Accepted for publication, Journal of Digital Imaging. (Also PMRL Technical Report 93-5, Dept. of Electrical and Computer Engineering, University of Arizona, January 1993.)

A MODULAR METHOD FOR EVALUATING THE PERFORMANCE OF PICTURE ARCHIVING AND COMMUNICATION SYSTEMS*

William H. Sanders, Ph.D., Latha A. Kant, M.S. and Abhijit Kudrimoti M.S.

Department of Electrical and Computer Engineering
The University of Arizona
Tucson, AZ 85721 USA
(602) 621-6181

whs@ece.arizona.edu and lkant@ece.arizona.edu

Correspondence regarding the manuscript may be directed to William H. Sanders.

*This work was supported in part by a grant from Medical Systems Division, Toshiba Corporation and the Digital Equipment Corporation Faculty Program: Incentives for Excellence.

ABSTRACT

Modeling can be used to predict the performance of picture archiving and communication system (PACS) configurations under various load conditions at an early design stage. This is important, since choices made early in the design of a system can have a significant impact on the performance of the resulting implementation. Because PACS consist of many types of components, it is important to do such evaluations in a modular manner, so that alternative configurations and designs can be easily investigated. Stochastic activity networks (SANs) and reduced base model construction methods can aid in doing this. SANs are a model type particularly suited to the evaluation of systems in which several activities may be in progress concurrently, and each activity may affect the others through the results of its completion. Together with SANs, reduced base model construction methods provide a means to build highly modular models, in which models of particular components can be easily reused. In this paper, we investigate the use of SANs and reduced base model construction techniques in evaluating PACS. Construction and solution of the models is done using *UltraSAN*, a graphical oriented software tool for model specification, analysis and simulation. The method is illustrated via the evaluation of a realistically sized PACS for a typical United States hospital of 300-400 beds, and the derivation of system response times and component utilizations.

Keywords: Performance Evaluation, Picture Archiving and Communication Systems, Stochastic Petri Nets, Stochastic Activity Networks.

I Introduction

Completely automated network-based systems for the generation, storage, transportation, processing and presentation of digitized medical images in a hospital are expected to be an integral part of all hospital management systems in the next few years. These systems are called picture archiving and communication systems (PACS). PACS aspire to offer the advantages of better image accessibility, reduced overhead costs of image handling and improved diagnostic image quality. However, the huge amount of data handling and processing involved in PACS leads to problems related to limited storage and long processing and network delays, which may cause unacceptable response times to image viewing requests [1].

Model-based evaluation can be used to identify such problems and to investigate potential solutions in the early design phase of a project. If successful, such studies can aid in producing better system designs in a shorter period of time than would be possible if prototyping were the sole evaluation method employed. In particular, we can predict the performance of PACS under various conditions by varying different parameter values in the simulation, e.g., workloads and hardware specifications. Furthermore, simulation can be used to evaluate the algorithms and software processes which constitute the image processing aspect of PACS, e.g., the image compression and image prefetching algorithms [2]. The simulation of these algorithms and processes can then be used as a starting point for their implementation [1].

In this paper, we investigate the use of stochastic activity networks (SANs) [3, 4] and reduced based model construction methods [10] in constructing models of PACS. SANs are used to build models of PACS components (i.e., modalities, viewing workstations, networks, and database archives), and the reduced based model construction formalism is used to connect these components into different system configurations. This combination allows us to build models that are modular in the sense that many different configurations can be investigated by changing the interconnection of previously constructed component models. Construction and solution of the models is done using *UltraSAN* [7], a graphical software tool that aids in the construction and solution of systems represented as stochastic activity networks.

To illustrate the use of these methods, we consider a comprehensive model of a complete PACS intended for a typical United States hospital of 300-400 beds. The PACS consists

of three different types of imaging modalities (X-ray, ultrasound, and MRI) with several units of each type, a central database with short term storage and long term archiving facilities, and, a cluster of workstations for image viewing. The various system components are connected together by a high speed network. We have modeled the image generation process at a modality, the transmissions of new image folders to the database, the image migration algorithm in the database, the database operating system scheduling, and the processing of viewing requests from a workstation. We use the model to estimate the expected response times of viewing requests made at a workstation, the portion of this time due to database processing, the utilization of the communication network, and the utilizations of the hard disk and optical disk in the database. We also investigate a system configuration where all modalities are of type X-ray, which places the heaviest demand on the PACS.

The results presented are significant in that they illustrate the suitability of the stochastic activity network formalism in modeling PACS. For the particular design considered, the results also provide useful information regarding times to perform system operations and utilization of resources. These performance figures are important in identifying system bottlenecks and thus aid in the design of an efficient PACS. Specifically, we are able to 1) incorporate all the components of a PACS, i.e., modalities, database, workstations and network access protocols in a single model, 2) make use of the reduced base model formalism to represent the model in a modular manner, and 3) make use of the efficiencies of reduced base model construction to reduce the time required to simulate the resulting model [7, 6].

In the remainder of the the paper, we first describe the the PACS configuration and the operations of all the PACS components viz., imaging modalities, the central database, viewing workstations and the local area network connecting the different components together. We then briefly review the SAN and reduced base model construction formalisms and illustrate their use in modeling a portion of an image database. Next, we provide detailed descriptions of the SAN models of all the components. Finally we discuss the simulation of the model and the performance figures obtained from the simulation. Performance values have been obtained with the aim of identifying system bottlenecks and predicting the response time of the PACS to image viewing requests under varying intensity and mixture of workloads.

Table 1: Modality data relevant to PACS modeling

Name	No. of Units	Patient Arrivals/sec	No. Images per Patient	Image Size	Hard Disk Speed
MRI	1	0.000175	32	98.3 KB	2 MBps
Ultrasound	4	0.000075	36	0.786 MB	2 MBps
X-ray	4	0.0012	3	5.24 MB	2 MBps

II PACS Model Description and Assumptions

In this section, we describe the PACS whose performance has been evaluated. The system consists of a central database, several diagnostic imaging modalities, and a number of image viewing workstations, as shown in Figure 1. The imaging modalities and the workstations are connected to the central database by a 100 Mega bits per second (Mbps) LAN.

We have chosen to model four X-ray machines (with digital input mechanisms), four ultrasound scanning machines, and one magnetic resonance imaging (MRI) unit. This PACS size is about the same as would be found in a 300-400 bed hospital in the United States [9]. Table 1 provides data for the three modality types relevant to the modeling of the PACS, and was hypothesized using [9] and statistics on available hardware. Machines of a particular modality type are identical in terms of hardware and imaging software. We have also assumed that there are a certain number of viewing workstations in the PACS and that all workstations have identical hardware and run the same system software. The number of viewing workstations is varied, to study the effect on system performance.

It must be emphasized that the configuration and design choices made in this section are to illustrate the use of SANs in modeling PACS. While they represent one possible configuration for a 300-400 bed hospital, many different configurations are possible and may lead to different performance results. Indeed, the importance of this work is that it is possible to evaluate these alternatives, for the workload and hardware configurations at a particular site.

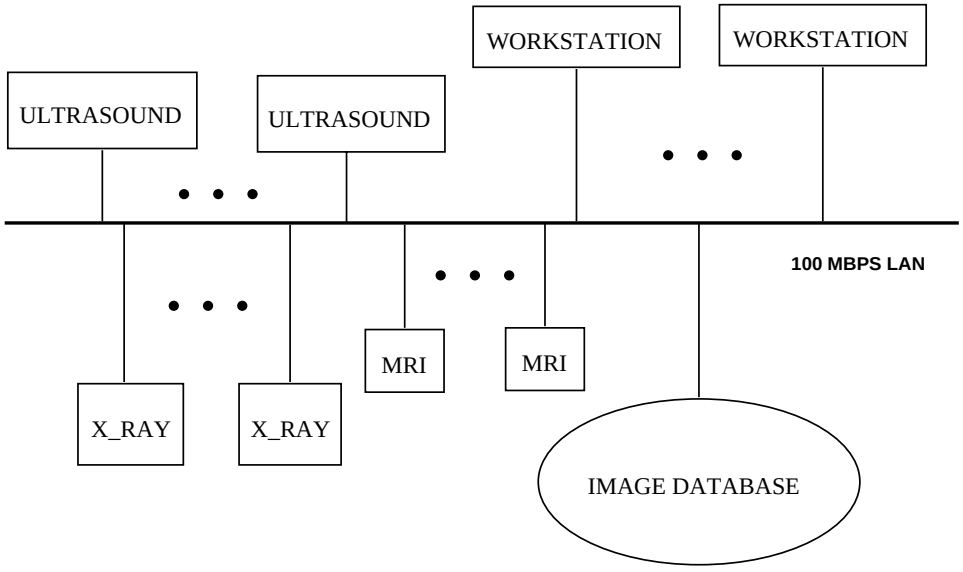


Figure 1: A picture archiving and communication system

A Imaging Modalities

The main function of an imaging modality in a PACS is to generate diagnostic images of a patient and transmit these images over the LAN to the database. It is assumed that patients arrive at a modality at a fixed rate, with time between two patient arrivals being exponentially distributed. The number of images generated for a particular patient would, in general, be variable and would also depend upon the modality type. Our model of an imaging modality is as follows. First, we assume that for each modality type, a fixed number of images is always generated, as given in Table 1. All the images generated for a patient are accumulated in a *folder*, and when the image generation process is complete, the folder is stored on a local hard disk. The newly generated folder is then queued up in a network interface buffer for transmission to the database. The delay for this procedure consists of the delay due to the writing of the folder to the hard disk, the time to read the folder from the hard disk and move it to the transmission queue, the queueing and transmission delay. These operations are shown graphically in Figure 2.

B Viewing Workstations

Workstations are used in a PACS for viewing images of patients. These images may be newly generated images which have just been stored in the database, or images generated

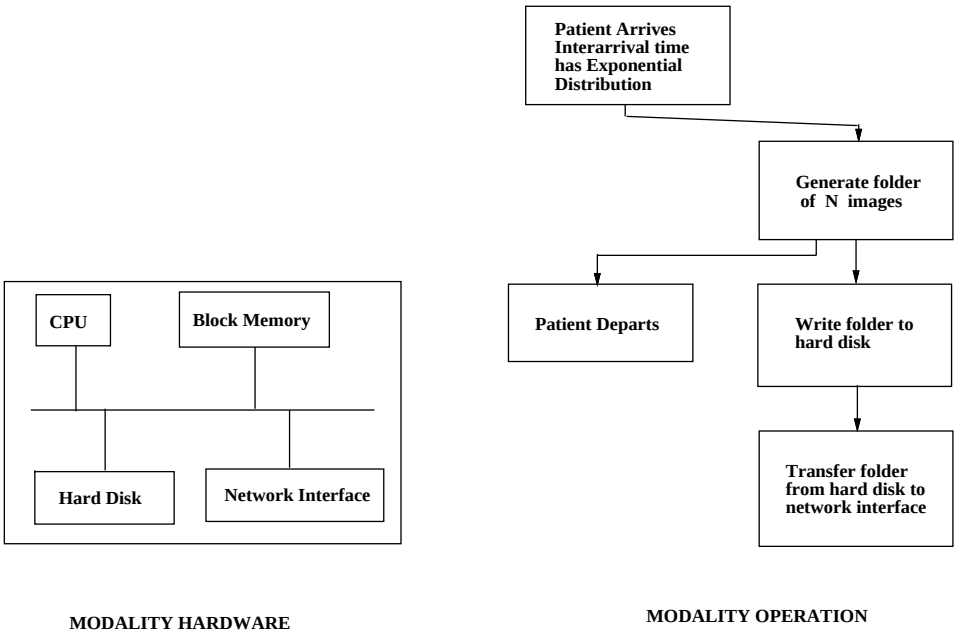


Figure 2: Image generation at a modality

in the past and archived in the database. It is assumed that a radiologist is present at all workstations at all times during the day continually viewing sets of images.

A radiologist's work at a workstation is assumed to consist of a continuous cycle of requesting a set of images, waiting for these images to arrive at the workstation, and then viewing the set of images. Furthermore, we assume that a radiologist views an image set for an exponentially distributed length of time. We have varied this time to study its impact on the performance of the PACS. We have also varied the workload mix by retaining the modality type with the highest resource demand (X-ray), and, turning off the other modalities in order to study its impact on the overall performance. We have assumed that a radiologist requests four images at a time. These images may have been generated at different modalities in the multiple modality case, or, at the particular modality in the single-modality case. The probability distribution of requests across modality types can be varied, and in the current study, is taken to be proportional to the normalized image generation rate (product of columns 3 and 4) at the various modality types as listed in Table 1.

When the radiologist makes the request for an image set, the workstation operating system translates this request into four requests (for the four images) and queues them into

a transmission buffer at the network interface hardware in the workstation. The requests are then transmitted serially to the database. The database examines each image request in turn, retrieves the image, and queues it in its transmission buffer to send it to the requesting workstation.

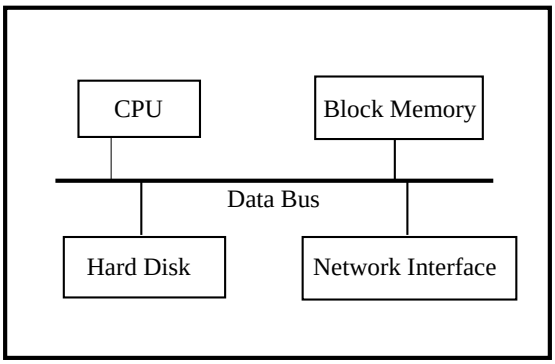
An image received by a workstation is buffered in its network interface and then transferred on the data bus to the local block memory. From there, it is written to the local hard disk. The hard disk access rate in the workstation is assumed to be 2 Mega Bytes per second (MBps). When the images have been received by the workstation and saved on its local hard disk, the request made by the radiologist is considered to be completely satisfied. The radiologist then views the image set for an exponentially distributed time before making the next request. Figure 3 illustrates the operation of a workstation, as described above.

C Local Area Network

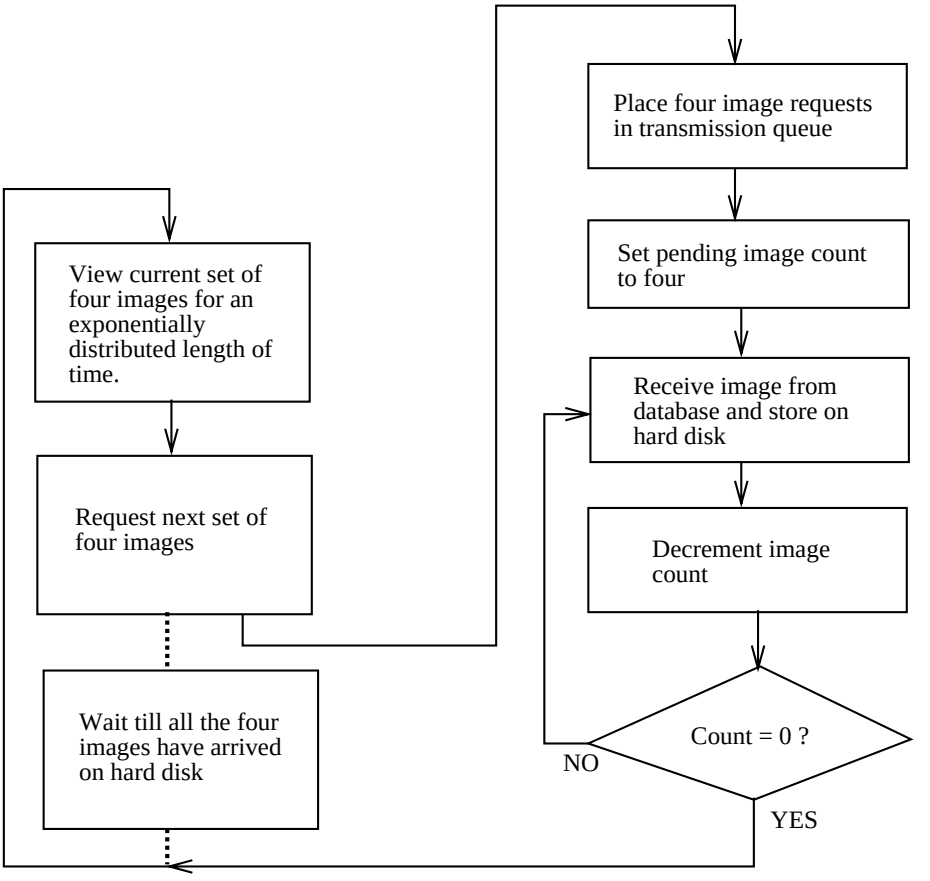
The workstations and imaging modalities all share a common link to the database. This is a local area network with a capacity of 100 Mbps.

Imaging modalities use the network to transmit a folder of newly generated images to the database for storage. The imaging unit queues up folders for transmission at the network interface buffer while awaiting the channel. This network interface buffer is common among all modalities, and, is capable of storing one complete folder generated by any given modality type. Transmission of a folder completes when the folder arrives at the database network interface buffer. The database at the other end cannot accept a new folder into its network interface buffer until the buffer has been cleared. This is done by removing a previously arrived folder from the buffer and sending it for further processing inside the database. After transmission is complete, the channel is released by the imaging unit and re-acquired if necessary.

Workstations use the network to transmit requests to the database and receive images from the database. As with imaging modalities, requests are transmitted one at a time, i.e., the channel is released after every transmission and re-acquired if more requests are queued up in the transmit buffer. The workstation receives a requested image in its network interface buffer and can accommodate an image of the maximum size in the PACS. Again, the buffer has to be cleared of the previously received image in order to receive another image. As mentioned earlier, the database queues all incoming requests and processes each request one at a time. When the required image is retrieved, it is queued up for transmission



WORKSTATION HARDWARE



RADIOLOGIST WORK CYCLE WORKSTATION OPERATION

Figure 3: Workstation operations

Table 2: Database data relevant to PACS modeling

Specification	Value
Compression Rate	16MPixels/s
Bits/Pixel:	
X-ray	10
Ultrasound	24
MRI	12
Hard disk access rate	10 MBps
Optical Disk access rate	300 KBps

and then transmitted one at a time to the requesting workstations.

These network operations are shown in Figure 4. Note that the media access control protocol is modeled very simply, without explicitly representing the control information that must be passed between nodes to achieve communication. This is because earlier studies (e.g., [8]) have found that this is a negligible part of the total transfer time on the network, and hence negligible to the total response time of the PACS.

D Central Database Archive

The central database archive is an integral part of the PACS. In addition to optical and hard disk storage, the hardware resources in the database include a network interface unit with finite buffer space and an image compressor unit. All these resources share a common data bus inside the database system. Table 2 provides data relevant to modeling the database.

The database serves four main purposes:

1. Store newly generated images on the hard disk cluster in a compressed format,
2. Implement an algorithm for prefetching images from optical disk to hard disk,
3. Process image viewing requests from workstations, and
4. Maintain a long term archive of images on the optical disk by transferring all images on the hard disk cluster to the archive at the end of each working day.

The time required to read an image from the hard disk is much less than that required to perform a corresponding read from the optical disk. This in turn implies that the time

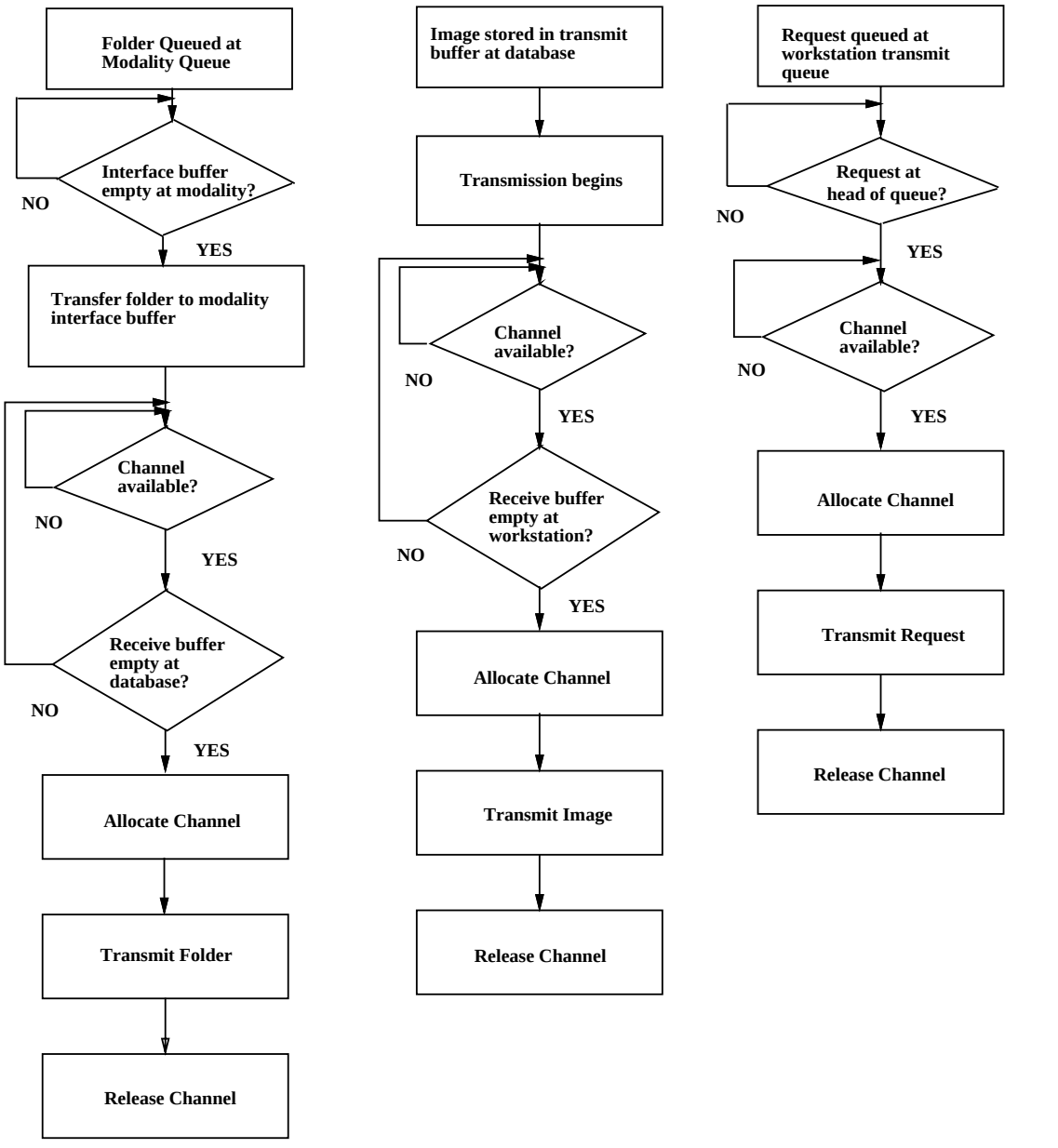


Figure 4: Network operations

to respond to a viewing request would be less if it were always located on the hard disk. But this may not always be the case, since images generated in the past are archived on the slow optical disk storage unit. Hence, in order to increase the probability of locating a requested image on the hard disk cluster and thereby minimize the expected response time, a prefetching algorithm is modeled. In this algorithm, a write to the hard disk due to an incoming folder, generates with a certain probability, a request to read a folder which already exists on the optical archive. The interpretation is that a related folder is prefetched to the hard disk, *e.g.*, a past folder of the same patient. We note in this context that a write to the hard disk could occur either due to a newly generated incoming image folder, or, due to fetching a particular image that was requested by a viewing workstation from the stored optical archive if it was not already available on the hard disk. Since a prefetch is performed only upon receipt of a new input folder, the algorithm first ascertains that this is indeed the case before proceeding with a prefetch.

The probability of prefetching a particular image folder is dependent on the image generation rate at the various modality units. From the data in Table 1, it is observed that this rate (product of columns 3 and 4) is a maximum for the MRI unit and a minimum for the ultrasound unit, with that for the X-ray unit falling in between. Thus, the probability of a prefetch (given by the normalized image generation rate) is the highest for the MRI folder and the least for the ultrasound folder, with that for the X-ray folder falling in between. Second prefetches on a single incoming folder are prevented by incorporating appropriate checks in the database model. Finally, at the end of the working day, all the images present on the hard disk cluster are once again archived on the optical disk.

The database operating system schedules a set of seven image-related hardware activities. The activities, which have been named in italics for convenience, are:

1. The transfer of a newly arrived image folder from the network interface receive buffer to the main memory area (*niu_bm_xfer*).

2. The passing of an uncompressed (newly arrived) image folder through the image compression unit and the writing of the compressed data back into main memory(*compression*).

3. The writing of a compressed image folder to the fast hard disk cluster in the database (*bm_hd_xfer*).

4. The reading of an image or folder from the slow optical disk archive (*od_read*).

5. The searching of the image database catalog to locate the requested image (*catalog_search*).

6. The reading of a compressed image from the hard disk cluster and its subsequent write to the main memory (*hd_read*).

7. The transfer of a compressed image from the main memory to the network interface transmit buffer (*bm_niu_xfer*).

Figure 5 is a flowchart of activity sequences for image storage, prefetching and retrieval in the database. The scheduling algorithm we model divides the above activity set into two subsets. Subset A contains activities *bm_niu_xfer, hd_read* and *catalog_search*, which are associated with the processing of requests generated by the workstations. Subset B contains activities *od_read, bm_hd_xfer, compression* and *niu_bm_xfer*, which are in turn connected with new folder acquistion and storage and a corresponding prefetch. The CPU of the database is required for execution of each activity. A CPU allocation request for an activity in this set is usually made by another activity in the same set. Subset A has a higher scheduling priority than subset B. Also, within a subset, the activities have been listed above in the order of their priority, e.g., activity *od_read* has the highest priority in subset B. This is just one possible scheduling algorithm, others could also be studied.

There is a certain amount of coupling between the two subsets, as is now explained. Based upon the location of the requested image, completion of *catalog_search* is followed by either one of the following two sets of activity sequences. If the image is located on the hard disk (a hit), then activity *hd_read* is scheduled which is followed by a *bm_niu_xfer*, both of which belong to subset A. If, on the other hand, the requested image is located on the optical disk (a miss), it first needs to be read off the optical disk before it can be transferred to the hard disk in order to be dispatched to the workstation. This now implies a scheduling of the activity sequence *od_read* and *bm_hd_xfer*, both of which belong to Subset B, prior to scheduling *hd_read* and *bm_niu_xfer*. Thus, based on the probability of a miss, we observe that a certain amount of coupling exists between the two subsets of activities. However, for lower response times, the probability of this coupled sequence should be very low, as it corresponds to the probability of the image not being present on the hard disk for fast reading.

The PACS under consideration is built from the four component types connected together via the LAN. In the next section we provide a brief review of the SAN modeling

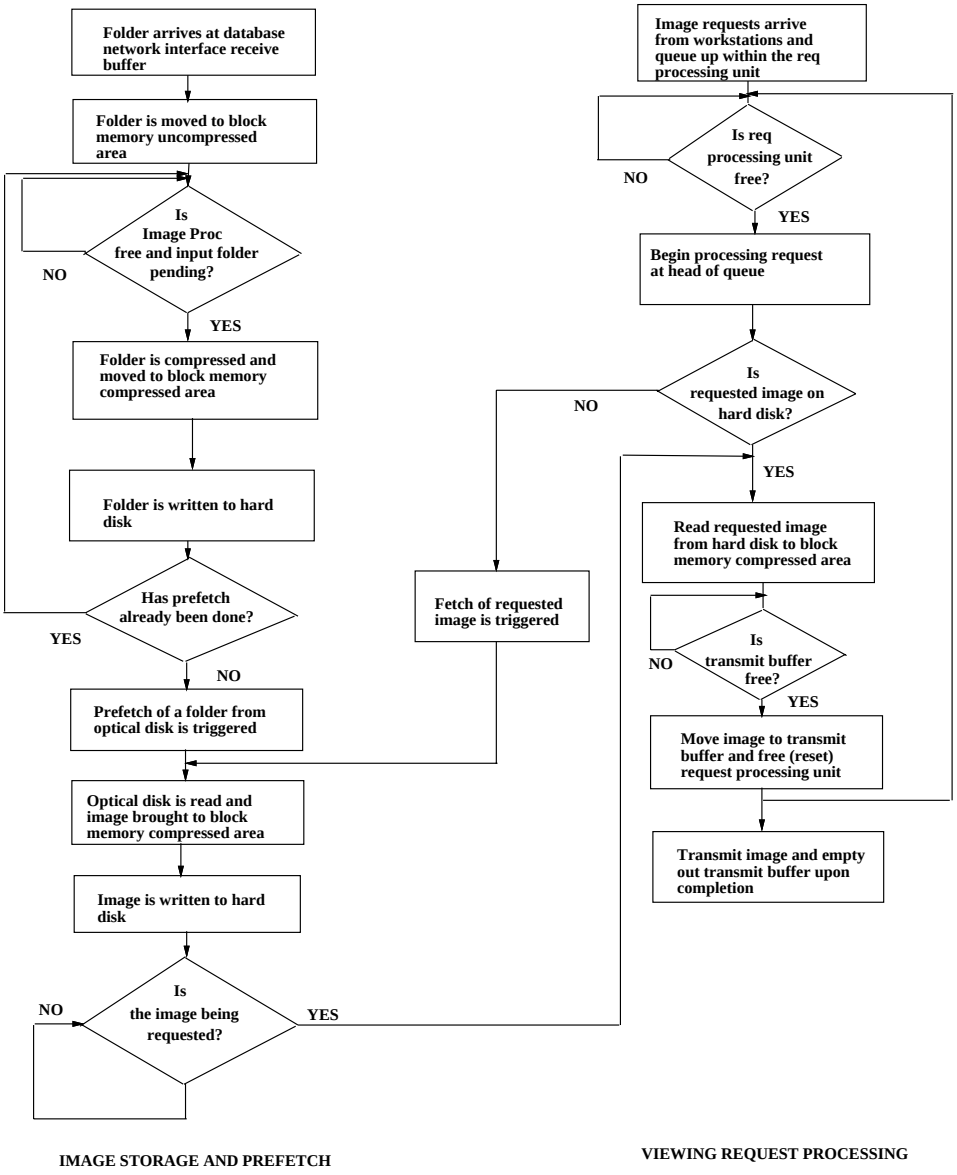


Figure 5: Database operations

formalism and then describe the modeling of a PACS configuration using SANs.

III Modeling Framework

The models employed are hierarchical, with stochastic activity networks of individual components replicated and joined together with other models.

A Stochastic Activity Networks

Stochastic activity networks (SANs) are a stochastic extension to Petri nets [3, 4]. Structurally, they consist of activities, places, input gates, and output gates. To illustrate activities and other SAN components, we consider a SAN model of a part of a PACS database, shown in Figure 6. The function of this model will be explained later; we introduce this model here only to illustrate the components that make up a SAN model.

Places are used to represent the "state" of a system (e.g., places *im_niu_mri, bm_out_place, od_read_place*, etc., in Figure 6) and may contain *tokens*. The number of tokens present in a place is called the *marking* of that place. The marking of the SAN is the vector consisting of the markings of all the places in the SAN. Tokens in a place may be given different interpretations, depending upon the purpose of the particular place in the model. For example, the number of tokens in *bm_out_place* represents the number of bytes of image data in the block memory of the database, while that in *cpu* indicates the availability or non-availability of the CPU.

Activities, which are similar to transitions in normal Petri nets, are of two types: *timed* and *instantaneous*. Timed activities (e.g., *bm_hd_xfer*) represent activities of the modeled system whose durations impact the system's ability to perform. Instantaneous activities (e.g., *t1*) on the other hand, represent system activities which, relative to the performance variables in question, complete in a negligible amount of time. When an activity *completes*, one token is removed from each of the places directly connected (i.e., not through a gate) to the input of the activity, and one token added to each of the places directly connected to the output of the activity. Table 3 shows the timings associated with the activities present in Figure 6. As can be seen, the activity timing can depend upon the marking of places in the network. For example, activity *bm_hd_xfer* has a deterministic timing which is dependent on the marking of place *bm_out_place*. Although all the timed activities in this submodel have deterministic times associated with them, activity times can be generally distributed.

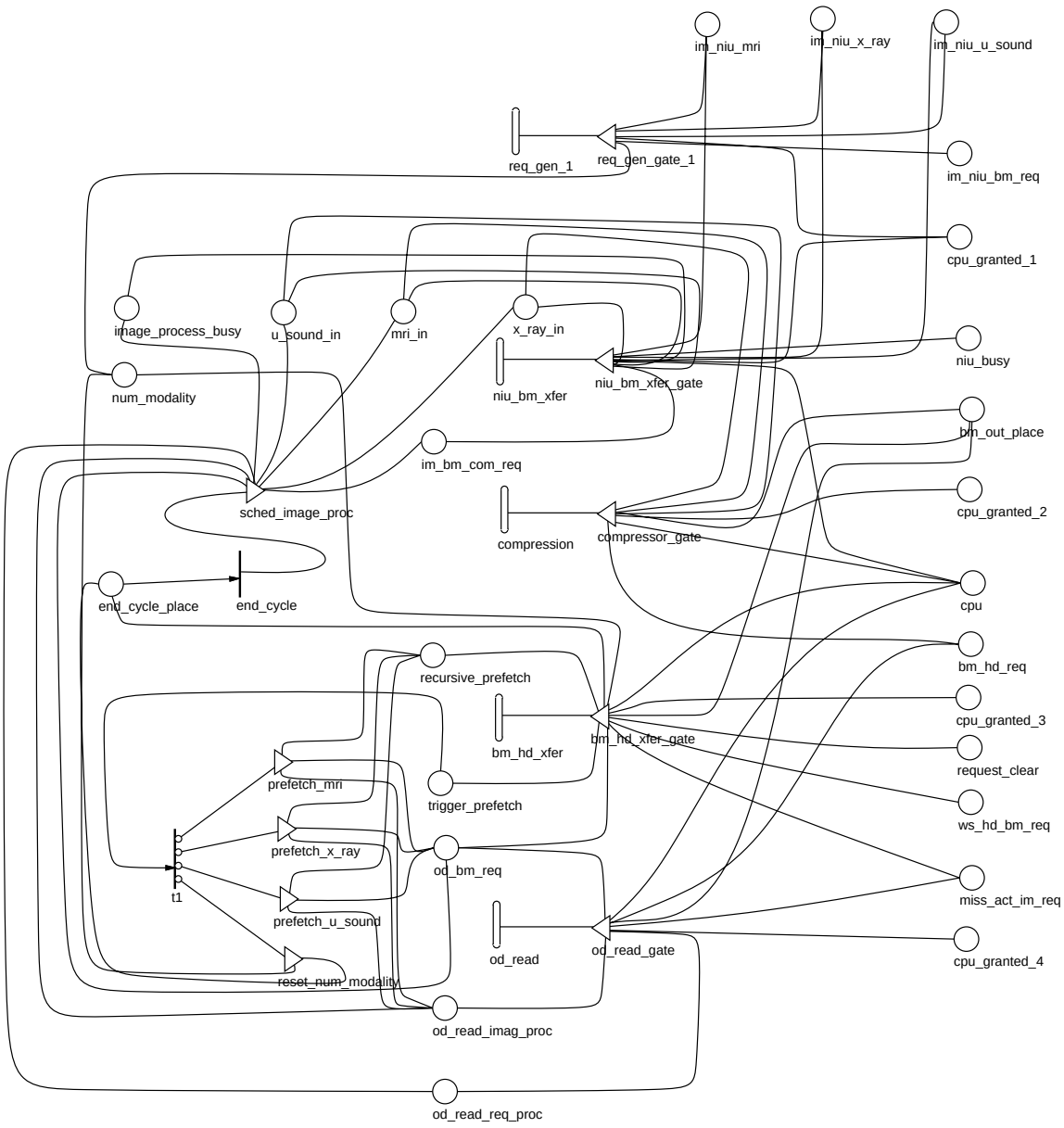


Figure 6: SAN representation of image acquisition and storage operations

Table 3: Activity timings for image processing

Activity	**Distribution**
bm_hd_xfer	deterministic(MARK(bm_out_place) * 0.001)

compression	if((MARK(mri_in))
	return(deterministic(MARK(mri_in) * 0.0004166))
	else
	if(MARK(x_ray_in))
	return(deterministic(MARK(x_ray_in) * 0.0005))
	else
	return(deterministic(MARK(u_sound_in) * 00020833))

niu_bm_xfer	deterministic((MARK(im_niu_x_ray) + MARK(im_niu_u_sound) + MARK(im_niu_mri)) * 0.0005)

od_read	if(MARK(od_read_req_proc))
	return(deterministic(MARK(od_read_req_proc) * 0.0333333))
	else
	return(deterministic(MARK(od_read_imag_proc) * 0.033333))

req_gen_1	deterministic(0.00000000000001)

Table 4: Activity case probabilities in image processing

Activity	**Case**	**Probability**
t1	1	0.2352941
	2	0.1512605
	3	0.11344531
	4	0.5

Cases associated with activities (represented as small circles on one side of an activity) permit the realization of uncertainty concerning what happens when an activity completes. In our example SAN, activity *t1* has four cases. The choice of a particular case decides the size of a folder to be retrieved (prefetched) from the optical disk in the database. As with activity times, case probabilities can be dependent on place markings. Table 4 gives the case probabilities associated with activity *t1*.

Input gates and *output gates* permit greater flexibility in defining enabling and completion rules than with regular Petri nets. In particular, input gates have *enabling predicates* and *functions*, while output gates have only *functions*. The enabling predicate can be any computable predicate (taking on true and false values) of the places connected to it, and, as seen in that which follows, controls the enabling of an attached activity. The function

Table 5: Input gate specification for image processing

Gate	Enabling Predicate	Function

compressor_gate	$MARK(cpu\_granted\_3) == 1$	$if(MARK(mri\_in))$ {
		$MARK(bm\_out\_place) = 158;$
		$MARK(mri\_in) -= 315;$ }
		else {
		$MARK(bm\_out\_place) = 786;$
		$MARK(mri\_in) -= 1572;$ }
		else {
		$MARK(bm\_out\_place) = 1415;$
		$MARK(mri\_in) -= 2830;$ }
		$MARK(cpu)++;$
		$MARK(cpu\_granted\_2)--;$
		$MARK(bm\_hd\_req) = 1;$

associated with each input or output gate describes an action (change in marking) that will occur upon completion of the activity. As with predicates, gate functions can be any computable function on the places connected to the gate.

Activities are *enabled* if there is at least one token in each of the places directly connected to the activity and if the predicate of each connected input gate is true (i.e., *holds*). As an example, consider activity *compression* which has input gate *compressor_gate* connected to it. This activity models the compression of an image folder to half its original size. According to the activity enabling predicate specified in Table 5, the activity is enabled only when the CPU has been allocated for compression work. If the predicate remains true, then the activity will complete, after a time determined by its activity time distribution. On completion, the gate function, as specified in Table 5, will be executed causing tokens to be removed from certain places and/or transferred to other places.

Output gates, together with directly connected output places, are used to specify the action to be taken upon completion of an activity. The function associated with each output gate can be any computable function on the connected places. For example, as shown in Table 6, output gate *od_bytes_1* deposits 786 tokens in place *od_read_place*, corresponding to a compressed X-ray folder of size approximately 7.86 megabytes, and places a request token in *od_bm_req* to schedule a read operation on the optical disk.

Given the brief description of stochastic activity networks above, it is possible to construct models of small systems that permit solution for a specified set of performance vari-

Table 6: Output gate functions for image processing

Gate	Function
$prefetch\_mri$	$MARK(recursive\_prefetch) = 1;$ $MARK(od\_read\_image\_proc) += 158;$ $MARK(od\_bm\_req) = 1;$
$prefetch\_x\_ray$	$MARK(recursive\_prefetch) = 1;$ $MARK(od\_read\_image\_proc) += 786;$ $MARK(od\_bm\_req) = 1;$
$prefetch\_u\_sound$	$MARK(recursive\_prefetch) = 1;$ $MARK(od\_read\_image\_proc) += 1415;$ $MARK(od\_bm\_req) = 1;$

ables. For large-scale systems, a model construction technique known as "reduced base model construction" [10] greatly enhances the suitability of the SAN formalism for modeling systems with many different physical and logical entities with multiple copies of each entity possibly present in the system. This construction technique is highly modular in nature and takes advantage of the symmetries in the overall SAN structure. The reduced base model construction technique has been incorporated into *UltraSAN* [7, 6] and has been used in the modeling of the PACS described in this paper.

To use this approach, a complete (or *composed*) model is built from one or more SAN submodels using "replicate" and "join" operations. The *replicate* operation replicates a submodel a certain number of times. Thus the replicate operation allows one to construct composed models that consist of several identical component submodels. The combination of several different submodels is accomplished using the *join* operation. The join operation produces a composed model which is a combination of the individual submodels. An example of the use of these operations will be given in the next section, where a complete model of the PACS is given.

IV PACS Model Description with SANs

The PACS under consideration has been modeled using composed stochastic activity network models. Modeling with composed models and SANs allows the model to be decomposed into smaller submodels of logical entities, such as a scheduling algorithm for the database CPU, or physical components, such as an imaging modality. There are several

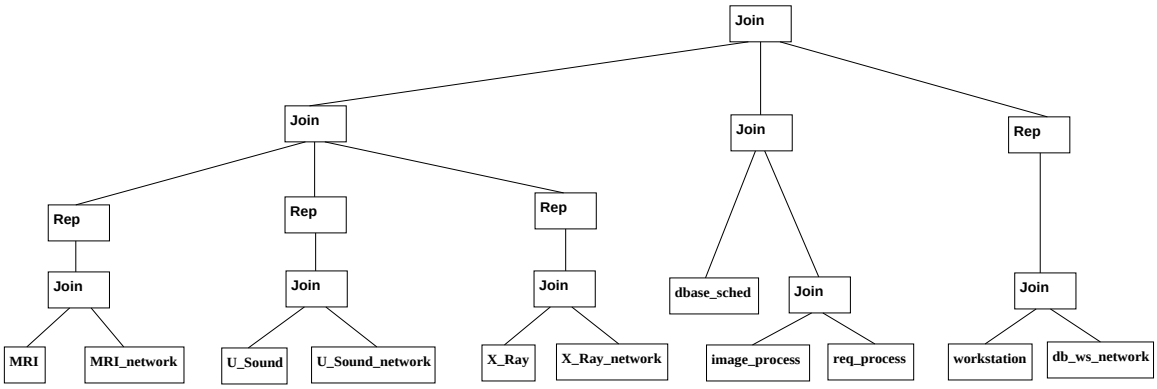


Figure 7: Composed SAN model of PACS of Figure 1

identical units in the system (e.g., X-ray machines), in a PACS, and these can be represented by one SAN submodel replicated as many times as there are units, using the replicate construction operation described earlier. All the SAN submodels of the system are joined together in a hierarchical fashion, using the join operation at appropriate stages in the model building process.

Figure 7 shows the composition of the PACS model using the model construction techniques described earlier. The system model has been divided into three main parts: the imaging modalities, the database and the workstations. There are three different imaging modalities in our model, viz., X-ray, ultrasound and MRI.

SAN submodels for each of these modality types are structurally very similar except that submodel parameters are different for each type. The modality submodel is further split up into the modality operational submodel and the modality network interface submodel. These two submodels are joined to form the modality submodel which is then replicated to represent several units of that modality type. Replicated submodels of different imaging modalities are then joined together to form a common node which connects to the database and workstation submodels using another join at the topmost level in the model hierarchy. The workstation submodel is similarly constructed, building upward from the workstation operational SAN submodel and the workstation network interface SAN submodels. It is indeed this modular nature that has enabled us to reconfigure the PACS with a great deal of ease and thereby investigate its performance under different system configurations.

The database submodel has been split into three SAN submodels: the image acquisition submodel, the viewing request processing submodel and the CPU scheduler submodel.

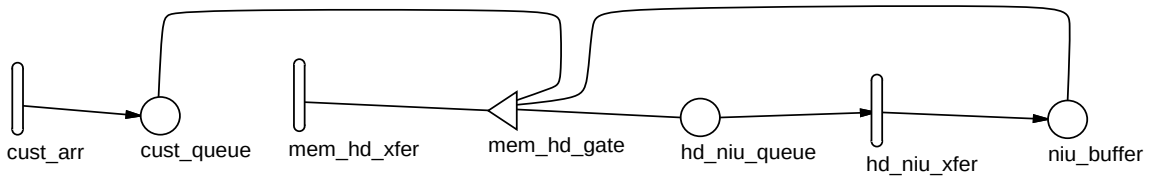


Figure 8: SAN representation of modality operation

The image acquisition submodel represents the activities in the database related to new image acquisition and storage, and the image prefetching algorithm. The viewing request processing submodel describes database activities connected with the processing of image viewing requests arriving from viewing workstations. These submodels are joined together and again joined with the scheduler submodel. All the submodels are linked together by a common join which is the topmost node in the composed model.

In the following subsections, the SAN models of the different PACS components are briefly explained. Important input/output gate specifications, activity time distributions and case probabilities for each of the submodels described below have been included in [5].

A Imaging Modality Submodel

The SAN model of an imaging unit consists of two submodels which are then joined together. They are: an operational submodel and a network interface submodel. Figure 8 is the SAN representation of the operational model. Activity *cust_arr* models the arrivals of imaging jobs (patients) at a modality, while place *cust_queue* represents the queue of these jobs at the corresponding modality. A job corresponds to the generation of a folder of images, with the number of images per folder (and hence per patient), depending on the modality type. For example, from Table 1, the number of images generated per patient for an MRI unit is 32 while those for ultrasound and X-ray units are 36 and 3, respectively. The image folders upon generation are first written to the hard disk and then transferred across the network to the central database archive.

There are two image folder data transfers which use the CPU of a modality: the transfer from the modality RAM to its hard disk (activity *mem_hd_xfer* and place *hd_niu_queue*), and the transfer from hard disk to the network interface memory (activity *hd_niu_xfer* and place *niu_buffer*). The model ensures (by using predicates to inhibit activities) that two activities which use the CPU can not take place at the same time. The CPU is thus not

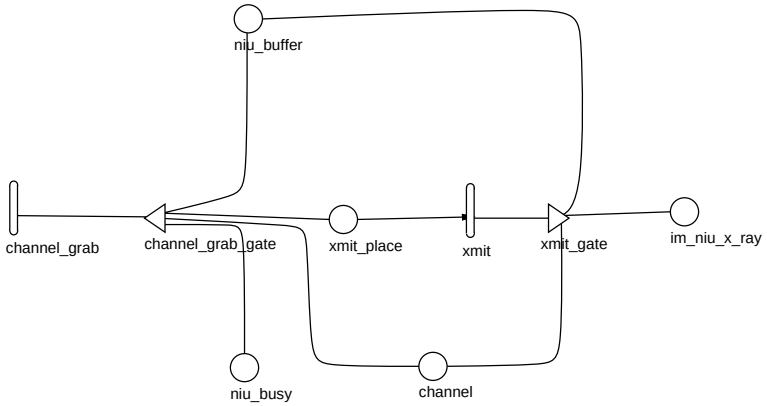


Figure 9: SAN representation of modality network interface

modeled explicitly as a resource in this subnet.

The network interface buffer is represented by *niu_buffer* and is common to all modalities. A new folder cannot be written to this buffer until the previously written folder has been transmitted across the network and deposited at the corresponding buffer on the database side. This is again taken care of by the use of appropriate gate predicates in the SAN model. Finally, the image folder transfer times depend both on the modality hard disk speed and on the size of the particular folder. The former is assumed to be 2MBps while the latter is obtained by multiplying columns 4 and 5 of Table 1.

Figure 9 is the SAN representation of the network interface connecting a modality to the database. Place *niu_buffer* is the modality network interface buffer discussed earlier. The network interface buffer at the database side of the channel is represented by *im_niu_modality* and is common to all modalities of a given type. Place *niu_busy* is used to indicate the availability of the network interface buffer. The presence of a token in this place indicates that the network interface buffer is full. Thus, before attempting to obtain the channel, *channel_grab_gate* checks to see if this buffer is empty. The channel is acquired if this buffer is found to be empty, and if of course, the channel itself is free. The availability of the channel is indicated by the presence of a token in place *channel*. The acquisition of the channel is accomplished by completion of activity *channel_grab*. The completion of *channel_grab* is accompanied by the following changes in the marking of this SAN. A token is removed from *channel* in order to indicate that it is busy. A token is added to each of the places *niu_busy* and *xmit_place*, with the former denoting the non-availability of the network interface buffer and the latter representing the onset of transmission.

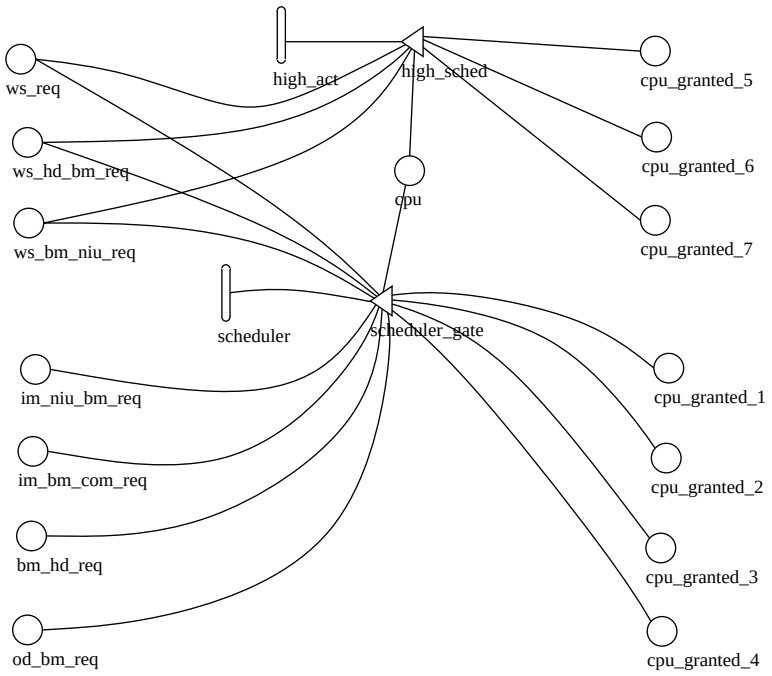


Figure 10: SAN representation of database scheduling algorithm

Once the channel has been allocated, transmission begins (activity *xmit*), and upon completion of transmission, the image folder is written to the network interface buffer (place *im_niu_modality*). The number of tokens placed in this buffer depends on the size of the image folder, which in turn is dependent on the modality type (see Table 1). Due to the large sizes of various image types, a representational unit of 10 KB for every one token is used in the model. Thus, for example, 315 tokens are used to represent an uncompressed MRI folder of size 3145.6 KB. As before, the transmission time for a folder depends upon its size.

B Database Submodel

The database submodel consists of the SAN representations of the database CPU scheduler, the image acquisition and storage submodel, and the viewing request processing submodel. The image acquisition and storage submodel was shown earlier in Figure 6. The scheduler and viewing request processing submodels are shown in Figures 10 and 11, respectively. These SAN representations are joined together, as shown earlier in Figure 7.

The scheduler SAN consists of two activities, *high_act* and *scheduler*, with the former having a higher priority than the latter. Tokens in each of the places located to the left of

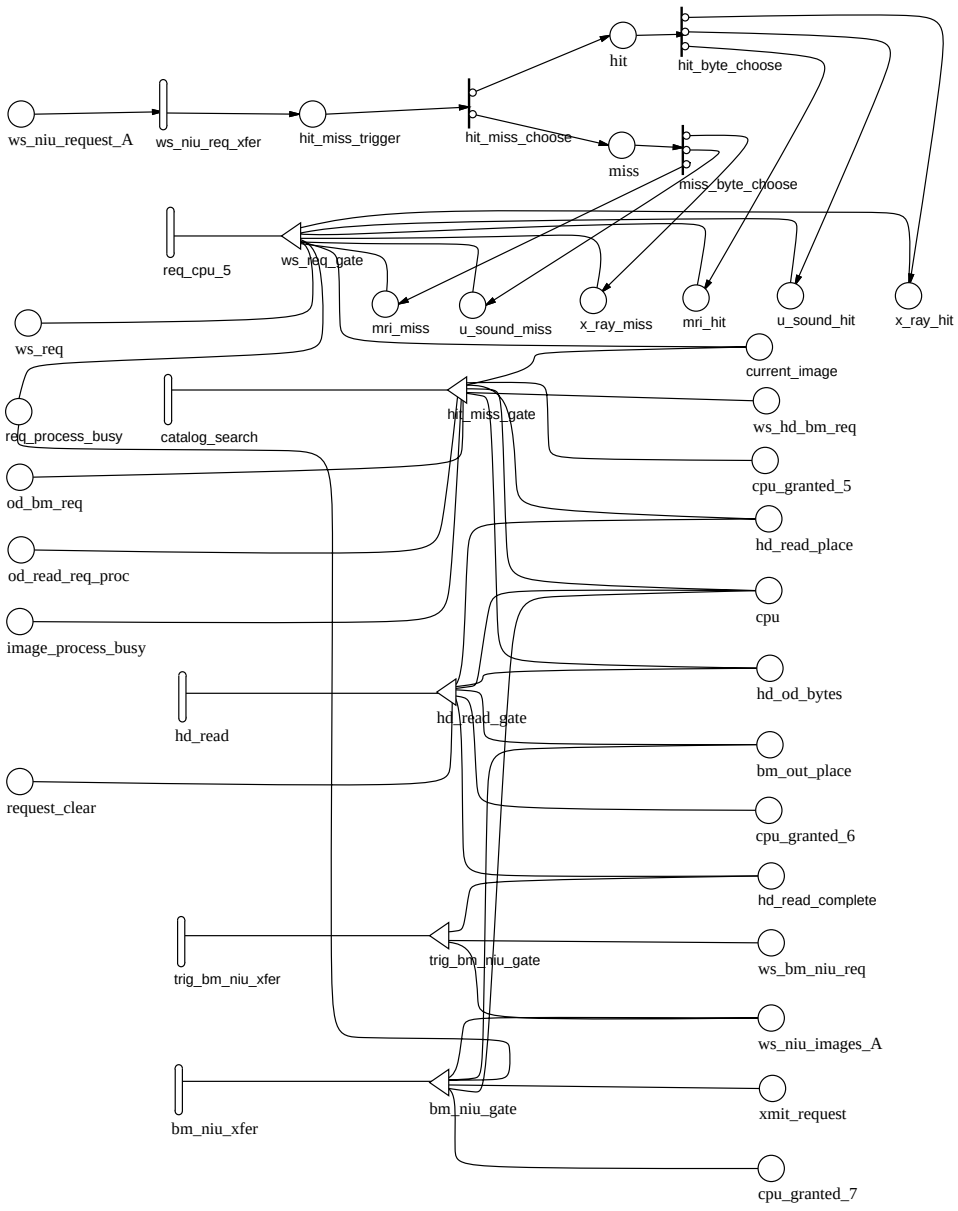


Figure 11: SAN representation of viewing request processing operations

these activities (*i.e.,* places *ws_req* through *od_bm_req*) represent CPU requests while those in corresponding places to the right (places *cpu_granted_1* through *cpu_granted_7*) denote the granting of the CPU for the required operation. The scheduling of activities in Subset A is controlled by input gate *high_sched* while that in Subset B is performed by *scheduler*. A request for the CPU in any of the above mentioned request places, is first checked for by the appropriate input gate, and, based on the priority rules, the CPU is granted by removing a token from place *cpu* and placing it in the appropriate *cpu_granted_number* place. Upon completion of the requested activity (which may be either in Subset A or Subset B), a token is removed from the *cpu_granted_number* place and returned to the place *cpu*.

The image acquisition and storage SAN (Figure 6) is used for the following: acquistion and storage of newly generated image folders, prefetching of subsets of image folders from the stored archive of images, and, fetching of requested images that are stored on the optical disk but are not present on the hard disk. The *im_niu_modality* places represent the database network interface buffer. Recall that this buffer space is common to all modalities, and we can at any given instant, have only one image folder residing in this buffer space. This is taken care of by placing a token in *niu_busy* upon the arrival of a folder to any of the *im_niu_modality* places. Since *niu_busy* is common to all modalities, this ensures that the network interface buffer will never overflow.

The arrival of an image folder is sensed by input gate *req_gen_1*, which then proceeds to request the CPU by placing a token in *im_niu_bm_req*. Places *modality_in* represent the queue of jobs (image folders) within the central database that are waiting to be compressed and archived. The time required for this transfer (time for activity *im_niu_xfer*), is dependent on the image size. The hard disk speed for the database is assumed to be 10 MBps.

Completion of activity *im_niu_xfer* is accompanied by the following actions. First, the contents of the database network interface buffer are transferred to the appropriate *modality_in* place and the CPU is released. This is followed by clearing the *niu_busy* place to indicate the fact that the interface buffer has now been emptied out. Finally, the image processing unit is checked to see if it is free by checking the number of tokens in place *image_process_busy*, which contains one token if any of the activities *compression, bm_hd_xfer*, or *od_read* is busy, and zero otherwise. If it is ascertained to be free, a CPU request to initiate the image compression phase is scheduled, and, a token is placed in *image_process_busy* to indicate that the image processing subunit is now busy.

The time required to compress an image folder depends on the image pixel size and the compression rate. These values are obtained from Tables 1 and 2. In the case of multiple folder types, the *modality_in* places are scanned in the order of their folder size and the first non-empty image folder is chosen for compression. On completion of *compression*, the folder data is compressed to half its original size and moved on to *bm_out_place* and a request is then made for the next activity in sequence, *bm_hd_xfer*.

Activity *bm_hd_xfer* is used to move the compressed folder to the hard disk. Completion of this transfer triggers the image-folder prefetch algorithm which functions in the following manner. First, the the input data size is checked to ascertain whether it is an image folder or just a single image. If it is an image folder, the algorithm proceeds to check if this is the first prefetch, in order to avoid a second prefetch on a single folder. Second prefetches are indicated by the presence of a token in the place *recursive_prefetch*. If this happens to be the first prefetch, then activity *t1* is enabled, which performs a prefetch with a 50% probability. If, on the other hand, a prefetch has been performed, then no more prefetching is done for the current folder, and, the cycle is ended by triggering activity *end_cycle*.

The probability of prefetching a particular image type, as discussed in Section IID, is proportional to its generation rate. Hence, for the present PACS, these probabilities for the MRI, X-ray and ultrasound folders assume the case values listed in entries 1 through 3 respectively of Table 4. The fourth case corresponds to the probability that no image-folder prefetch is triggered. Finally, as the folder sizes differ for the different modality types, the output gates *prefetch_mri, prefetch_x_ray* and *prefetch_u_sound* are used to place different numbers of tokens in place *od_read_imag_proc*, and a CPU request token in *od_bm_req*.

Image requests generated by the workstation units are processed by the viewing request processing submodel which is shown in Figure 11. The arriving image request tokens are first stored in *ws_niu_request*, and, after a short deterministic delay (*ws_niu_req_xfer*) are transferred to place *hit_miss_trigger*. This delay represents the time to transfer a finite size request token from the network interface buffer (*ws_niu_request*) to the buffer space within the database. The size of a request token is taken to be 100 bytes. Activity *hit_miss_choose* is used to determine if the image corresponding to the current request is located on the hard disk (a hit, 0.99 probability, see Table 20) or on the optical disk (a miss, 0.01 probability). The size and hence type of the requested image in both cases (hit or miss) is determined from the case probabilities on their respective activities (*hit_byte_choose* or *miss_byte_choose*). The distribution of these case probabilities is, as mentioned before, proportional to the

image generation rate (refer to columns 3 and 4 in Table 1) and is included in Table 20.

Places *modality_hit* and *modality_miss* represent the queue of jobs (image viewing requests) within this submodel, since all arriving requests are translated into either a hit or a miss and are stored in these modality hit/miss queues. Upon arrival of a particular image request, the following functions are performed by the input gate *ws_req_gate*. First, the request processing submodel is checked to see if it is free, *i.e.,* if none of the activities *catalog_search, hd_read, trig_bm_niu_xfer* and *bm_niu_xfer* are busy. This is done by ascertaining that place *req_process_busy* has no token in it. (A token in *req_process_busy* indicates that this submodel is busy.) If it is found to be free, a token is placed in *ws_req* in order to request for the CPU to perform the catalog search. Next, the *modality_hit* and *modality_miss* places are scanned in an increasing order of their image size, and, a certain number of tokens corresponding to the first non-empty image request is placed in *current_image*. Finally, a token is placed in *req_process_busy* to indicate that this unit is now busy processing a request.

After the completion of catalog search (activity *catalog_search* with a time of 1 ms), the following set of actions are performed. If a hit has occurred, then the image may be fetched from the hard disk, which results in a scheduling request for activity *hd_read*. If on the other hand the image was not located on the hard disk cluster, a miss has occurred. The requested image therefore needs to be fetched from the optical disk. This is indicated by placing a certain number of tokens (corresponding to the size of the desired image whose type is stored in *current_image*) in *od_read_req_proc* and scheduling a CPU request for an optical disk read. However, since optical disk reads are performed by the image processing unit, and, as it could be busy currently, place *image_process_busy* is first checked to see if it contains a token. If it does not (which implies that the image processing subunit is currently free), a scheduling request for activity *od_read* is made by placing a token in *od_bm_req*. If on the other hand the image processing subunit is found to be busy, no immediate request for the CPU is made. In this case, place *od_read_req_proc* contains tokens which also serve as an indicator to the image processing unit to process this request as soon as it is done processing its current job.

From the discussion above, we observe that activity *hd_read* may be triggered under two circumstances, namely, during a hit, or, during a miss. In the former case, however, it could be made to complete with a smaller delay (a delay corresponding to the hard disk read of the corresponding image), while in the latter case it would need to wait until the optical disk read is complete (indicated by the presence of a token in *request_clear*), and,

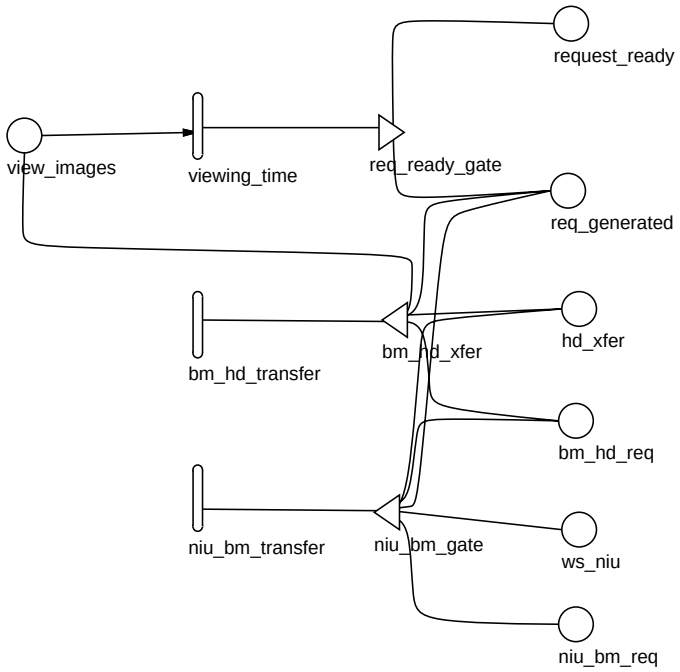


Figure 12: SAN representation of workstation operation

the image is fetched and placed into *hd_od_bytes*. The sequence of actions after this is common to both the hit and miss cases. A token is placed in *hd_read_complete* to indicate the completion of a hard disk read. Activity *trig_bm_niu_xfer* is used to admit one image at a time to the transmission buffer *ws_niu_images*. It is also used to place a CPU request to transfer the image from the database buffer memory to the transmission buffer. The time to accomplish the above mentioned transfer (database buffer memory to transmission buffer) is represented by *bm_niu_xfer* and is dependent on the size of the requested image (see Table 19).

C Workstation Submodel

The SAN representation of a workstation is shown in Figure 12. Activity *viewing_time*, as its name denotes, represents the time taken to view the set of requested images. An exponential viewing time and a set of four images per request set have been assumed. Output gate *req_ready_gate* is executed upon completion of activity *viewing_time* and deposits four tokens in each of the places *request_ready* and *req_generated*. Activity *niu_bm_transfer* represents the time required to transfer the requested image from the transmission buffer to the

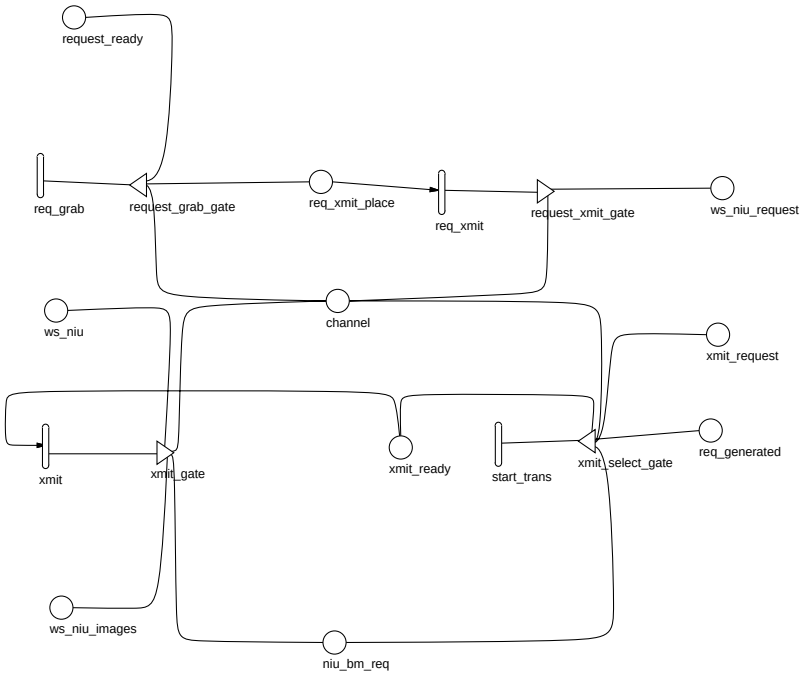


Figure 13: SAN representation of workstation database network interface

block memory in the workstation, while *bm_hd_transfer* denotes the time to copy this image on to the hard disk. Since both these activities need to use the CPU of the workstation, the model ensures, via the use of indicator places *niu_bm_req* and *bm_hd_req*, that these two activities refrain from using the CPU simultaneously. The transfer times (*niu_bm_transfer* and *bm_hd_transfer*) are marking dependent on the image size and a workstation hard disk speed of 2 MBps has been assumed. The number of tokens in *req_generated* represent the number of outstanding images within the current image set, and, is decremented by one (by *niu_bm_gate*) each time an image is sent to the corresponding workstation. Finally, after checking to see if all the four images in the given image set have been received (zero tokens in *req_generated*), *bm_hd_xfer* deposits a token back in place *view_images*.

The SAN representation of the network interface between a workstation and a database is shown in Figure 13. The top half of this figure represents the set of places and activities involved in the transmission of an image request token from the workstation to the database, while the bottom half is used to represent the places and activities instrumental in transmitting the retrieved image back to the workstation. The workstation deposits its image request token in *request_ready*, which is then transmitted across the channel to the

request processing unit of the central database. The image, upon retrieval from the database, is stored in the transmission queue buffer place *ws_niu_images*. The channel access protocol is the same as that used for transmitting image folders to the database from the modalities. The transmission time (*xmit*) is dependent on the image size. Completion of image transmission is marked by transferring the contents of *ws_niu_images* over to the workstation interface buffer *ws_niu*.

V Results

The described models were simulated using *UltraSAN*, to estimate various measures of system performance. In particular, we have determined, for the case of two different modality (workload) mixtures and each with a variety of system loads, estimates of the utilizations of the various resources in the system. We have also estimated performance variables (e.g., queue lengths at various points) which are used to determine the response time to a viewing request. All the performance variables specified in the simulations were estimated to a 95% level of confidence, and error bars are given for each result plotted (in many cases, these error bars are too small to be seen). The system was started cold, i.e. empty, and the initial portion of each run was discarded to avoid using the transient period in calculating the estimator. The batch size used was made large enough to assume independence between batch means, and a large enough number of batches was used to avoid problems due to non-normality of the batch samples.

Variation of the system load was accomplished in two different manners. In one case, the number of workstations was fixed at 15 and the viewing time of the images was varied, while in the other case, the average viewing time was fixed at 90 seconds and the number of workstations was varied. The patient arrival workload at all imaging modalities was kept fixed, at levels suggested in [9] and given in Figure 1. Hardware parameters, such as disk access times and bus speeds, were also fixed at reasonable values. For the case with differing modality mixtures, the following two types were used. In the Type-I mix, all three modalities, viz., the MRI, ultrasound and X-ray units were included. In the Type-II mix however, the modality with the highest resource demand (X-ray) was retained while the others were turned off (MRI and ultrasound).

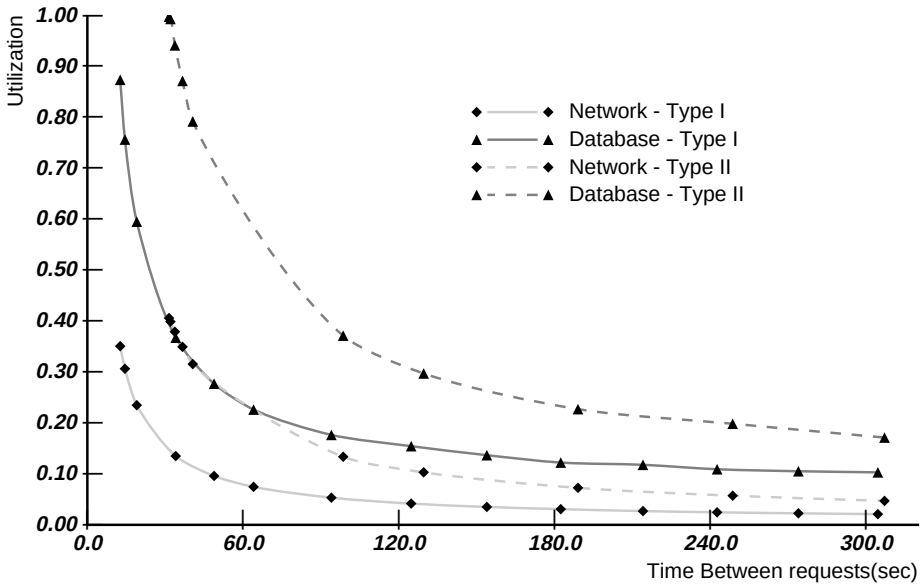


Figure 14: Database CPU and network utilizations as a function of viewing time

A Component Utilizations

Figure 14 gives the utilization of the network and the database CPU versus viewing time of a set of images with 15 workstations. Figure 15 illustrates the above mentioned utilizations versus the number of viewing workstations for an average viewing time of 90 seconds.

The figures show that the network utilization is fairly low for all loads, thereby indicating that the network channel is not the bottleneck in the present PACS environment. The database utilization indicates the percentage of time the database CPU is busy. From the database utilization curves it is seen that the utilization increases rapidly for higher loads (small viewing times) for both types of modality mixtures. This indicates to us that the database CPU is indeed the bottleneck in the PACS. We also observe that for a given viewing time, both the database and channel utilizations are higher for the case with the Type-II mix than the case with the Type-I mix due to the fact that in the former case we are always requesting the maximum sized image (an X-ray image), while in the latter this may not always be the case. In fact, for high system loads (viewing times less than 60 seconds), the database utilizations for both types of workload mix begin to rise rapidly as expected,

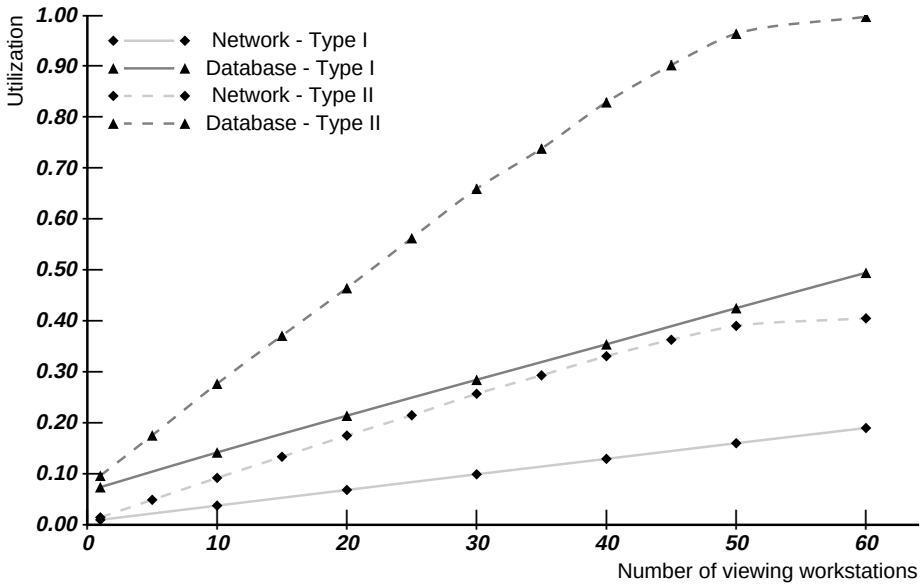


Figure 15: Database CPU and network utilizations as a function of the number of viewing workstations

but the rise for the Type-II mix is much steeper. An estimate of these values would prove to be very helpful to a PACS designer, especially in light of widely differing system loads. Finally, for low system loads (viewing times greater that 300 secs), the database utilization tapers off to a constant 11% and 18% for the case of Type-I and Type-II modality mixtures respectively. This utilization is due to the load of storing new folders from modalities, and the subsequent execution of the prefetching algorithm.

Figures 16 and 17 demonstrate the hard disk and optical disk utilizations in the database. As in the previous case, Figure 16 has been used to illustrate the above mentioned utilizations versus viewing times for 15 workstations, while Figure 17 depicts these utilizations versus the number of workstations with a fixed viewing time of 90 seconds. From the above figures, it is observed that the hard disk utilization is lower than the optical disk utilization for low loads (large viewing times or a small number of workstations). However, as the system load begins to increase, a crossover occurs (at about 8.0% for the Type-I mix and 9.0% for the Type-II mix), after which the hard disk utilizations exceed those of the optical disk with the difference in magnitude becoming very large for high loads. This sharp disparity at high loads may be attributed to the higher number of read operations on

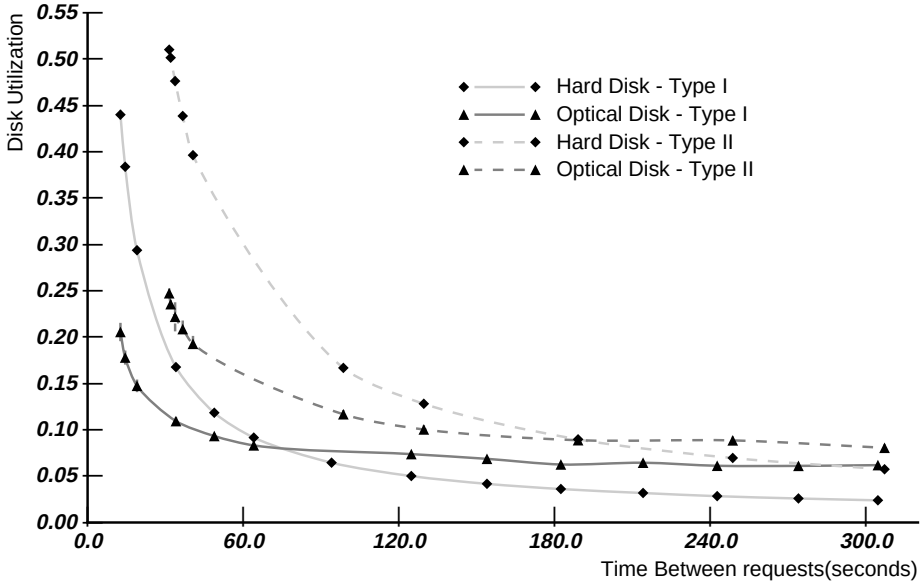


Figure 16: Disk utilizations in database as a function of the viewing time

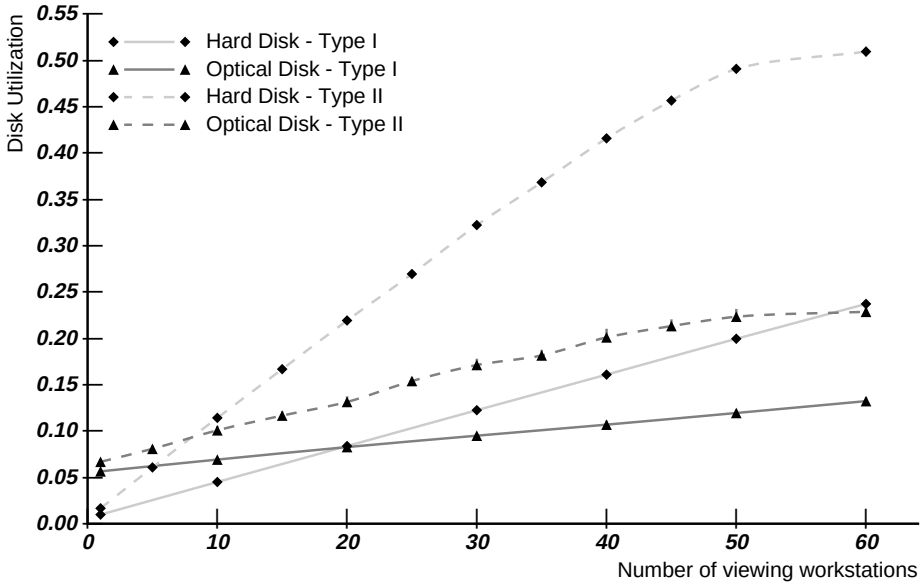


Figure 17: Disk utilizations in database as a function of the number of viewing workstations

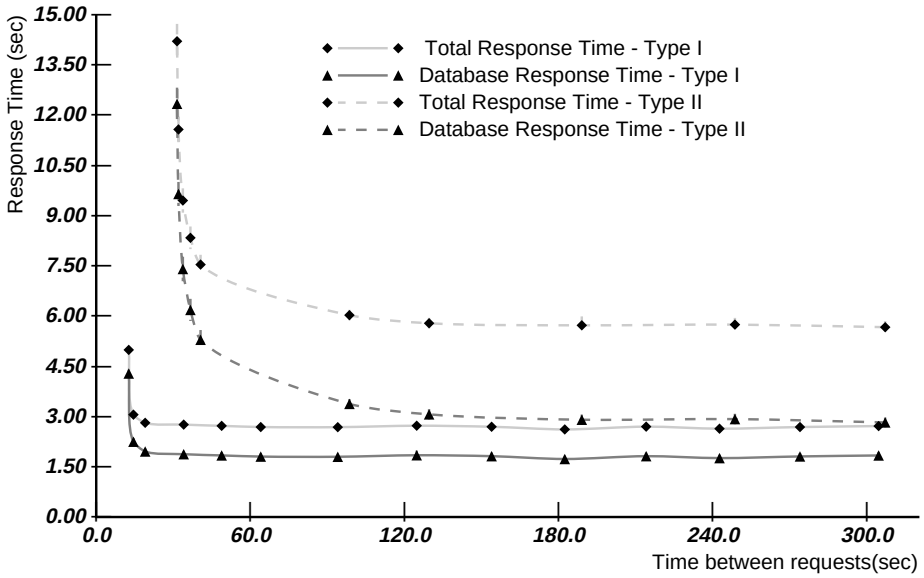


Figure 18: Response time vs viewing time

the hard disk, whereas the optical disk is not influenced so strongly (the miss probability for a request was taken to be .01). The prefetch algorithm would affect the optical disk utilization at higher loads since it would change the miss rate. For lower loads, the hard disk shows very low utilization whereas the optical disk utilization tapers off to around 6% and 8% for the two modality mixes. Again, this is because of the prefetch algorithms employed in the database and because the optical disk access times are much higher than hard disk access times. Finally, as observed before, the disk utilizations for a given load are higher with the Type-II mix than with the Type-I mix.

B Response Time Calculation

Figure 18 reports the total response time and the database response time for viewing requests as a function of the image set viewing time, while, Figure 19 illustrates the same set of response times as a function of the number of workstations.

The "total response time" in these figures is defined to be the time between the generation of a request for four images and the writing of the first image on the workstation hard disk. Similarly, the "database response time" is the portion of the total response time due to database processing. There are five delay components which contribute to the total

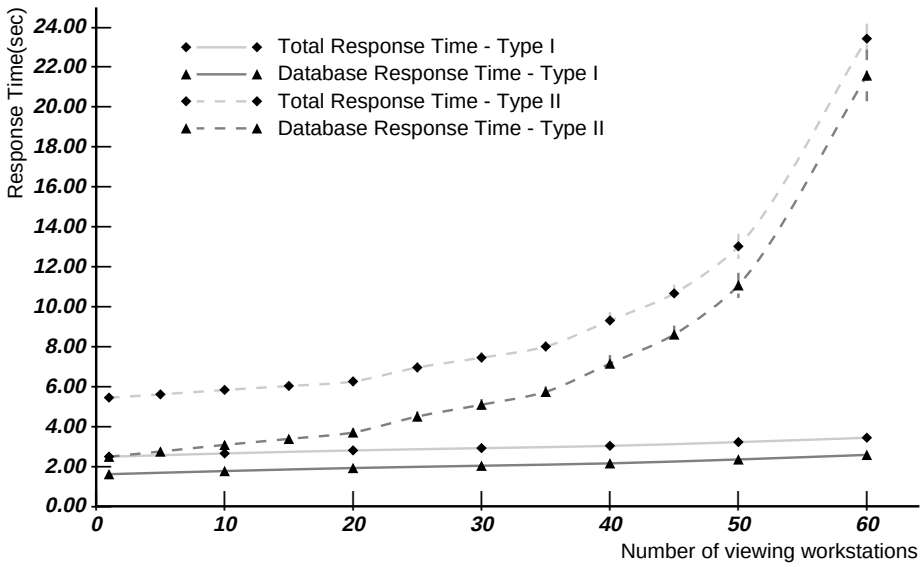


Figure 19: Response time vs number of viewing workstations

response time:

1. Delay in the request queue at workstation.
2. Transmission delay for a request.
3. Delay at the database due to retrieval time.
4. Transmission delay for the retrieved image.
5. Delay due to writing the image to hard disk at the workstation.

The above mentioned delay components may be calculated using Little's result [11], as shown in Figure 20. Boxes B through E are seen to contain a set of places that are present in the SAN model of the PACS. These places represent either a queue or a server. The arrival rate of tokens into all places in a box is the same. Little's result can be applied to each individual box and the total expected delay can be obtained by summing these partial delays. The rate in box A is the rate at which the viewing images are requested. Since each request token in box A represents a request for 4 images, the effective rate into the other boxes B through E, is four times this rate. Box B is used to compute the delay involved in

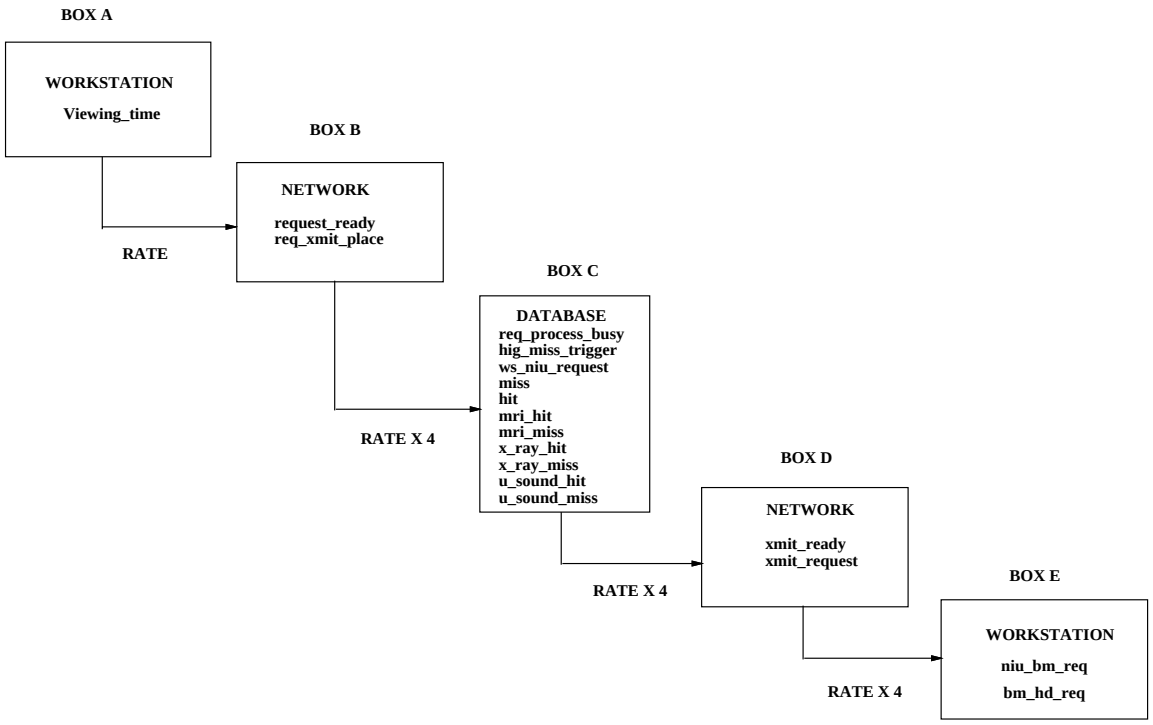


Figure 20: Response time calculation using Little's result

the transmission of a request token. The delay in the database is calculated with the help of box C, while boxes D and E respectively compute the image transmission delay and the delay in the workstation after the image has been received.

From the total response time curve, we observe that the delay remains fairly low and constant under a wide range of loads for both types of modality mixtures. In fact, for the Type-I mix (wherein we have all the three types of modalities present), the response times are pretty low up and until a viewing time of just 10 seconds between requests, indicative of a fairly long "stable operation portion" with a low and constant delay. The delays for Type-II mix however are larger than their counterparts for the Type-I mix, which, as observed before, is due to the fact that *all* the images requested in this set require a considerable (in fact, the maximum) amount of system resources. Furthermore, the "low and constant" portion of the delay curve in this case is seen to break away at around the 60 seconds zone, which is earlier than the Type-I mix case. Thus, depending on the modality mix, an estimate of the total delay for the given PACS at low system loads is seen to be around 2.6 and 5.6 seconds for the two widely differing modality mix cases studied in the current work.

The stable operation is seen to extend to 10 seconds in one case and 60 in the other, after which the delays, under the increasing system loads, begin to rise sharply just as expected. Thus the ideal operating region for the PACS would be in the flat portion of the response time curve. The database response time curve essentially follows that of the total response time, indicating that the database is causing the increase in response time for higher system loads. The reason for this is evident from the utilization curves for the database which were discussed earlier. Clearly, a faster database CPU and faster hard disk access times in the database would improve the performance of the PACS.

These models and results thus show that SANs provide a suitable modeling framework to evaluate the performance of PACS. Heirarchical (composed) models of the various PACS components may be constructed, and the performance of the composed model may be evaluated and bottlenecks identified. The effect of varying system paramters on the overall performance of a given PACS configuration may also be examined, as was done in the current work. Furthermore, while the assumptions made in modeling this particular PACS do not apply to all PACS, they illustrate how to construct SAN models of PACS given assumptions reasonable for a particular installation. Different assumptions would lead to different SAN models, without a great change in the complexity of the model. Hence, this work shows how SANs and composed models can be used to construct and evaluate a particular PACS configuration as dictated by the application on hand, along with the desired system parameters.

VI Summary

Picture archiving and communication systems are expected to have a high impact on the efficiency of hospital management systems. Currently, most of the radiological image processing in hospitals is manual, which limits the speed and efficiency of processing. Upgrading to a completely automated electronic system would be an expensive process in terms of equipment and software costs. Thus, the modeling and simulation of PACS is a relatively inexpensive approach to obtaining performance estimates for different system configurations prior to actually upgrading the hospital with a PACS. These performance estimates help trim down the system size and hence the cost. In addition, they can help in identifying system bottlenecks so that more effort can be channeled into reducing the bottlenecks.

For the particular PACS studied in this paper, simulation studies have indicated that the performance bottleneck in the system is the database and that the database and overall performance can be improved by having a faster database processor and hard disk clusters with faster access times. Performance curves were obtained for component utilizations in the database (hard disk and optical disk), and, for response times to image viewing requests made at the workstations. Finally, the performance of the PACS system was examined under the influence of a widely differing modality mix including the performance of the system in the presence of a single modality type with the heaviest workload demand.

The results and models developed also illustrate that stochastic activity networks are a modular and efficient modeling formalism which can be effectively used to evaluate PACS. PACS tend to be large with many different types of imaging modalities, databases and workstations, and many units of each type. Reduced base model construction makes the modeling process for such systems with identifiable physical and logical subunits easy. Large models can be constructed in a hierarchical fashion using the construction operations discussed in this work.

REFERENCES

[1] W. J. J. Stut, Jr., M. R. van Steen, L. P. J. Groenewegen et al.: Picture archiving and communication system design issues: The importance of modelling and simulation. Journal of Digital Imaging 3:246-253, November 1990

[2] A. R. Bakker, H. Didden, J. P. J. de Valk et al.: Considerations on an algorithm for activation of images in a multilayered storage system within a PACS. Proc. SPIE PACS IV 626, 1986

[3] A. Movaghar and J. F. Meyer: Performability modeling with stochastic activity networks. Proc. 1984 Real-Time Systems Symp., Austin, TX, Dec. 1984

[4] J. F. Meyer, A. Movaghar and W. H. Sanders: Stochastic activity networks: structure, behaviour and application. Proc Int. Workshop on Timed Petri Nets, Torino, Italy 106-115, July 1985

[5] W. H. Sanders, L. Kant, and A. Kudrimoti: A modular method for evaluating the performance of picture archiving and communication systems. Performability Modeling Research Laboratory Technical Report, The University of Arizona, January 1993

[6] W. H. Sanders and R. S. Freire: Efficient simulation of hierarchical stochastic activity network models. To appear in Discrete Event Dynamic Systems: Theory and Applications

[7] J. Couvillion, R. Freire, R. Johnson et al.: Performability modeling with *UltraSAN*. IEEE Software 8:69-80, September 1991

[8] W. H. Sanders, R. Martinez, Y. Alsafadi et al.: Performance evaluation of a picture archiving and communication network using stochastic activity networks. To appear in IEEE Transactions on Medical Imaging

[9] American Hospital Association Hospital Technology Series (ISSN 0888.711X), Guideline Report, Copyright 1989

[10] W.H. Sanders and J. F. Meyer: Reduced base model construction methods for stochastic activity networks. IEEE Journal on Selected Areas in Communications 9:25-36, January 1991

[11] J. D. C. Little: A proof of the queueing formula $L = \lambda W$. Operations Research 9:383-387, 1961