A Formal Approach to Design Optimized Multimedia Service Overlay

Hirozumi Yamaguchi
Graduate School of Info. Sci. and Tech.
Osaka University
Toyonaka, Osaka 560-8531 JAPAN
h-yamagu@ist.osaka-u.ac.jp

Khaled El-Fakih
School of Engineering
American University of Sharjah
P.O. Box 26666, Sharjah, UAE
kelfakih@aus.ac.ae

Akihito Hiromori
Graduate School of Info. Sci. and Tech.
Osaka University
hiromori@ist.osaka-u.ac.jp

Teruo Higashino
Graduate School of Info. Sci. and Tech.
Osaka University
higashino@ist.osaka-u.ac.jp

ABSTRACT

Service overlay networks have recently attracted tremendous interests. In this paper, we propose a new integrated framework for specifying services composed of service components running on different service nodes and for executing the services considering efficient utilization of overlay network resources. For a given service description written in an extended Petri net model, our method automatically derives a set of descriptions of service nodes' behavior which specifies how service nodes on an overlay network collaborate to provide the specified services. The derived descriptions minimize channel utilization, total response time or load of service nodes based on a given cost criterion. The experimental results show that a multimedia service for decorating and transcoding video contents can be well specified and implemented.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed Applications

General Terms

Algorithms, Design

Keywords

Overlay Network, Service Composition, Multimedia Service

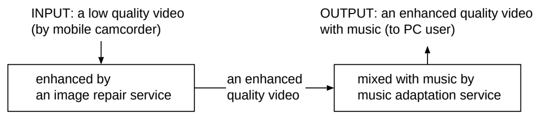
1. INTRODUCTION

A service overlay network is composed of service nodes where application service components (mainly multimedia service components like filtering, media transcoding and proxy services) are running. Virtual connections are assumed between those service nodes. A service overlay is federation of those service components to provide highly complex and integrated services to its service users. Several algorithms have been presented so far to design service overlay. For example, Xu et al. [13] has proposed a QoS-aware service path mapping algorithm. A service path is a sequential composition of service components (see Fig. 1(a)) and the service path is mapped onto service nodes so that the required QoS and network/computation resource constraints can be satisfied. Wang et al.[11] and Gu et al.[5] have extended the algorithm so that DAGs can be used to specify services. The other recent publications [2, 4, 12, 1, 8, 10] have their own characteristics. Most of them focus on bandwidth provision between service nodes, assuming simple service specifications or traffic models (see Ref. [12] for brief survey).

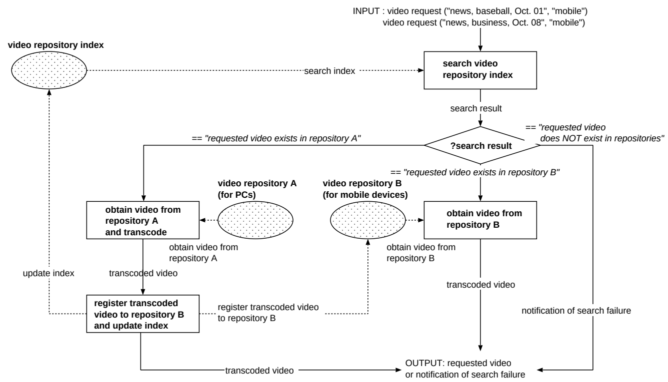
In this paper, we propose a formal approach to design optimized service overlay networks where a more general service class based on an extended Petri net can be treated. For a given extended Petri net description of service and a given overlay network topology where service nodes are fully-connected with each other, we automatically derive a set of descriptions of service nodes' behavior which specifies how those service nodes collaborate to provide the required service. Moreover certain costs of the service can be optimized to provide better utilization of overlay network resources even though the service includes multimedia contents processing and transmission. The experimental results, using an example description of video contents decoration/transcoding service and a real network simulator, have shown applicability of our method.

Compared with the existing methods[13, 11, 5, 2, 4, 12, 1, 8, 10], the major contributions of this paper are followings. (i) We can consider *service resources* such as multimedia contents repositories located on some service nodes. For such a service that includes service resources, we should consider not only simple service path mapping, but also *design of protocols for exchanging the contents of those service resources on overlay networks.* Fig. 1(b) shows such a service that uses service resources. This provides a video con-

NOSSDAV'05, June 13–14, 2005, Stevenson, Washington, USA.



(a) A service path example in Ref. [13] by Xu et al.



(b) Our target service.

Figure 1: Comparison of service descriptions.

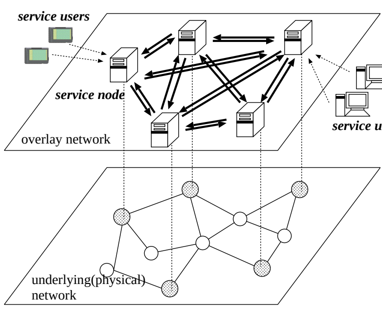


Figure 2: Service overlay architecture.

tents retrieval and transcoding service, using three service resources; a repository A of video contents for PC users, a repository B of low bit-rate video contents for mobile device users and an index table of repositories A and B to process users' queries of video contents by keywords. Suppose that a mobile device user requests a video content with keywords. This request is processed using the index table. If the index search result indicates that the content is found in repository B, the video content is taken from B and is provided to the user. If it indicates that the content is found only in repository A, the video content is taken from A and transcoded into low bit-rate video. The transcoded video is stored to repository B for other users and is provided to the requested user. Since these repositories (service resources) are allocated to some service nodes, it is easily understood that executing this service on cooperative multiple service nodes requires a complex protocol between the service nodes. We automatically derive a set of service nodes' behavior descriptions from a given description of a service with service resources like Fig.1(b). (ii) *Overlay network resource utilization* such as bandwidth and computation power of service nodes can be optimized. Of course this is the main issue considered and solved elegantly in the existing work. However, for a more general service class which includes complex service flows with service resources, we need different cost criteria and cost optimization methodology. Moreover, we have conducted realistic experiments, using a real network simulator and a realistic example. In general, theoretical optimization may not be able to capture all the situations in system usage. Thus we have proved that our optimization is really effective in real environments, under various arrival ratios of service requests.

2. DERIVING BEHAVIOR DESCRIPTION OF SERVICE NODES

A service overlay is composed of service nodes on which service components are running, and between any pair of service nodes, there exists a virtual channel (*i.e.* a unicast channel) called an *overlay channel*. An example of the architecture is shown in Fig.2. Service users use appropriate services from outside of the service overlay networks. From these service users, the architecture of service overlay networks is completely hidden and it is seen as a single server. However, actually the service nodes have to cooperate to provide the required services under given limitation of network and computing resources.

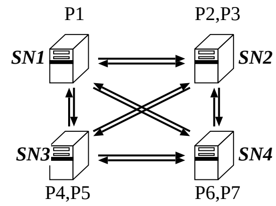
2.1 Inputs to the Algorithm

[**Service Overlay Graph**] We model the architecture as a complete directed graph, where nodes SNi $(i = 1, 2, ...)$ correspond to service nodes and edges correspond to overlay channels between service nodes. This graph is given as an input to the algorithm. Fig. 3(a) shows an example of a service overlay graph.

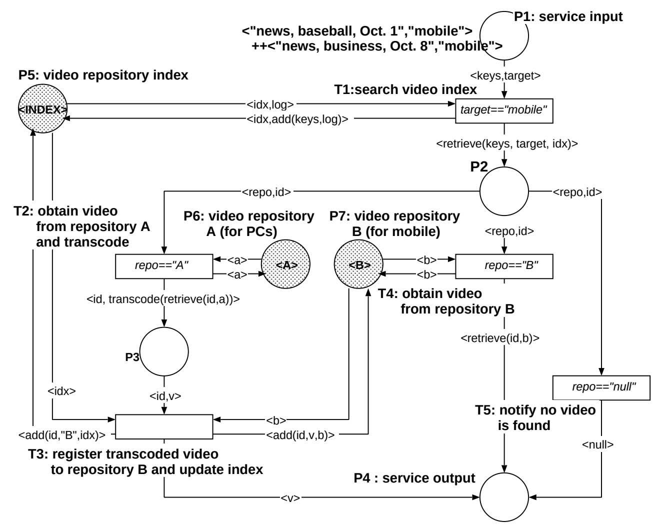
[**Service Description**] A service description is also given to the algorithm. Fig. 3(b) shows an example of a service description. This example corresponds to the one in Fig. 1(b) and is written in an extended Petri net called a *predicate/transition-net* (*Pr/T-net*)[3]. We use Pr/T-nets for the description of services.

Intuitively, in Pr/T-nets, each arc from a place p to a transition t has a form of a linear sum $L(p, t) = \sum_{1 \le i \le n} k_i X_i$ of n-tuples of variables (n can be an arbitrary integer for each place) where k_i is a non-negative integer and X_i is a tuple of variables like $\langle x_1, x_2, ...x_n \rangle$. This is called an *arc label*. Each token in place p is a n-tuple of values $C_i = \langle c_1, c_2, ...c_n \rangle$, and a set of tokens which can be assigned to arc label $L(p, t)$ is called an *assignable set*. Moreover, a transition may have a logical formula of variables from the labels on input arcs of t, called a *condition*. The arc from transition t to a place p' also has an arc label, which is a form of a linear sum $L(t, p') = \sum_i k_i Y_i$ of n-tuples of functions where k_i is a non-negative integer and Y_i is a tuple of functions from the variables on the labels of input arcs of t. A transition t may fire *iff* there exists an assignable set in each input place and the assignment of values to variables by the assignable set satisfies the condition of t. If t fires, new sets of tokens are generated and put into the output places according to the labels on the output arcs of t. Note that the data type of token values in a place and the data type of labels of the arcs from/to the place must be the same.

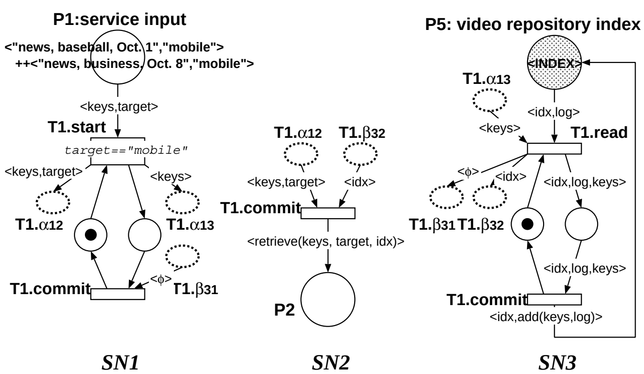
For modeling services, we define the following class of Pr/T-nets. Places are classified into two types, (i) *resource places* and (ii) *service flow places*. Resource places represent service resources such as databases. The number of tokens in the resource places never increases or decreases, *i.e.*, the transitions from the resource places are represented as self-



(a) Service overlay graph (with allocation of places).



(b) Service of Fig. 1(b) described in a Pr/T-net.



(c) Service nodes' behavior descriptions (for transition $T1$)

Figure 3: Derivation example.

loops where the contents of the resources may be changed by executing the transitions from the resource places. In Fig. 3(b), places $P5$, $P6$ and $P7$ are such resource places. The other places are service flow places. By removing the resource places and their self-loop arcs, the service flow places and the transitions form a partial Pr/T-net which models the execution flow of the service. It is called a *service flow net*. The resource places and associated transitions are used to represent the utilization of service resources in the service flow net. To model multiple users who use the same service, multiple tokens with different values can be given in the service flow net. This class of Pr/T-net considerably extends the expressive power to describe service requirements, compared with the existing work (most of the work considers only sequential services or services represented as DAGs).

In Fig. 3(b), service flows are modeled by service flow places (denoted as white circles) and transitions, and video repositories and their index are modeled by resource places (denoted as meshed circles). Inside the service flow places, there may exist multiple tokens to distinguish different users' requests. For example, two tokens in place $P1$ represent two service requests. Pr/T-nets can handle such multiple flows in a single net, thus it is better to be used to write a system which handles multiple users. Similarly, resource places may have tokens inside, which represent the entities of resources. Moreover, inside the transitions (rectangles), conditions may be specified which use the variables from the arc labels of service flow places. Using the condition, we can describe conditional branches or iterations. For example, service flow place $P2$ has multiple output transitions that have conditions on the variable "repo" from the labels of the input arcs from $P2$ to these transitions.

[**Allocation of Places to Service Nodes**] The algorithm requires all the places of the given service description to be allocated to the service nodes. For example, in Fig. 3(a), places $P1$ to $P7$ are allocated to service nodes $SN1$ to $SN4$. overlays will be discussed in Section 3. An allocation may be designed manually by developers or an optimized allocation that minimizes certain costs of the service can be obtained using the Integer Linear Programming (ILP) technique described in Section 3.

2.2 Output

[**Description of Service Nodes' Behavior**] We assume that places $P1$, $P2$ and $P5$ are located on service nodes $SN1$, $SN2$ and $SN3$ respectively, as shown in Fig. 3(a). Assuming this allocation of places, we see, in Fig.3(c), a part of behavior of service nodes that corresponds to transition $T1$ to understand service nodes' behavior descriptions.

Each service node has its own behavior description written in a Pr/T-net. Between the Pr/T-nets, we introduce places for modeling asynchronous and reliable communication on overlay channels called *communication places*. We assume that two communication places with a common name "$T.X_{ij}$" (T represents the corresponding transition and X=α or $X = \beta$, explained later) in the Pr/T-nets of two different service nodes SNi and SNj represent message passing from service node SNi to service node SNj through the overlay channel between them. If a token is put on "$T.X_{ij}$" at service node SNi, the token is eventually removed and put it onto "$T.X_{ij}$" at service node SNj. Note that the prefix "T" means that these communication places are used with respect to the execution of transition T of the service description. Communication places are represented as dotted circles in the following figures.

The transition $T1$ is executed as follows. At first, service node $SN1$ takes a token (an assignable set) from $P1$ (user request spool) by firing of transition $T1.start$ and assigns it to the tuple of variables "$\langle keys, target\rangle$". If the assignment satisfies the condition of $T1$, then these values are sent to $SN2$ and $SN3$ via communication places $T1.\alpha_{12}$ and $T1.\alpha_{13}$, respectively, since $SN2$ needs both values to generate a token to $P2$ and $SN3$ needs the value of "keys" to generate a token to $P5$. In response, $SN3$ takes a token from $P5$ by firing of $T1.read$. Since the value of token taken from $P5$ is used to generate a token to $P2$, the value of "idx" is sent to $SN2$ via $T1.\beta_{32}$. Using these values sent from $SN1$ and $SN3$, $SN2$ generates a token to $P2$ by firing of $T1.commit$. Similarly, $SN3$ generates a token to $P5$ by firing of $T1.commit$. Note that $SN3$ sends an empty value ϕ to $SN1$ via $T1.\beta_{31}$ to let $SN1$ know that $SN3$ could obtain a token from $P5$. After knowing it, $SN1$ is ready to accept the next token. For this mutual exclusion control purpose, we introduce two places with black dots (empty value tokens ϕ) in $SN1$ and $SN3$.

Tokens carried through $T.\alpha_{ij}$ and $T.\beta_{jk}$ are called *α-messages* and *β-messages*, respectively. An α-message through $T.\alpha_{ij}$ is used to carry values that will be used to generate token values at the receiver service node SNj as well as to let SNj obtain assignable sets from the input places of T allocated to SNj. A β-message through $T.\beta_{jk}$ is sent in response to reception of the α-message, and is used to carry the obtained values that will be used to generate tokens at SNk.

2.3 Algorithm

The basic idea of the algorithm is inspired by our protocol synthesis technique in Ref. [14]. For each transition t of the given service description, depending on a given allocation of places, we identify the set of service nodes called *reading service nodes* which have at least one input place of t, and also the set of service nodes called *writing service nodes* which have at least one output place of t. Then we select one of the reading service nodes as the *primary service node*. Afterward, in order to execute t over multiple service nodes, we apply to transition t the derivation algorithm based on a protocol called a *transition execution (TE) protocol* that determines how the behavior of t is simulated by service nodes. By decomposing every transition of service description based on the TE protocol, we finally obtain the set of behavior descriptions of service nodes.

3. FOR OPTIMIZED SERVICE OVERLAY

The algorithm described in the previous section assumes that an allocation of places is given. In this section, we determine an optimal allocation of places to service nodes to derive an optimized service overlay. The optimal allocation of places leads to the optimized service overlay where its *cost* is minimized. The cost can be one of *maximum channel utilization*, *maximum response time* and *maximum load of service nodes*. If we choose one of these costs to be minimized according to application domains, we may give some constraints on the other metrics. Using the optimal allocation of places, we can derive an optimized service overlay according to the algorithm in the previous section.

We derive an optimal allocation of places using an Integer Linear Programming (ILP) problem. We introduce the following 0-1 integer (boolean) variables; (i) $\alpha^t_{i,j}[v]$: one if an α-message is sent from SNi to SNj and carries the value of a variable v on the execution of transition t (zero otherwise), (ii) $\beta^t_{i,j}[v]$: one if a β-message is sent from SNi to SNj and carries the value of a variable v on the execution of transition t (zero otherwise), and (iii) alc^p_j : one if place p is allocated to SNj (zero otherwise).

The followings are the cost criteria. **(i) Maximum channel utilization**. Since these services deal with large-sized (*e.g.* orders of giga-bytes) resources transmitted between service nodes, it is very important to prevent those large-sized resources from being transfered at the same time through a single or a few overlay channel(s) with poor bandwidth. Here, we try to identify a set of transition instances that can be executed in parallel, for example, by maximum occurrence distance analysis [7]. Moreover, if two independent users execute different transitions at almost the same timing, those transitions can be also regarded as parallelized transitions. Using those techniques and service request patterns of users, we can identify multi-sets of transitions that may be executed in parallel. Let MT denote a set of those potential multi-sets of transitions. Here, for each overlay channel (i,j), we define the *maximum channel utilization* of (i,j), denoted as $util_{i,j}$, as the ratio of the maximum amount of data to be transmitted through the channel (i,j) at the same time, to the capacity of the channel. $util_{i,j}$ can be defined as follows.

$$util_{i,j} = \max_{M \in MT} \left\{ \frac{\sum_{t\in M}\sum_{v\in var(t)} SZ(v) * (\alpha^t_{i,j}[v] + \beta^t_{i,j}[v])}{BW(i,j)} \right\}$$

Here, $BW(i,j)$ denotes the capacity of overlay channel (i,j), $SZ(v)$ the size of a variable v, and $var(t)$ the set of variables that appear in the labels of the input arcs attached to the transition t. We may want to minimize $\max_{(i,j)\in E}\{util_{i,j}\}$, denoted as $util$, the maximum of the maximum channel utilization of all the channels to avoid concentration of bandwidth utilization where E denotes the set of the overlay channels. **(ii) Maximum response time**. The maximum response time of the service for a user, is the cumulative execution time of the transitions on the "longest path" of the service. According to the derivation algorithm, the execution time of transition t can be defined as the maximum of the transmission time of a sequence of an α-message from SNi to SNj and a β-message from SNj to SNk plus the maximum of the execution time of functions that appear in the labels of the output arcs from t to the places of t allocated to SNk. Let $TS(t,p)$ denote the sum of the task sizes of functions attached to the arc (t,p) and $PW(j)$ denote the size of tasks which SNj can process per unit of time. The maximum execution time of t, denoted as $exec^t$, can be defined as follows.

$$\begin{aligned} exec^t \quad = & \max_{(i,j),(j,k)\in E} \left\{ \frac{\sum_{v\in var(t)} SZ(v) * \alpha^t_{i,j}[v]}{BW(i,j)} \right. \\ & + \frac{\sum_{v\in var(t)} SZ(v) * \beta^t_{j,k}[v]}{BW(j,k)} \\ & \left. + \max_{p\in t\bullet} \left\{ \frac{TS(t,p)}{PW(j)} * alc^p_k \right\} \right\} \end{aligned}$$

where $t\bullet$ denotes the set of output places of transition t. Here, let LS denote the set of all the potential longest paths (transition sequences) of a given service. We may want to minimize $resp = \max_{L\in LS}\left\{\sum_{t\in L} exec^t\right\}$, the maximum response time of the service. **(iii) Maximum load of service nodes**. The maximum load of a service node SNj, say $load_j$, can be defined using MT, which was used in the definition of maximum channel utilization. $load_j$ is given as follows.

$$load_j = \max_{M\in MT} \left\{ \sum_{t\in M} \sum_{p\in t\bullet} \frac{TS(t,p)}{PW(j)} * alc^p_j \right\}$$

We may want to minimize $load = \max_{j\in S}\{load_j\}$, the maximum load of service nodes where S denotes the set of service nodes.

Here, according to application domains, we can choose one of the above metrics to be optimized, giving certain constraints on the others if necessary. However, due to limitation of space, hereafter we only present the ILP problem that minimizes maximum channel utilization where certain constraints are given to the maximum response time and maximum load of service nodes.

The following is the objective function.

$$\min \ util \tag{1}$$

From the definition of "max" functions, the following constraints are necessary.

$$\forall(i,j) \in E; \ \ util \geq util_{i,j} \tag{2}$$

We may want to set certain thresholds of the maximum load $load_j$ of each service node SNj and the maximum response time $resp$. Let $LTH(j)$ and RTH denote the thresholds of $load_j$ and $resp$, respectively. We obtain the following constraints.

$$\forall j \in S \ \ load_j \leq LTH(j) \tag{3}$$

$$resp \leq RTH \tag{4}$$

In addition, we need the following constraints some of which come from the definitions of variables and others from the algorithm. pri_i^t is a 0-1 integer variable and the value is one *iff* SNi is the primary service node in the execution of transition t of the service. Ps is the set of service flow places.

$\forall(i,j) \in E, \forall t \in T, \forall p \in \bullet t, \forall p' \in t\bullet,$
$\forall v \in var(L(p,t)) \cap var(L(t,p'));$

$$\alpha_{i,j}^t[v] \ \ \geq \ \ alc_i^p + alc_j^{p'} + pri_i^t - 2 \tag{5}$$

$$\beta_{i,j}^t[v] \ \ \geq \ \ alc_i^p + alc_j^{p'} - pri_i^t - 1 \tag{6}$$

$$\forall(i,j) \in E, \forall t \in T, \forall v \in var(t); \ \ \alpha_{i,j}^t[v] + \beta_{i,j}^t[v] \leq 1 \tag{7}$$

$$\forall p \in P; \ \ \sum_j alc_j^p = 1 \tag{8}$$

$$\forall t \in T; \ \ \sum_j pri_j^t = 1 \tag{9}$$

$$\begin{aligned} &\forall j \in S, \forall t \in T, \forall p \in \bullet t \cap Ps; \\ &\quad alc_j^p = pri_j^t \qquad \text{iff } |p\bullet| > 1 \\ &\quad alc_j^p \geq pri_j^t \qquad \text{otherwise} \end{aligned} \tag{10}$$

We note that if the ILP problem is hard to obtain an optimal solution within realistic computation time, several techniques can be used such as ε-approximation or linear relaxation.

4. TOOL SUPPORT, APPLICATION EXAMPLE AND EXPERIMENTAL RESULTS

[**Toolset**] We have developed a toolset in Perl, which works together with a graphical tool "CPNtools". In the toolset, we first describe the service using CPNtools. Second, our toolset parses the given description of service in the CPNtools format (described in XML with DTD) using XML Parser, and generates the corresponding ILP model presented in Section 3. Third, we use the tool CPLEX to solve the ILP problem and determine the optimal allocation of places. Fourth, using the optimal allocation of places, our Perl toolset generates the corresponding set of service node behavior descriptions.

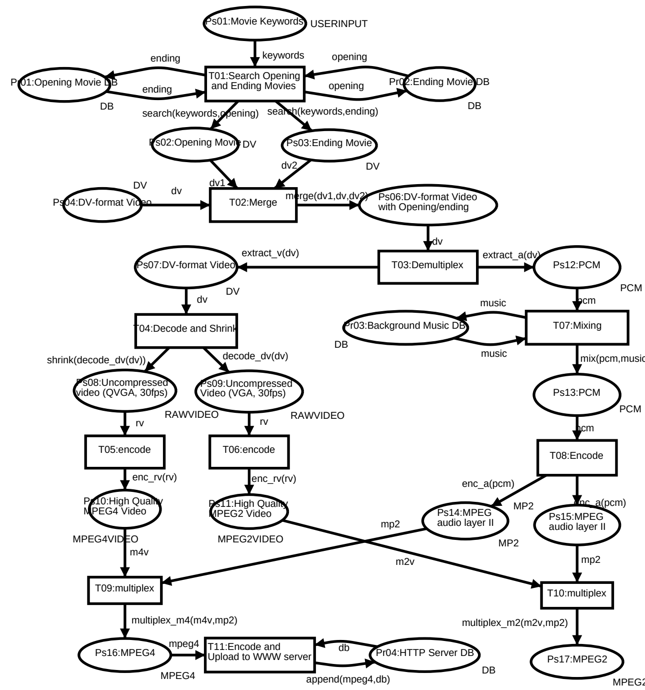
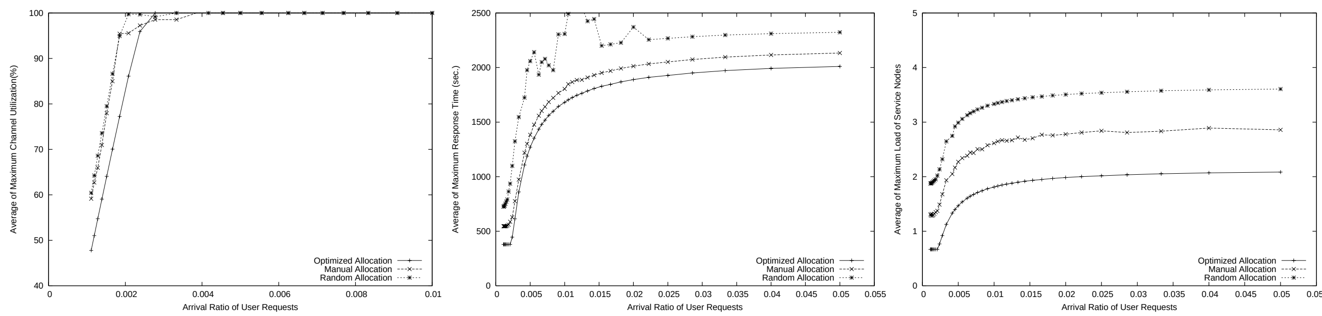


Figure 4: Automated Video Decoration and Transcoding Service

[**Application Example**] Here, we consider an automated video decoration and transcoding service on overlay networks as an application example. In this service, a given movie in DV format is decorated by opening and ending movies, and the decorated movie file is de-multiplexed into video and audio files. The quality of the video is adjusted depending on capabilities of users' devices and then encoded into appropriate formats, such as MPEG2 and MPEG4. The description is given in Fig. 4.

[**Experimental Results**] Since our optimization is based on estimated arrival ratios of user requests (recall that a set MT of parallelized transitions used in the definition of cost functions depends on arrival ratios of user requests), we need to validate that the values of cost functions are really improved by our optimization at any arrival ratio in realistic network environments.

To do so, we have developed a simulator toolset by letting two software tools collaborate with each other, (i) a high-level Petri net simulator called Maria[6] and (ii) a real network simulator GTNetS[9]. Maria executes the descriptions of service node behavior which are derived by our Perl toolset explained at the beginning of this section. If it finds a token in a communication place, it tells the transmission requirement to GTNetS with the size of token(s). GTNetS simulates a TCP-based overlay network and starts generating traffic according to the transmission requirement between two service nodes at packet level. When it finishes the transmission, GTNetS tells the end of transmission to Maria. Maria and GTNetS continue their operations cooperatively until Maria finds that no executable transitions is left.



(a) Maximum channel utilization (b) Maximum response time (c) Maximum load of service nodes

Figure 5: Service costs of example service (vs. user request interval).

From the given service and five service nodes, we have derived, using our Perl toolset, three different sets of service node behavior descriptions optimized using three cost functions introduced in Section 3; (i) maximum channel utilization, (ii) maximum response time, and (iii) maximum load of service nodes. We have assumed 0.01 arrival ratio of service user requests to determine MT in the cost functions. Then we have measured, using our simulator toolset, the average values of those metrics, varying the arrival ratios of service users' requests. We have given random values between 70 Mbps and 130 Mbps as the physical link capacity. We have used TCP as the overlay channels, and set each overlay channel capacity to the minimum capacity of links on the channel. We have set the sizes of contents as follows; Raw video file has 5 Gbytes and its corresponding formats are 4 Gbytes (DV), 1 Gbyte (MPEG2) and 128 Mbytes (MPEG4). In a multiplexed stream, we assume that the ratio of video:audio is 9:1.

For comparison, we have derived additional two sets of service node behavior descriptions for each cost function. (i) "*random allocation*" where video contents are allocated to service nodes with enough computation power and communication capacities and then other service components are allocated randomly. We obtain the best solution from 10 trials. (ii) "*manual allocation*" which is a manually well-thought-out allocation where we repeat trail and error so that sequences of service components can be executed without communicating with other service nodes as long as possible.

Fig. 5 shows the results. We note that the definition of the maximum channel utilization in Fig. 5(a) is different from the one in Section 3. We have measured in Fig. 5(a) the ratio of the bandwidth that is actually used, to the capacity of the channel, since it is a more natural metric to represent network performance. We can see that even under high arrival ratios of user requests, all the costs of the optimized service have the advantage over the random/manual allocations. In other words, even though the optimization is done using some (high) arrival ratio of user requests, the result under any ratio shows our advantage. The time for deriving optimized allocations was a few minutes in most cases.

5. CONCLUSION

In this paper, we have proposed an approach to design optimized service overlay networks. The introduction and evaluation of more adaptive QoS control mechanism depending on fluctuation of usable bandwidth of the overlay channels is part of our future work.

6. REFERENCES

[1] Z. Duan, Z.-L. Zhang, and Y. Hou. Service overlay networks: SLA, QoS, and bandwidth provisioning. *IEEE/ACM Trans. on Networking*, pages 870–883, 12 2003.

[2] X. Fu, W. Shi, A. Akkerman, and V. Karamcheti. CANS: Composable, adaptive network services infrastructure. In *USENIX USITS 2001*, 2001.

[3] H. J. Genrich and K. Lautenbach. System modeling with high-level Petri nets. *Theoretical Computer Science*, 1981.

[4] S. Gribble, M. Welsh, R. Behren, E. Brewer, D. Culler, N. Borisov, S. Czerwinski, R. Gummadi, J. Hill, A. Joseph, R. Katz, Z. Mao, S. Ross, and B. Zhao. The ninja architecture for robust internet-scale systems and services. *Computer Networks*, 2001.

[5] X. Gu, K. Nahrstedt, and B. Yu. Spidernet: An integrated peer-to-peer service composition framework. In *Proc. of IEEE Int. Symp. on High-Performance Distributed Computing (HPDC-13)*, 2004.

[6] M. Makela. *Maria – The Modular Reachability Analyzer for Algebraic System Nets –*. `http://www.tcs.hut.fi/Software/maria/`.

[7] T. Murata. Petri nets: Properties, analysis and applications,. *Proceedings of the IEEE*, 77(4):541–580, Apr. 1989.

[8] B. Raman and R. H. Katz. Load balancing and stability issues in algorithms for service composition. In *Proc. of IEEE INFOCOM 2003*, 2003.

[9] G. F. Riley. The georgia tech network simulator. In *Proc. of ACM SIGCOMM Workshop on Models, Methods and Tools for Reproducible Network Research*, pages 5–12, 2003.

[10] S. Vieira and J. Liebeherr. Topology design for service overlay networks with bandwidth guarantees. In *Proc. of IEEE IWQoS 2004*, 2004.

[11] M. Wang, B. Li, and Z. Li. sFlow: Towards resource-efficient and agile service federation in service overlay networks. In *Proc. of 24th Int. Conf. on Distributed Computing Systems (ICDCS2004)*, 2004.

[12] D. Xu and X. Jiang. Towards an integrated multimedia service hosting overlay. In *Proc. of ACM Multimedia*, 2004.

[13] D. Xu and K. Nahrstedt. Finding service paths in an overlay media service proxy network. In *Proc. of Int. Conf. on Multimedia Computing and Networking 2002 (MMCN2002)*, 2002.

[14] H. Yamaguchi, K. El-Fakih, G. Bochmann, and T. Higashino. Protocol sythesis and re-synthesis with optimal allocation of resources based on extended Petri nets. *Distributed Computing*, pages 21–35, 2 2003.