L. Wells, "Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems", Intl. Journal on Software Tools for Technology Transfer, 9(3), pp.213-254, 2007

[13]http://www.telestream.net/pdfs/whitepapers/wp-video-workflow-management.pdf

[14] S. Chen, M. Shyu, N. Zhao, "SMARXO: towards secured multimedia applications by adopting RBAC, XML and object-relational database", Proceedings of the 12th annual ACM international conference on Multimedia, 2004

[15] J. Joshi, Z. Li, A. Ghafoor, "A Model for Secure Multimedia Document Database System in a Distributed Environment", IEEE Trans. On Multimedia, 4(2), pp.215-234, 2002

[16]http://www.davidsystems.com/fileadmin/pdf_data/Workflow_VideoApps.pdf

[17] L. Kleinrock, Queueing system, John Wiley & Sons, 1975