Multimed Tools Appl (2011) 51:99–132
DOI 10.1007/s11042-010-0627-7

QoS and resource management in distributed interactive multimedia environments

Klara Nahrstedt · Ahsan Arefin · Raoul Rivas · Pooja Agarwal · Zixia Huang · Wanmin Wu · Zhenyu Yang

Published online: 28 October 2010

Abstract Quality of Service (QoS) is becoming an integral part of current ubiquitous Distributed Interactive Multimedia Environments (DIMEs) because of their high resource and real-time interactivity demands. It is highly influenced by the management techniques of available resources in these cyber-physical environments. We consider QoS and resource management influenced by two most important resources; the computing (CPU) and networking resources. In this paper, we survey existing DIME-relevant QoS and resource management techniques for these two resources, present their taxonomy, compare them, and show their impacts on DIMEs. Finally, we discuss appropriateness of those techniques in a sample DIME scenario.

Keywords Quality of service · Resource management · DIME · Bandwidth management · Delay management

K. Nahrstedt · A. Arefin (✉) · R. Rivas · P. Agarwal · Z. Huang · W. Wu
Department of Computer Science, University of Illinois at Urbana-Champaign,
201 N. Goodwin Avenue, Urban, IL 61801–2302, USA
e-mail: marefin2@illinois.edu

K. Nahrstedt
e-mail: klara@illinois.edu

R. Rivas
e-mail: trivas@illinois.edu

P. Agarwal
e-mail: pagarwl@illinois.edu

Z. Huang
e-mail: zhuang21@illinois.edu

W. Wu
e-mail: wwu23@illinois.edu

Z. Yang
School of Computing and Information Sciences, Florida International University, 11200 SW 8th Street,
Miami, FL 33199, USA
e-mail: yangz@cis.fiu.edu

1 Introduction

Distributed Interactive Multimedia Environments (DIMEs) are becoming ubiquitous cyber-physical spaces, where we work and play on regular basis. Examples of DIMEs are tele-presence environments [4, 21], and multi-player gaming spaces [18]. More recently, we are seeing even more advanced tele-presence and gaming DIME environments as new 3D sensory, actuating and display devices are becoming available and cost-effective [77]. Examples are 3D tele-immersion environments [79] and 3D games.

Several major problems of the advanced DIME environments are becoming clear. First, the **scale** of the sensors and actuators in each of the cyber-physical spaces connected via high-performance networks is increasing due to availability and cost effectiveness. We are seeing rooms with ten's of 3D cameras (see Fig. 1), microphone arrays, body sensors, and multiple displays that surround physical spaces for dance, exercise, and/or physical games. Second, **real-time interactions and real-time guarantees in distributed settings** are in high demand due to tasks interactivity among users and devices located across geographically distributed DIME spaces (see Fig. 2).

To solve the above problems, we need to carefully **organize** end-to-end computing and networking resources via appropriate local and distributed services such as resource admission, allocation, scheduling, adaptation services as well as **parameterize** the performance metadata (called Quality of Service parameters) that describe the DIME resources services. Over the last 20 years, there has been a large body of related work on *Quality of Service (QoS)* and resource management of multimedia systems and networks for distributed multimedia applications such as video-conferencing, video-on-demand, and others (e.g., [49, 50]). In this paper, we revisit, survey and evaluate the existing QoS and resource management techniques for the advanced cyber-physical DIME environments that are emerging. Our goal will be (a) in Section 2 to provide readers with our DIME assumptions and the corresponding taxonomy and comparison criteria, (b) in Section 3 to survey and compare DIME-relevant resource management techniques across two most important resources, the computing (CPU) and networking resources, and (c) in Section 4 to evaluate and discuss appropriateness of techniques in example DIME scenarios. Section 5 will conclude the paper with further directions of this area.

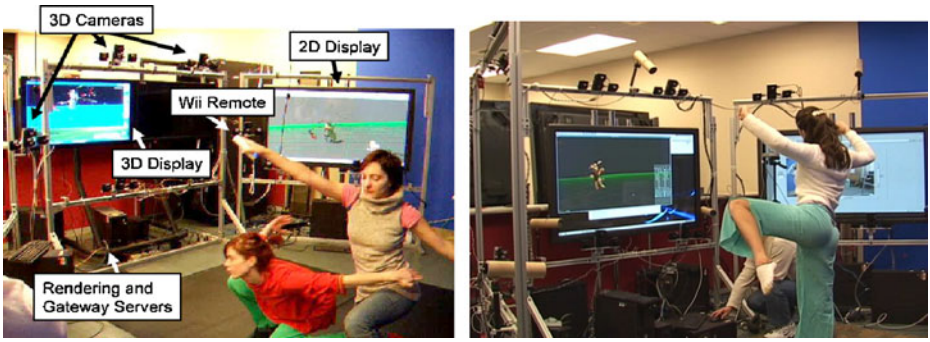

Fig. 1 Cyber-physical spaces/rooms with large scale of interactive sensors and actuators

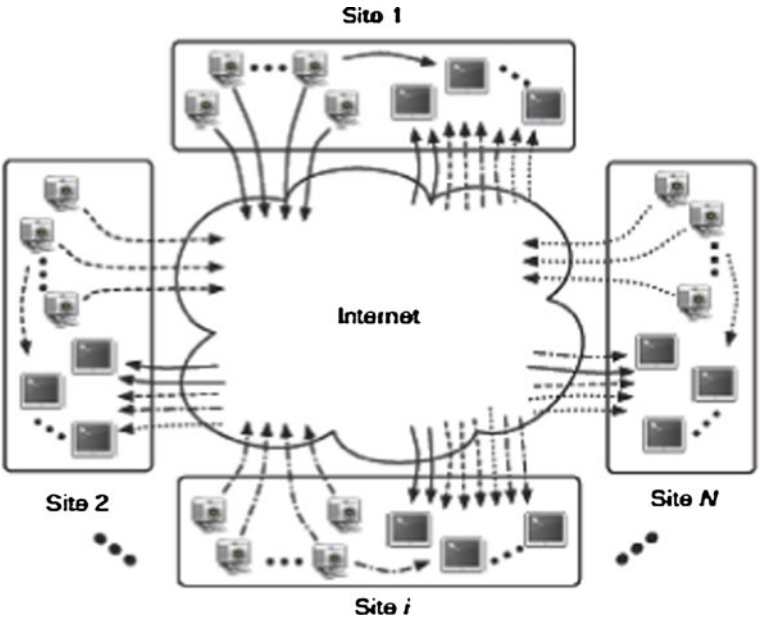

Fig. 2 Multi-site 3D DIME infrastructure. Each site includes multiple cameras and displays, all connected via LAN. All sites are connected via WAN Internet

2 DIME assumptions, taxonomy and comparison criteria

2.1 DIME assumptions

Large scale of sensory and actuating devices per site (cyber-physical space), real-time correlated streams, real-time and synchronous guarantees for multi-modal data delivery and interactive distributed communication are *key characteristics* of DIMEs [77]. Users not only interact with their own cyber-physical environment that surrounds them (as shown in Fig. 1), but also with other remote users and their cyber-physical environments through multiple heterogeneous communication channels. Usually these types of multi-site geographically distributed cyber-physical DIME spaces as shown in Figs. 1 and 2 are organized from the computing and network infrastructure point of view as follows: each cyber-physical site consists of several local computing nodes (LN), that support digital sensing (input cameras, microphones, body sensors) of the physical space (e.g., users) as well as digital actuation/display (output displays, haptics) of information to the physical space (e.g., users). The local computing nodes LN at each site connect to an aggregation point, called rendezvous point (RP), via a local area network (LAN). *Wide Area Network (WAN)* Internet communication occurs among remote cyber-physical sites. The communication among sites happens via multiple logical channels through RPs. Hence, RPs represent a peer-to-peer overlay network over which real-time interactions and guarantees must be delivered. Figure 3 shows an example of five sites DIME, and how their computing (LN/RP nodes) and networking (logical connections among LN/RP nodes) resources are organized.

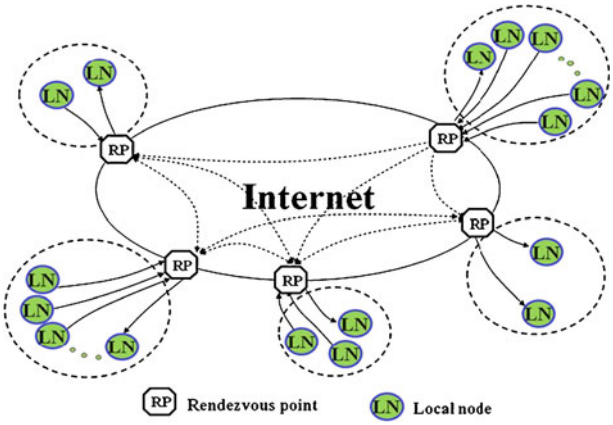

Fig. 3 Architectural model of distributed interactive multimedia environment

DIME architecture can be also modeled as a multi-tier architecture, where (1) the LN and RP nodes, capturing and sending multi-streams,[1] represent the capture tier at the sender-side, (2) the LN and RP nodes, displaying and rendering streams, represent rendering tier at the receiver-side, and (3) the protocols between LN and RP nodes, transmitting stream data, represent the transmission tier.

2.2 Taxonomy of DIME resources

The two most important resources that DIMEs need to carefully manage are (a) the **computing resource** CPU and (b) the **network resource** within each local site (LAN resources), and across multiple sites (WAN resources).[2]

CPU resource is crucial for DIME systems at the LN and RP nodes since it needs to process various sensing streams in real-time as well as schedule different interactive media and control tasks in real-time manner. The CPU resource can become bottleneck very quickly at the LNs and RPs due to the competition among large scale of devices even though they generate correlated tasks/streams. If this resource is not carefully managed, real-time interactivity violations, temporal degradation of sensory media and tasks delays start to happen very early in the DIME end-to-end communication. To achieve real-time guarantees, high real-time interactivity, and high temporal quality among the multi-modal sensory information, we need to deploy multimedia-aware best effort CPU schedulers or multimedia-specific soft-real-time CPU schedulers. We will survey and compare various CPU schedulers in Section 3.1.

Network resource is crucial for DIME systems (between LN and RP nodes) since large amount of multi-modal data needs to be exchanged in real-time. The network resource among the LNs and RPs can become bottleneck very quickly due to the large scale of networked devices in DIME, high number of competing flows, and high traffic rate demands. Often to remedy the bottleneck, network service providers upgrade the routers, links and offer higher bandwidth and throughput to RPs. However, this is not sufficient for DIME applications since large bandwidth and throughput do not prevent *head-of-line* blocking effects of the multi-modal sensory traffic, hence yielding undesirable communication

[1] We will use streams and flows notations interchangeably throughout the paper.
[2] Another resource would be storage, but since we are discussing distributed interactive systems, networking and CPU are of higher importance.

delays. Moreover communications between RPs are done over unreliable Internet. Therefore, for DIMEs we need to consider bandwidth and delay management concurrently and carefully. In bandwidth management, to achieve real-time guarantees, high real-time interactivity and high spatial and temporal media quality, we need to deploy network resource allocation schemes based on reservation and/or adaptation principles. In delay management, to achieve real-time interactivity and guarantees, we need to deploy per flow packet/message scheduling techniques over queues in the network protocol stack as well as buffer control and stream synchronization. We will survey and compare various bandwidth and delay management techniques in Section 3.2.

2.3 DIME evaluation/comparison criteria

As we alluded above, DIME resources and their services (reservation, adaptation, scheduling, buffering, and synchronization) will be evaluated according to their performance such as delay they induce or bandwidth they demand. Performance levels of services are described and represented by Quality of Service and its metrics. We will consider two major metrics: (a) **bandwidth** and (b) **delay**, and where appropriate also their related metrics, **packet loss**, **jitter**, and **skew**.

Bandwidth Though the term bandwidth is usually used in networking applications, here we classify bandwidth in two classes: network bandwidth and CPU bandwidth.

- Network Bandwidth: Network bandwidth refers to the bandwidth capacity or available bandwidth in bit/s, which typically means the net bit rate, channel capacity or the maximum throughput of a logical or physical communication path in a digital communication system.
- CPU Bandwidth: CPU bandwidth refers to the available processing power in cycle/s, which typically means the net processing rate, net utilization and maximum throughput of CPU tasks.

Network bandwidth is important for high quality data transmission among DIME LNs and RPs to ensure high quality of spatial and temporal multi-modal sensory delivery. CPU bandwidth is important for high task throughput and high quality spatial and temporal media processing at all computational nodes in the system. The major goal of bandwidth management is to achieve high throughput of tasks at the end hosts and high throughput of packets in the network. This is necessary due to the high scale of sensory data as well as high sampling rate of sensory and actuating devices during running DIME sessions. Different kinds of network admission control and reservation protocols have been proposed to manage network bandwidth in current Internet. On the other hand, many CPU admission and scheduling algorithms at the computational nodes have been explored to manage CPU bandwidth as we will survey in Section 3.

Delay Each DIME computational or communication task and each packet/message processing along the end-to-end path(s) between LNs and RPs take time and cause timely overhead, called delay. DIME considers three types of delays: sender-side delay, network delay and receiver-side delay.

- Sender-side delay (D_{sender}): We define sender-side delay as the sum of computing delays incurred by input computing devices (LNs at the sending site) and intermediate node(s) (RP at the sending site), and the communication delays incurred by traffic from

input LNs to RP. Computing delays caused by stream acquisition and encoding tasks are main part of the sender-side delay in DIMEs. Video capturing and computing (including 2D motion detection and 3D reconstruction) and audio capturing are some of the sources of data acquisition delays.

- Network delay (D_{net}): *Metropolitan Area Network (MAN)* and WAN transmission delays among RPs in the current Internet are the sources of network delay.
- Receiver-side delay ($D_{receiver}$): Receiver-side delay is defined as the sum of computing and communication delays at the receiving site. Computing delays happen in the output devices (rendering displays) and communication delays are incurred by the traffic from RP to output LNs. Rendering and decoding tasks are the main sources of receiver-side delay.

The end-to-end delay (D_{e2e}) is the sum of these three types of delay in the system, i.e.,

$$D_{e2e} = D_{sender} + D_{net} + D_{receiver}$$

The overall goal for DIMEs is to achieve D_{e2e} small enough for each packet of each stream so that interactive communications among remote sites is possible.

Packet loss Another QoS metric that affects the performance of DIME system specially the audio quality is the packet loss. Packet loss is quite common in current Internet using DCCP and UDP type transport protocol. Lost packet might mean a missing frame. Packet loss can be caused by a number of factors, including signal degradation over the network medium due to multi-path fading (in wireless channel), corrupted packets rejected in-transit, packet drop because of channel congestion, faulty networking hardware, faulty network drivers or normal routing routines. Packet loss in DIME applications may create sudden silence in audio communication and void or still screen in case of video transmission. Some applications try to mask packet loss by forward error correction (e.g. adding redundant information or retransmitting the lost packets). Both delay and bandwidth management consider avoiding packet loss and/or try to mask it in case of congested network.

Jitter Deviation in expected packet arrival time means variability in packet delivery, called jitter. Jitter is used as a measure of the variability over time of the packet latency across a network. A network with constant latency has no variation (or jitter). Packet jitter is expressed as an average of the deviation from the network mean latency. Low jitter smoothes the traffic and improves the real-time interactivity and hence it is the most desirable in DIMEs.

Skew Another important QoS parameter in a multimedia environment is skew. It is defined as the difference between the delays suffered by the monomedia flows belonging to the same multimedia stream. For example, skew defines how well the synchronization is between audio and video flows. Lower skew means better synchronization. As most of the DIME applications have separate source of monomedia flows (e.g., microphone for audio and camera for video), maintaining synchronization over the internet is a big challenge.

Table 1 summarizes the resource taxonomy and the QoS-based evaluation metrics that we will use to compare the various resource management techniques, appropriate for DIME cyber-physical environments.

Table 1 Impact of system resources on QoS metrics

QoS Metrics →	Bandwidth		Delay			Packet Loss	Jitter	Skew
	CPU	Network	D_{sender}	D_{net}	$D_{receiver}$			
CPU resource	√	X	√	X	√	X	√	√
Network resource	X	√	√	√	√	√	√	√

3 DIME resource management

We divide this section into two subsections: **CPU Resource Management** and **Network Resource Management**. Bandwidth and delay management techniques are discussed in network resource management subsection.

3.1 CPU resource management

Managing CPU resource across multiple tasks is a scheduling problem. Based on the strictness in the QoS guarantees they provide, we can classify CPU management algorithms into two broad categories: **best-effort** multimedia schedulers and **real-time** multimedia schedulers. The rest of this section surveys in detail both of these categories and shows how they impact the QoS metrics mentioned in Table 1.

3.1.1 Best effort schedulers

Schedulers in this category follow application models similar to those of time-share general purpose systems. A system can concurrently run multiple tasks. Each of these tasks is assigned a time slice of the CPU based on a predefined policy. At the end of its time slice the scheduler would make another scheduling decision. We further divide this into two classes: **priority-based scheduler** and **proportional-share scheduler**.

A. Priority-based schedulers

General purpose Operating Systems extend their time-sharing schedulers to allow priorities to support multimedia applications resulting in a round-robin with multilevel feedback policy. Under this policy the scheduler always selects the process with the highest priority. An example of this is the Portable Operating System Interface (POSIX) [55].Under heavy system loads, processes with low priorities scheduled by these schemes suffer from starvation and unfairness, which causes increased delay. Developers using this scheduler resort to use over-provisioning to minimize unfairness and starvation at the expense of low system utilization.

In order to reduce these problems some systems incorporate a decay-usage policy [20], as part of their schemes. Under this policy the scheduler will temporarily decrease the priority of CPU intensive applications and will favor tasks with lower CPU utilization. Linux and SVR4 Unix are examples of these systems. These schedulers are effective providing balanced CPU resource allocation across all the streams but in some cases will fail to provide the sustained throughput required by multimedia applications.

More recent approaches try to tackle this transient problem. For example, the *Multimedia Class Scheduler* implemented in Windows Vista defines various

multimedia classes. Each multimedia class specifies different QoS parameters and patterns. The scheduler then temporarily boosts the priority of all the threads that have indicated that are delivering a stream as part of the process running in the foreground [59].

Another important approach is the scheduler proposed by Banachowski et al. [5], which uses techniques from real-time systems to improve the responsiveness of time-share systems. In his approach, multimedia processes preserve the time-share system model from an application point of view, but internally the scheduler will assign them a deadline in a similar fashion as it occurs with real-time processes. Since the application model is unchanged, a heuristic is used to estimate the task deadline. This scheme has the potential of providing the same QoS guarantees in terms of bandwidth and delay as those of real-time schedulers discussed in Section 3.1.2. However, in practice the results are heavily dependent on the quality of the deadline values determined by the heuristic for the particular application.

B. Proportional-share schedulers

This type of scheduler is slice-based with a weight-based round robin policy. These schedulers guarantee that if there are Mcredits (shares/tickets) in the system, then application holding $N<M$ credits will get at least N/M of the CPU time in average. In some sense, each process appears to run in its own processor with N/M of the capacity of the real hardware. A very important advantage of this scheduler is that it uses the same application model of time-share general purpose systems, allowing existing applications to obtain statistical QoS guarantees without modification.

In many proportional-shares scheduling algorithms the throughput of a task is directly proportional to the number of tickets (credits/shares) it holds and the response time is inversely proportional to it. However, in multimedia systems, applications do not always follow this property. One important algorithm that decouples the number of shares from the average latency is *Earliest Eligible Virtual Deadline First (EEVDF)* from Stoica et al. [67].

However, EEVDF does not solve the latency problem completely, because applications cannot specify precise timing requirements, just proportions. To solve that, Kim et al. [37] proposed a proportional-share scheduler for multimedia applications based on the *Stride Scheduling* [73]. While these approaches solve the problem of scheduling user level tasks to guarantee statistical QoS, multimedia systems are still subject to *livelock* problems introduced by the operating system.[3] Jeffay et al. [31], propose to assign weights to kernel threads and activities to solve this livelock problem.

One of the main problems of proportional share schedulers is that they only provide statistical guarantees and they do not guarantee that the actual deadlines are met. For that reason, applications that require more stringent QoS requirements usually employ real-time schedulers along with some reservation mechanism. Next section will describe in detail of these schedulers along with their advantages and their shortcoming.

3.1.2 Real-time schedulers

Schedulers in this category usually follow a periodic model, in which a task t defines a certain computation time C_t that must be completed before a deadline D_t. The arrival

[3] OS that is I/O intensive and spends most of its time processing I/O in the kernel, leaves very little time to applications that are scheduled under a Proportional-Share policy.

intervals of this computation task requests arrives at a constant rate $1/P_t$. If the system is constrained in such way that for every task t in the system the period is equal to the deadline that is $P_t=D_t$, then the system follows the Liu-Layland model [43]. This model is the most commonly used, due to its simplicity and the fact that it can still efficiently model a great number of multimedia systems. Moreover, the delay of any task that meets its deadline is bound to the period P_t specified by the application.

Optimal schedule solutions for this model along with exact and bounded schedulability tests based on the processor utilization are well studied. Two of the major algorithms to solve this model are the *Rate Monotonic (RM)* schedule using static priorities and the *Earliest Deadline First (EDF)* schedule using dynamic priorities, both introduced by Liu and Layland as part of their model. Formally, for the Rate Monotonic schedule the bounded schedulability test is expressed by the following condition:

$$\sum_{i=n}^{n} \frac{C_i}{P_i} \leq n(2^{1/n} - 1)$$

Here n is the number of tasks in the system, C_i is defined as the processing requirements in units of time required by the i-th task in the system and P_i is the period of the i-th task in the system. For the case of the Earliest Deadline First schedule, the bounded schedulability test is the following:

$$\sum_{i=n}^{n} \frac{C_i}{P_i} \leq 1$$

where n is the number of tasks in the system, C_i is defined as the processing requirements in units of time required by the i-th task in the system, and P_i is the period of the i-th task in the system.

One of the many systems in real-time schedulers is the *Resource Kernel (RK)* proposed by Rajkumar et al. [56]. RK is a multi-resource kernel scheduler for real-time applications. RK allows applications to perform reservations of multiple resources. RK uses a priority inheritance algorithm with a bounded waiting time and a static priority algorithm (e.g. Rate Monotonic) to schedule CPU resources.

The problem with static priority algorithms is that in some cases the maximum achievable system throughput or utilization is under 100%. To solve this problem the Rialto system [33] uses a dynamic priority algorithm, the *Least Laxity First (LLF)* algorithm. This algorithm is similar to Earliest Deadline First and it also allows for full resource utilization. In addition, the Rialto architecture also considers the scheduling of Best-Effort applications that do not require QoS guarantees (e.g., a word-processor).

While the approach mentioned above allows seamlessly integration of best-effort and real-time applications. Best-effort applications tend to suffer from starvation due to the fact that they do not specify any reservation in the system. To solve this problem the SMART system [54] uses a best-effort real-time approach based on Proportional Share scheduling for overall resource allocation and the Earliest Deadline First scheduling to ensure that real-time applications meet their deadlines. This allows users to prioritize best-effort applications over real-time applications.

Another approach to solve this problem is to use hierarchical scheduling. In hierarchical scheduling a parent scheduler assigns time slots to various child schedulers. The challenge with these schedulers is to find a compositional real-time guarantee. Shin et al. [64] proposes a model to find such guarantee when both schedulers are periodic.

Usually systems based on compositional guarantees have complicated delay management and might reduce the ability of the scheduler to take fully aware decisions. This is due to the partition of resources across different child schedulers, resulting in reduced throughput. To address this issue, Brandt et al. present a scheduling model called *Resource Allocation/Dispatching (RAD)* [8] that decouples the resource allocation with the delivery time of these resources. They use a modified EDF scheduler with dynamic period adjustment and dynamic process rate adjustment.

A more recent approach that allows the seamless integration of best-effort and real-time applications is *Dynamic Soft-Real-Time (DSRT)* [12], proposed by Chu et al. In DSRT, he introduces the concept of CPU service classes. These service classes partition the set of tasks in the system according to its QoS requirements. Each service class is handled by its own scheduler. DSRT also uses reservation of resources, and adaptation to cope with task resource demand variations along with cooperation across best-effort and EDF real-time schedulers. Figure 4 shows the basic architecture of DSRT and its main components: the Overrun Protection Timer, the Soft Real-Time Scheduler, the Admission Control and the Cycle Demand Adaptor.

The problem with DSRT real-time schedulers is that usually the integration of network constraints with CPU constraints. iDSRT [52] represents an integrated architecture using 3 main components: a CPU scheduler similar to DSRT, a network packet scheduler using implicit EDF scheduling, and a Coordinator entity in the middleware using *Time Division Multiplexing Access (TDMA)* to coordinate the access to the shared medium of the Wireless Local Area Network. Figure 5 shows the basic architecture of the iDSRT framework along with its main components previously discussed.

Finally, recent emerging virtualization technologies have brought back the problem of hierarchical schedulers and compositional guarantees. Rivas et al. [58] proposes a much simpler approach in which he uses a *Dynamic Mapping (DM)* of real-time tasks into virtual CPU resources and then into physical CPU resources. It consists of three main entities: a Distributed **Kernel-level Coordinator**, a **Real-Time Scheduler**, and a collection of services and interfaces referred as the **Janus Executive**. Figure 6 shows the basic architecture of Janus with its main entities. Janus dynamically maps real-time tasks into Virtual CPU's at the Virtual Machine Level and Virtual CPU's with physical CPU's at the Virtual Machine Monitor level. These mappings result in a flat allocation of resources that can be scheduled by the real-time scheduler. The real-time scheduler uses the partitioned

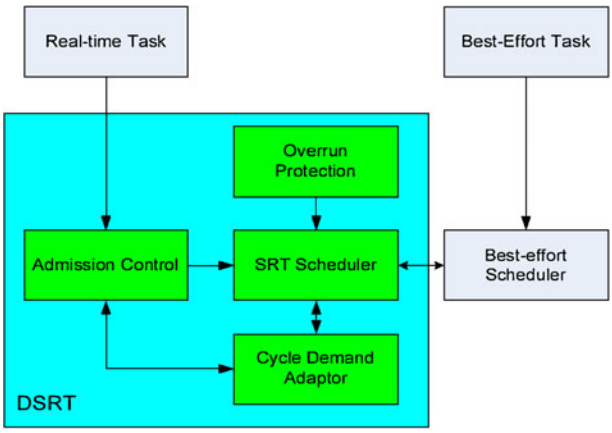

Fig. 4 Basic architecture of DSRT

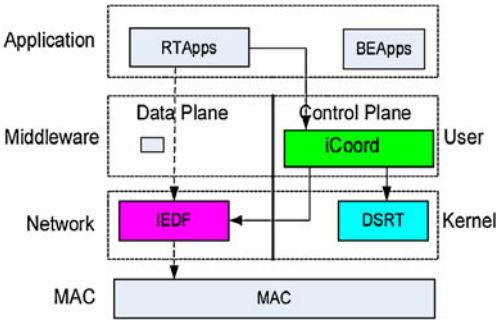

Fig. 5 Basic architecture of iDSRT

Earliest Deadline First algorithm. Under this approach the flattened set of tasks are partitioned across the physical CPU's available in the machine. Then the EDF policy is applied to each partition independently.

A summarized view of the scheduling systems and algorithms discussed above is shown in Table 2.

3.2 Network resource management

Network management can be further classified into **bandwidth management** and **delay management**. Bandwidth management uses network bandwidth effectively to reduce packet loss by assigning sufficient bandwidth to the participants. Delay management reduces jitter, skew for synchronization and most importantly network delays.

3.2.1 Bandwidth management

The bandwidth management techniques can be classified broadly into two categories: 1) Reservation-based bandwidth management, and 2) Adaptive bandwidth management.

Reservation-based bandwidth management The bandwidth management in this category is based on reserving bandwidth for an allocated request in advance during the session initiation and this reservation is maintained till the session ends. *Reservations* are triggered by one of the end host and the reservation requests are forwarded to all routers along the

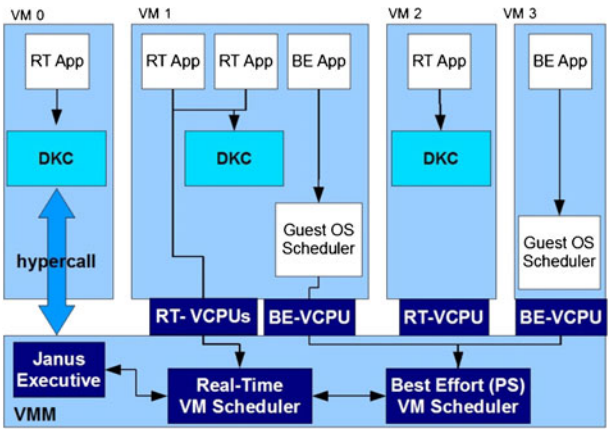

Fig. 6 Basic architecture of Janus

Table 2 Comparison of CPU scheduling algorithm suitable for DIME systems

Algorithm	Scheduling Priority	Best Effort Support	Algorithms	Adaptive
RK	Static	No	RM, DM	No
Rialto	Dynamic	Yes	LLF	No
SMART	Dynamic	Yes with QoS	PS+EDF	Yes
Hierarchical Scheduling	Both	Yes	Any but requires to meet constraints	No
RAD	Dynamic	Yes	EDF+Priority and Period Adjustment	Yes
DSRT	Dynamic	Yes	EDF	Yes
Janus	Dynamic	Yes	EDF+Dynamic Mappings	No
iDSRT	Dynamic	Yes	EDF+TDMA	Yes

end-to-end path for admission control and finally QoS enforcement. *Admission control* is a mechanism used to accept, modify or reject new connections. Admission control is based on specific negotiation protocols in use. If $QoS_{\min}$ is the minimum QoS requirement for a service, QoS_{req} is the requested QoS and QoS_{bound} is the maximum supported QoS for a service, then we can define admission control by the following equation:

$$QoS_{\min} \leq QoS_{req} \leq QoS_{bound}$$

The bandwidth admission control checks the availability of bandwidth over shared network resources using different types of admission tests. An example of admission test for ensuring proper bandwidth allocation and throughput guarantees is given as under the following equation, where j is the number of channels with already allocated bandwidth:

$$\sum_{j} B_a^j + B_r \leq B_g$$

Here, B_a^j is the allocated bandwidth for channel j, B_r is the requested bandwidth for a new channel and B_g is the total bandwidth in the system. The basic structure of reservation-based model is shown in Fig. 7. Existing reservation protocols follow either *sender-driven* reservation or *receiver-driven* reservation. In sender-driven reservation the sender initiates the reservation mechanism while in receiver-driven reservation the receiver initiates the reservation mechanism. After the reservation is done, the data flows from the sender to the receiver. The reservation protocols also differ in how the monitoring states are maintained at the routers and hence can be classified as *hard-state* or *soft-state*. In hard-state reservation the states are maintained at the routers from session initiation until the session end, i.e. until the end hosts explicitly send a teardown message. In a soft-state reservation the states are maintained only for a given time period (e.g., 30 sec), and hence, the end hosts need to send refresh messages to keep the reservation states at the routers alive.

There are two well-known Internet architectures supporting *advanced reservation*: 1) **Integrated Services** (Int-Serv), and 2) **Differentiated Services** (Diff-Serv). Int-Serv allows

Fig. 7 Request driven bandwidth reservation model

for fine grained QoS specification while Diff-Serv provides coarse grained QoS specification. The details of both the architectures are explained further in the following sections.

A. Integrated services (Int-serv)

Int-Serv allows for individual resource allocation and reservation for each request originating at end hosts, and hence, provides finer guarantees of QoS requirements. To implement Int-Serv, *Resource Reservation Protocol (RSVP)* was proposed by IETF [7]. RSVP is a network layer protocol providing control signals and commands to disseminate reservation requests/responses along the end-to-end path between end hosts. RSVP enables resource allocation and reservation for applications having different QoS requirements for their data flows. RSVP is a control protocol and depends on underlying routing protocols to route the data flows. The sessions are identified by combination of destination IP address, protocol ID, and destination port. It is a receiver-driven, uni-directional, soft state reservation protocol and supports both unicast and multicast sessions. RSVP supports four major functionalities (1) admission and policy control, (2) packet classifier, (3) packet scheduler, and (4) resource reservation.

Each router in the end-to-end path runs a RSVP daemon, which upon receipt of a path request message, determines if the router has sufficient available resources (admission control) and determines if the user has administrative permission to make the reservation (policy control). An error notification is sent to the requester if either check fails. If both checks succeed, the RSVP daemon sets parameters in a packet classifier and packet scheduler.

The traffic specification classifies the packets into three classes of services:

- **Best-effort**: traditional IP traffic with reliable delivery of data.
- **Rate-sensitive**: guaranteed transmission rate between source and destination, helpful for *Constant Bit Rate (CBR)* traffic.
- **Delay-sensitive:** guaranteed timeliness of delivery with/without reliable delivery, helpful for *Variable Bit Rate (VBR)* traffic.

The flow specification is used to communicate the required QoS requirements in terms of level of service required for a data flow. The packet scheduler uses the flow specification to schedule a packet at a node to achieve the promised QoS for each stream.

RSVP supports two types of reservations: *distinct reservations* and *shared reservations*. In distinct reservations an individual flow is installed for each requesting node in a given session while in a shared reservation, the resources are shared by non-interfering set of requesting nodes. RSVP supports several filters which define various reservation styles like:

- **Wildcard-Filter Style**: It specifies a shared reservation in which flows from all upstream senders are mixed based on the largest resource requests on a shared link between receivers and different senders.
- **Fixed-Filter Style**: It specifies distinct reservation with explicit scope on a link based on the total reservation needed for all the requested senders in a session.
- **Shared-Explicit Style**: It specifies shared reservation with explicit scope in which flows from all upstream senders, explicitly specified by the receiver, are mixed together.

A major drawback of Int-Serv based protocol like RSVP is that it requires universal deployment over all the nodes on the end-to-end path which requires changes in all the

intermediate nodes. The amount of states needed to be maintained at each router hop becomes very large in terms of memory and complexity with increase in number of requests. The soft state maintenance requires periodic refresh messages which add to the overall network traffic. In [74] and [69], authors have discussed remedy measures for increasing scalability and deployability, however even with enhanced scalability of Int-Serv, the slow upgrade of current networks towards deploying it makes it unlikely that we will see major deployment of RSVP in near future, hence usage by DIMEs.

B. Differentiated services

To mitigate the scalability issues of Int-Serv model, IETF proposed Differentiated Services (Diff-Serv) [53]. IntServ follows the flow-based QoS specification model, where the end-hosts define QoS requirements for each flow, while DiffServ works on class-based QoS model. The traffic in Diff-Serv is categorized into different classes, known as *Class of Service (CoS)*, and the network elements reserve resources based on these classes. The routers assign higher priority to higher priority classes. The sender uses *Type of Service (ToS)* bytes (redefined as DSCP field) in IP header, to specify the class for a packet.

A Diff-Serv domain is divided into two types of nodes: *boundary nodes* and *interior nodes*. The boundary nodes, which are essentially the end hosts, perform packet classification, traffic shaping, and congestion control. The interior nodes perform monitoring, traffic shaping, and re-marking packets based on the aggregate class specified.

Per-Hop Behaviors (PHBs) are used by Diff-Serv enabled routers to define how a packet is forwarded in the router. Diff-Serv provides some standardized set of PHBs, however; users can define their own PHBs to specify what types of traffic are given higher priority. A PHB provides queuing, packet scheduling, policing, traffic shaping based on the service class and the *Service Level Agreement (SLA)* policy. Multiple flows can belong to the same PHB and hence, are aggregated to form a *Behavior Aggregate (BA)* at a given router. Some of the PHBs currently being supported include:

- **Default PHB**: This class is used for traditional best effort traffic.
- **Class-Selector PHB**: This class has same forwarding behavior as IP-precedence based classification.
- **Expedited Forwarding PHB**: This class provides low-loss, low-latency, low-jitter, and assured bandwidth service.
- **Assured Forwarding PHB**: In this class different flows in a BA can be provided different forwarding assurances (like buffer space, bandwidth, loss-rate, etc.).

Though Diff-Serv provides better scalability, it suffers from several issues. Provisioning in Diff-Serv requires knowledge of the traffic statistics to setup various classes throughout the network. Also, the heterogeneity of the definition of classes existing between different autonomous servers (one AS's gold class might be another AS's bronze class) can lead to unknown behavior of the traffic. In Diff-Serv, we lose the ability of fine-grain control over the bandwidth allocated to flows and hence, for variable bit rate applications, this can potentially lead to over/under utilization of the network. The biggest drawbacks of both RSVP and Diff-Serv models is the fact that provisioning in independent of the routing process. Thus, there may exist a path in the network that has the required resources, even when RSVP/DiffServ fails to find the resources. To mitigate this effect, *Traffic Engineering (TE)* and *Multiprotocol Label Switching (MPLS)* are used.

Table 3 Properties of reservation-based resource management approaches

Algorithm	QoS Specification Model	Reservation Initiation	State Reservation Strategy	Amount of State Maintenance per hop	Scalability
RSVP [7]	Flow based (fine grained)	Receiver driven	Soft state reservation	High	Low
Diff-Serv [53]	Class based (coarse grained)	Receiver driven	Soft state reservation	Low	High

In Diff-Serv framework since each domain might be incompatibly configured, it does not solve the problem of providing end-to-end QoS. One entity which overcomes this problem is the Bandwidth Broker.

C. Bandwidth broker

A Bandwidth Broker [50] is an entity responsible for providing QoS across different network domains. A Bandwidth Broker manages the resources within the specific domain by controlling request admission and allocation of resources based on current network load and on SLA. To setup end-to-end QoS capability between two nodes, Bandwidth Broker contacts its peer Bandwidth Brokers in the other domains to negotiate and establish bilateral agreements and thus provides a purely administrative way to setup QoS guarantees across different domains.

Table 3 shows the properties to RSVP and Diff-Serv protocols for bandwidth management. The major drawback of reservation oriented bandwidth management is the fixed resource allocation which might not work for applications which show high variability in bandwidth needs. Thus, researchers propose to use adaptive bandwidth management techniques which perform bandwidth monitoring and based on the probing results assign and adapt to the changing requirements and availability of resources.

Adaptation-based bandwidth management The bandwidth management techniques in this category continuously monitor network statistics like RTT, loss rate, packet size, etc., to estimate current total available bandwidth between two nodes. Depending on the monitoring output, it adapts the QoS request parameters. Admission control is done after adaption and finally the admitted QoS parameters are enforced. The model of adaptation driven bandwidth management is given in Fig. 8.

Bandwidth estimation is an important part of any adaptive bandwidth management techniques and hence, many techniques either develop their own bandwidth estimation scheme or they make use of several bandwidth estimation schemes proposed in literature like Pathload [29, 30], Abget [2], Sting [62], and SProbe [60]. Pathload and Abget are similar to each other and perform end-to-end bandwidth estimation of a TCP flow. Abget, Sting, and SProbe

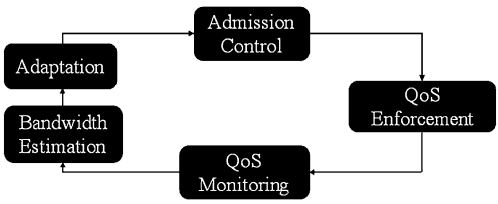

Fig. 8 Adaptation-based bandwidth management

use only single-end probing to estimate bandwidth on TCP flows. These techniques provide the estimated total bandwidth to the bandwidth management schemes which adaptively configure the bandwidth for the flows. This allows for dynamic bandwidth adaptation to the changes occurring in network conditions. The adaptive bandwidth schemes can be categorized based on if the bandwidth management is performed per flow or if it is performed on aggregated correlated flows between a single source and common destination.

A. Single flow adaptive bandwidth management

Many of the transport layer protocols employing congestion control mechanisms like TCP, DCCP perform simple adaptive bandwidth management on each flow based on the congestion experienced in the network by reducing the sending rate. Apart from transport layer protocols, many schemes perform bandwidth management at the application layer as it allows for adaptation based on the specific needs and demands of different applications. [1, 11, 41, 44, 71], and [81] are some of the examples of application layer, single flow, adaptive bandwidth management schemes which employ techniques like simple frame skipping, content based frame skipping using event/motion detection and compression techniques to compress data in the frames.

One of the major drawbacks of single flow adaptive bandwidth management schemes is that they do not allow aggregated congestion control on multiple flows. Hence, single flow adaptation schemes are well suited for applications like video streaming requiring one or more than one unrelated flows.

B. Multiple flows adaptive bandwidth management

In the face of emerging DIME-based multimedia applications like 3D tele-immersion dealing with sending multiple correlated flows produced by multiple cameras, audio devices, and different types of body sensors between a source and a destination node, it becomes more advantageous to perform aggregated bandwidth management on the correlated flows. We discuss two multi-flow adaptive bandwidth management schemes.

Ott and Patel in [Ott07] present *Coordination Protocol (CP)* which performs coordinated bandwidth distribution across flows and allows for application based context-specific coordination. CP is a transport layer protocol and uses a cluster-to-cluster (C-to-C) model which provides congestion control on the aggregated flows originating at a cluster and consumed at the other cluster. All the flows share a common intermediary path between the clusters. An *Aggregation Point (AP),* usually the first hop router away from the clusters is used to converge and **diverge** the flows at the sender cluster and at the receiver cluster, respectively. CP first probes the network state to find out the RTT, loss rate, and available bandwidth for each flow and uses the state tables to store this information. After this, CP uses *Bandwidth-Filtered Loss Detection (BFLD)* to estimate the correct amount of total bandwidth available for all the flows using the probe results and hence, allows for distributing appropriate amount to each flow based on the context-specific coordination. Based on CP, authors develop CP-RUDP for performing congestion control on multiple flows produced in 3D tele-immersive systems. The authors cover a case study of 3DTI systems, a specific type of DIME system, wherein the bandwidth allocation across multiple flows is defined as shown below:

$$B_{flow_i} = \frac{FSize(flow_i)}{\sum_i^{numflows} FSize(flow_i)} \times (B_{net} \times numflows)$$

where B_{flow_i} is the bandwidth allocation for $flow_i$, $FSize(flow_i)$ is the typical frame size in $flow_i$, B_{net} is the available bandwidth on a per flow basis, and $numflows$ is the total

number of flows at the application layer which need to be considered together. Thus, CP-RUDP protocol is an advanced protocol which achieves congestion control between different flows at the transport layer level.

Another scheme by Yang et al. [80] proposes a multi-stream adaptation framework for **view-based bandwidth management** in 3D tele-immersive environments. This scheme makes use of the knowledge of application-dependent cues like user view, correlation among multi-stream flows, and content. The scheme works with a ***bundle*** of correlated streams generated at one source, and prioritizes some flows over other flows, depending on the flow information contribution to the user's view. We define the *camera* view (camera orientation) $\overrightarrow{O_{vs_i}}$ of a stream vs_i to be the normal vector of the imaging plane of the camera that produces the stream vs_i, which can be obtained by the calibration parameters acquired via the initialization phase. We define the user's view as the desired direction $\overrightarrow{O_u}$ of the view, user u would like to see. The flow information contribution to the user's view is determined by the contribution factor CF_i of the stream vs_i. The contribution factor CF_i of a stream vs_i with respect to a given user rendering view $\overrightarrow{O_u}$ is thereby defined by the scalar product of the two vectors: $CF_i = \overrightarrow{O_{vs_i}} \cdot \overrightarrow{O_u}$. The number of streams to select from each site is a tunable parameter. One can either select the top k number of streams for a given view, or a dynamic number of streams with contribution factor larger than a threshold TH_{imp}, depending on the network bandwidth availability. This leads to a priority-based bandwidth allocation.

The priority-based bandwidth allocation scheme is based on the following principles; (1) flows with bigger *CF* value have higher priority to be sent, and (2) after sending most important flows with highest *CF*, if bandwidth is still available, a minimum frame sizes for other flows are granted to widen the *Field of View (FOV)*. Based on these principles, the internal **frame size allocation scheme** for bandwidth management estimates an ideal size of a frame belonging to a flow of a bundle. The algorithm is as follows. (1) flows of a bundle are sorted in descending order of *CF*, and a target macro-frame[4] size (*TFS*) is determined, i.e., the underlying network estimates number of bits (*TFS*) it can transmit at that time (*TFS* represents the available network capacity for a bundle of flows/streams). So the admission control will compare available network capacity *TFS* with the macro-frame demand that application desires to be transmitted, i.e., the admission test is $\left(TFS < fs \times \sum_{i \in SI} CF_i\right)$, where *fs* is the size of the frame in a macro-frame,[5] and *SI* is the set of flows in the bundle, and $|SI|$ is the number of flows in the bundle. We have two cases with respect to the admission control:

(1) If$(TFS \geq fs \times \sum_{i \in SI} CF_i)$, i.e., there is more network bandwidth available than the desired application macro-frame size, then the allocated frame size A_i to be transmitted is:

$$A_i = \min\left(fs, fs \times CF_i + \frac{(TFS - \sum_{j=1}^{i-1} A_j) \times CF_i}{\sum_{j=1}^{|SI|} CF_j}\right)$$

[4] Marco-frame is a group of correlated 3D frames captured at the same time t and at the same site.

[5] We assume that each frame in the macro-frame has the same size since the cameras producing the frames are the same, however, each frame might have a different contribution to the user's view. If the frame contributes fully to the view (i.e., camera is placed in the front of the user), then CF=1, and the desired frame size to be transmitted is $fs \times CF = fs$. If the frame contributes to the view only half-way (e.g., camera is placed on the side of the user), then CF=0.5 and the frame size to be transmitted will be $fs \times 0.5$.

(2) If $(TFS < fs \times \sum_{i \in SI} CF_i)$, i.e., the network bandwidth availability is lower than the desired amount of bits for the application macro-frame, then the minimum frame size A_i is allocated in order of priority as:

$$A_i = \min(fs \times CF_i, TFS - \sum_{j=1}^{i-1} A_j)$$

Thus, it is possible that some of the selected streams may not get the quota of transmission. The adaptation is done at the application layer and hence, the metrics used in the bandwidth management can be modified based on the requirements of the application. Table 4 shows a summarized view of different kinds of adaptive bandwidth management algorithms.

3.2.2 Delay management

Both delays and the variations of delays, i.e., "jitters", are critical QoS metrics in DIMEs. ITU G.114 [28] has prescribed that a one-way latency exceeding $400ms$ is unacceptable in the two-party conference. A large body of work has been done in order to minimize the delay components. On the other hand, managing jitters is mainly performed at the receiver side. We will first review the schemes designed to reduce sender-side delays (D_{sender}) and network delays (D_{net}) in DIMEs, and then discuss the jitter management in the part of receiver-side delays ($D_{receiver}$).

Finally, we discuss how delay management impacts the skew in the correlated streams. Since DIME can include multiple sensory streams (video, audio and etc.) that may be mutually correlated [25], a good adaptation scheme should be developed to allow the *real-time synchronization* among those streams. But it requires the coordination of all three locations (sender-side, network, receiver-side locations), and it is sometimes very difficult to achieve this without introducing additional latency overhead. We present the related

Table 4 Adaptation based bandwidth management algorithms

Algorithm	Bandwidth Management Model	Reservation Initiation	Bandwidth Management Strategy
Agarwal-Rejaie [1]	Single Flow	Receiver driven	Application/Layer and frame skipping based.
Vickers-Albuquerque [71]	Single Flow	Sender driven	Network/Layer and bit rate based.
Chang-Zhong [11]	Single Flow	Sender driven	Application/Content-based
Liu-Choudary [45]	Single Flow	Sender driven	Application/Content, compression based
Liu-Nelakuditi [LiuOct04]	Single Flow	Sender driven	Application/Content-based
Yeung-Liew [81]	Single Flow	Sender driven	Application/Content-based frame skipping
Coordinated Protocol [Ott07]	Multiple Flow	Sender driven	Network-based
Yang-Nahrstedt [80]	Multiple Flow	Sender driven	Application/view-based

studies of the delay management for multimedia synchronization, and consider delay issues for multi-stream (bundle) cases.

In "Sender-side delay" to "Receiver-side delay" we discuss sender-side, network, and receiver-side delay management for single streams. In "Multi-stream synchronization delay management", we discuss multi-stream synchronization delay management that utilizes the individual stream delay control at all three locations.

Sender-side delay Sender-side delay is composed of computing delays at the input LNs and local RP, as well as communication delays from the LNs to RP. Computing delays are mainly due to stream acquisition and encoding/packetization tasks whereas communication delays are composed of the sender buffering delay and transmission delay over the LAN.

Computing delay management can be categorized into two categories: software-based and hardware-based. Software-based approaches use algorithmic improvement to reduce 2D video encoding and 3D reconstruction delays. The optimization of 2D codecs includes speeding up the motion search [38, 42, 68] and the selection of macroblock coding-mode in the video [17, 78]. The improvement of 3D reconstruction has also been achieved in [70]. The latency incurred on 2D audio encoding or 3D audio reconstruction is usually small. Hardware-based approaches take advantage of parallelization on multi-core and GPU architectures to accelerate capturing and processing of media data. This can be achieved by using task multithreading on a single computer [23, 24], regardless of whether the computer is a multi-core machine with [24] or without [23] GPU support.

The sender buffering delay depends on the packet scheduling and adaptation algorithm at both application and transport layers. The sender buffer at the application layer can delay the packets by scheduling with time spacing [25] to avoid congestion at the sender site. The packets can also be delayed due to the interleaving scheme for loss concealment [72]. The buffering delay at the transport layer on the other hand, is caused by the transport protocol and its ability of adaptation in response to Internet dynamics. Both TCP's additive increase multiplicative decrease (AIMD) approach and DCCP (CID-2)'s TCP-like congestion control scheme can increase the sender buffering abruptly under a sudden congestion. DCCP (CID-3) implements a TCP-friendly rate control (TFRC) scheme which can allow comparatively smooth throughput and the transport-layer buffering will change less abruptly.

Internet network delay The network delay among DIME sites can be first categorized based on whether a leased, specialized network is used or the commodity Internet is used. Cisco's tele-presence solution deploys a converged IP next-generation network (IP NGN) with both inter-company and intra-company models [14]. Using such dedicated networks has the advantage of avoiding resource competition with unpredictable public traffic and providing abundant bandwidth as needed to minimize queuing and re-transmission delays caused by congestions. However, the disadvantage is the cost and inflexibility to extend to more sites.

Many more DIMEs (e.g., Skype, TEEVE) use the open and free alternative—Internet—as the communication medium. The end-to-end Internet delay management can be further categorized into two groups: 1) point-to-point and 2) multi-point management. Point-to-point delay management refers to the schemes that focus between two end points (i.e., RPs). It is tightly related with bandwidth management, e.g., congestion control, as discussed in Section 3.2.1, which can reduce the delays incurred by queuing and re-transmissions in times of congestions.

When there are multiple sites collaborating in DIMEs, multi-point (topological) delay management becomes important. Since the delay incurred on the network highly depends

on the geographical distance between the sender and receiver, maintaining an efficient multi-point *Application Level Multicast (ALM)* topology is thus critical for reducing transmission delays in DIMEs.

There are mainly two ways of disseminating streams among multiple DIME sites: 1) IP Multicast and 2) Application-Level Multicast. IP multicast [16] adds multicast support to routers, and therefore achieves highly efficient bandwidth usage by avoiding duplicate packets on the physical routing paths. However, its adoption is rather limited because of the low scalability (routers have to maintain per-group states) and difficulty to support higher-level functionalities due to semantic gap. To mitigate these issues, ALMs are proposed as an alternative where an application-level overlay is used to disseminate streams among multiple DIME sites. There has been a large body of work on constructing delay-efficient multicast topologies, which can be roughly classified into two groups: 1) Tree-based, and 2) Mesh-based.

Tree-based approaches construct a tree for each stream. In cases of multi-stream/multi-site DIMEs, multiple trees are used. Tree-based approaches construct delay-optimized/constrained trees depending on whether the goal is to minimize or bound the end-to-end delays. The problem is often a case of the Steiner Tree problem, which is known to be NP-complete [26]. Apart from the main goal of minimizing the total cost, DIMEs often imposes additional constrains such as bandwidth, delay, jitter, or a combination thereof. Hence the problem becomes a *Constrained Steiner Tree (CST)* problem. Researchers have proposed many heuristic algorithms to achieve sub-optimal results. [32, 34, 82] are some examples for the delay constrained CST problem.

Zhu's algorithm [82] achieves $O(KN^3 \log N)$ complexity, where N is the number of nodes, and K is the number of paths in the initial minimal cost tree. Kompella's algorithm requires $O(N^3)$ complexity to construct a delay-bounded routing tree, while Jia's distributed algorithm takes $O(2M)$ messaging overhead to construct a tree for M destinations.

Tree-based algorithms have the advantages of achieving efficient bandwidth and delay optimization in stable networks. However, the single tree approaches [13] can be vulnerable to churns. If an intermediate node leaves or fails, its descendant nodes will be affected. *Repair delays* are hence incurred by relocation, rearrangement of the topology. Mesh-based approaches have been proposed as a promising alternative to address the issue. SplitStream [10], for example, splits the stream into k stripes, and builds a separate tree for each stripe over Scribe [9]. The key is to have each node acts as an interior node in at most one tree so that the overlay becomes robust to node leaves and failures with proper content encodings. Bullet [35], as another example, uses meshes to distribute disjoint data sets to various points, and designs a decentralized algorithm for nodes to locate and retrieve missing data.

Receiver-side delay The delay incurred at the receiver side ($D_{receiver}$) usually includes three parts: the loss concealment overhead, the receiver buffering time and the time incurred on audio/video 2D decoding and 3D rendering. The 2D decoding and 3D rendering can usually be achieved in real time, and their delays are very small [75].

Different loss concealment schemes have different delay overhead. For instance, when we conceal audio packet losses using piggybacking approach [61], the maximum delay depends on the piggybacking degree which is the number of previous audio frames piggybacked in the current frame. But when we use packet interleaving to conceal losses, the delay is decided based upon the number of interleaving sub-frame partitions for each frame [72]. Video frames are usually concealed by simply replicating previous frames whose latency is negligible, or by motion vector estimation which requires huge computation resources [3].

The playout scheduling with jitter control decides the end-to-end delay D_{e2e} by adjusting the receiver buffering time $D_{receiver}$. Most of the previous studies focused on the audio buffering, and the video buffer were usually adapted to synchronize to the audio signals (see "Multi-stream synchronization delay management"). The audio scheduling algorithms can be broadly divided into three categories.

The non-adaptive playout scheduling scheme [57] estimates the average one-way network delay D_{net} and its variation v, and chooses the playout buffer with the estimated one-way end-to-end delay D_{e2e} using following equation:

$$D_{e2e} = D_{net} + 4v.$$

Both D_{net} and v can be obtained by sending probing packets from each other clients in the conference. This scheme is not able to adapt to Internet dynamics because of the fixed buffer size determined at the start of the conference.

On the other hand, the adaptive playout scheduling scheme [6, 45, 51] is able to dynamically adapt the buffer size during the DIME session, based on the feedback and gained statistics of the delay, jitter and losses from other clients. A simple implementation is to collect statistics at each client with regard to packets sent from each of the speakers [61]. Let F be the CDF of the network delay D_{net} between a speaker-listener pair in the past time window (e.g., 10 sec), D_{e2e} can be computed from the history window:

$$D_{e2e} = F(D_{net}) + \Delta T$$

Here ΔT is the additional allowance for the network jitter.

Although different adaptive playout scheduling schemes have all been proved to be successful under Internet dynamics, their implementations are very difficult. In addition, that the jitter buffer size cannot be changed within a *talkspurt* degrades the overall effectiveness.

Liang et al. [39] proposes the *Time Modification Scheme (TSM)* scheme, of which the main purpose is to further increase or decrease $D_{receiver}$ without changing D_{e2e}. TSM stretches or compresses speech frames at the very beginning and the end of a talkspurt to reduce listeners' perception annoyance. Therefore, it has small effects on the jitter buffer size.

Multi-stream synchronization delay management The synchronization of correlated multi-streams (bundle of streams) requires ***collaborative delay management*** at all three locations (sender-side, network, and receiver-side). When the synchronization needs to be done on only one audio and one video stream, we can achieve lip synchronization by scheduling the video playback time according to the audio reference [40, 46] or the audio playout time (by warping the audio signals) according to the video reference [19]. In distributed multimedia environments, we can synchronize multi-streams at the sender side by multicasting a global timestamp or at the receiver side by applying *Precision Time Protocol (PTP)* [27]. When multiple correlated audio and video streams need to be synchronized in a DIME, synchronization becomes a challenging task. Due to different protocols and adaptation algorithms used for different streams, the individual streams of a bundle may arrive at a receiver at different time. The system may no longer be real-time if we try to synchronize all streams.

To solve this issue, Huang et al. [25] propose the TSync framework. The authors propose the concept of *dominant stream(s)* within the bundle,[6] which are the *reference stream(s)* used for real-time synchronization. The re-synchronization of a bundle relies on *timed synchronization points* throughout the transmission to avoid synchronization skew

[6] **Bundle** is defined as a group of correlated streams/flows originated at one site.

propagations and degradations. TSync also adapts audio playout buffer to synchronize to video frames based on the previous study in [48]. TSync provides the following features:

- As it is impossible for an audio stream to synchronize to multiple video streams, TSync uses the concept of a *dominant stream* which has the largest *contribution factor* (*CF*) to the users view and represents the reference stream for other streams in the bundle for overall synchronization. The CF_i of each *i*-th video stream (vs_i) can be computed as discussed in "Multiple Flows Adaptive Bandwidth Management".
- To allow synchronous capturing of multiple video frames (i.e., to generate a video macro-frame as we discussed in "Adaptation-based bandwidth management") at the capturing tier (sender-side), TSync uses a video trigger server. The video trigger server gets the feedback from cameras when cameras capture their current frame, and sends a hardware beat signal to all cameras at the sender-site when cameras become ready to capture the next frame. In order to facilitate audio-visual synchronization, a small soft packet piggybacks onto a global timestamp and it is sent to all cameras as well as the microphone component at the sender-side (capturing tier).
- To reduce the sync skew caused by inadequate Internet network bandwidth availability, TSync relies on a coordinated resource allocation scheme for the multiple audio and video streams under different network topology.
- To reduce the audio-visual sync skew at the rendering tier, TSync allows audio buffer adaptation based upon the average 3D video rendering time and an empirical 80th percentile *audio-visual arrival skew* (the arrival time skew between the last frame of a video macro-frame and the audio frame with the same timestamp) at the receiver. The audio buffer is adapted during silence periods and a minimal audio buffering time of 60 *ms* is guaranteed to smooth the Internet jitters.
- To correct sync skew, and prevent sync skew propagation and further degradation, TSync applies *Timed Synchronization Points (TSP)* to the video macro-frames at both the capturing and rendering tiers. The basic idea is that at each TSP, the system waits (waiting implemented as a barrier) to release a video macro-frame until it receives all frames from the macro-frame or until it receives some frames of the macro-frame within a given bounded time, whichever is earlier. If a frame out of its macro-frame is missing, i.e., macro-frame is incomplete at the release, TSync system replicates the previous correctly received frame belonging to the same video stream.
- When a user requests a *view change* at the rendering tier (receiver-side), the dominant stream can change to another video stream which may not be available. In TSync, this issue is addressed by delaying the response[7] until resynchronization of audio signal with the new dominant video stream happens.

4 Discussion

Different resource management algorithms impose different guarantees on QoS metrics. Hence careful selection and adaptation of these algorithms must be made depending on application requirements. For example, the multi-player immersive gaming is more

[7] By delaying the response, we mean that during the resynchronization phase when bringing in a new dominant stream, DIME and its TSync service play the old video streams with the new audio until new video streams corresponding to the new view arrive. This action will cause a delayed synchronized response to the user's view change.

sensitive to delays whereas synchronized immersive dancing [63] is more sensitive to synchronization or skew across multiple video streams. In this section, we briefly present a sample DIME architecture, called TEEVE (*T*ele-immersive *E*nvironment for *Ever*ybody) [79], to show how some of the QoS and resource management algorithms are applied depending on application requirements.

TEEVE is a distributed 3D tele-immersive (3DTI) system that has been under development jointly at University of California at Berkeley and University of Illinois at Urbana-Champaign since 2004. The system is designed with multiple cameras and displays at each site (see Fig. 1) to capture 3D full-body human motion in geographically distributed sender-sites, aggregate the correlated video data (bundle) at each sender-site, send the bundles across the Internet, and render the bundles into a joint cyber-space at each receiver-site. Through the shared visual context, the participants can interact or collaborate in real time (see Fig. 9). TEEVE is meant to support real-time collaborative physical activities among multiple geographically distributed sites.

The TEEVE system is built upon *Commercial Off-The-Shelf (COTS) computing and networking infrastructures*. There exist other tele-immersive environments with very special purpose real-time operating systems, leased networks, or other special purpose and expensive hardware [15, Otto04, 3]. But the cost is too high for the system to work for "everybody". TEEVE goal is to build the system with off-the-shelf infrastructures, so that it can be easily and widely deployed. The lower end COTS-based 2D video-mediated collaborative environments such as Skype, NetMeeting [47, 65] are mostly designed for desktop uses (e.g., conferencing), hence minimal number of cameras are employed only to capture the upper bodies or the faces. The higher end 2D video-mediated collaborative environments such as Cisco Telepresence, HP's Halo system [14, 22] are designed as conferencing rooms with multiple cameras, but are employed to capture the upper bodies or faces for efficient business meeting purposes. Our goal is to broaden the application of the tele-immersive technology by supporting full-body capturing of the participant. The complete 3D representations of human body allow for a wide range of physical activities such as dancing, sports, and medical rehabilitation [36, 63, 76].

In the next subsections 4.1, 4.2 and 4.3 we will discuss the QoS and CPU/network management for TEEVE. It is clear that since TEEVE resides on COTS components (general purpose OS and Internet protocols), the primary QoS and resource management will be **adaptive** since we cannot rely on any real time enforcements in a single OS and inside of the network via Int-Serv or Diff-Serv.

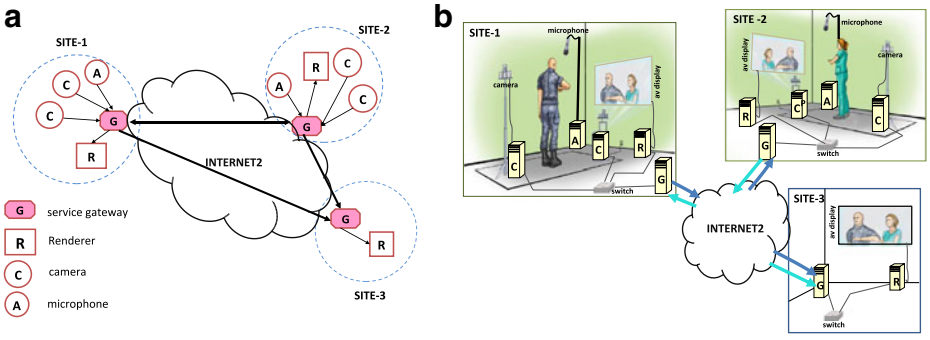

Fig. 9 **a** Logical architecture, and **b** physical setup of TEEVE with 3 sites

4.1 CPU management in TEEVE

The TEEVE system does not have any CPU management support at the LN and RP nodes for a single session. The TEEVE overlay system services utilize the best effort CPU scheduling of the Microsoft Windows and Linux OS systems. However, at the RP nodes, in order to support multiple simultaneous sessions (e.g. TI rooms), the TEEVE system uses the concept of real-time virtualization. Virtualization is a powerful technique that allows efficient multiplexing of resources to run multiple virtual machines in a single physical machine. Each RP service gateway in the TI system contains multiple Virtual Machines running on top of a Virtual Machine Monitor (VMM). These Virtual Machines run individual TEEVE sessions and are divided in two types: **Privileged Monitoring Virtual Machines** and **User Virtual Machines**. The first type of Virtual Machines contains monitoring and management tools required for the administration of the TI site and hence requires best-effort scheduling. The second type of VM's contains all the necessary disseminating infrastructures required to host one session and requires real-time scheduling.

To support this mixed requirement, TEEVE adapts Janus [58], a soft real-time cross layer CPU scheduling architecture as discussed in Section 3.1.2. This architecture allows services residing in each Virtual Machine to efficiently obtain reservation-based CPU guarantees within the VM and across VM's in the system, based on their individual QoS requirements. As shown in [58], it performs considerably better than XEN credit scheduler in terms of deadline guarantees, predictability and jitters among macro-frames.

4.2 Bandwidth management in TEEVE

TEEVE requires huge amounts of bandwidth for its multi-stream transmission. Streams are correlated since they come from multiple cameras capturing the same physical space from different viewpoints. Hence, TEEVE uses *view-based bandwidth management* [80] as discussed in "Multiple Flows Adaptive Bandwidth Management". that best serves the requirements in this multi-stream domain. This bandwidth management approach efficiently allocates available bandwidth while inducing low or no network congestion.

This scheme uses view-based bandwidth allocation, i.e., assigns more bandwidth to flows, which contribute more to the user's view than others. Figure 10 shows the impact of view-based bandwidth allocation (priority scheme) versus bandwidth management that does not differentiate among flows in the bundle (non-priority scheme). The impact of the bandwidth management is measured by the *Peak-Signal-Noise Ratio (PSNR)*, shown at the y-axis, of the overall joint macro-frame over time, shown at the x-axis (the time is expressed through macro-frame numbers as they are being played out). The results show an experiment [80] with 123D video streams in a bundle streaming between Illinois and Berkeley, and over time the amount of available network bandwidth (hence the target frame size TFS can get allocated less bits) has been gradually throttled. The graph clearly shows three major areas of interest: (a) area between 1-450 macro-frames, where there is plenty of network bandwidth (TFS is high) allocated, shows that each frame of each flow in the bundle has high PSNR, independent of the priority or non-priority scheme, and view-based differentiation of streams is not needed for bandwidth management, (b) area between 451–1000 macro-frames, where the network bandwidth (TFS decreases) gradually decreased, shows that view-based differentiation is needed and the priority scheme gets higher PSNR than non-priority scheme under the same certain network bandwidth availability, (c) area between 1001–1600 macro-frames, where low network bandwidth availability (TFS is low) is provided, shows that the view-differentiation does not help (i.e., PSNR of macro-frames

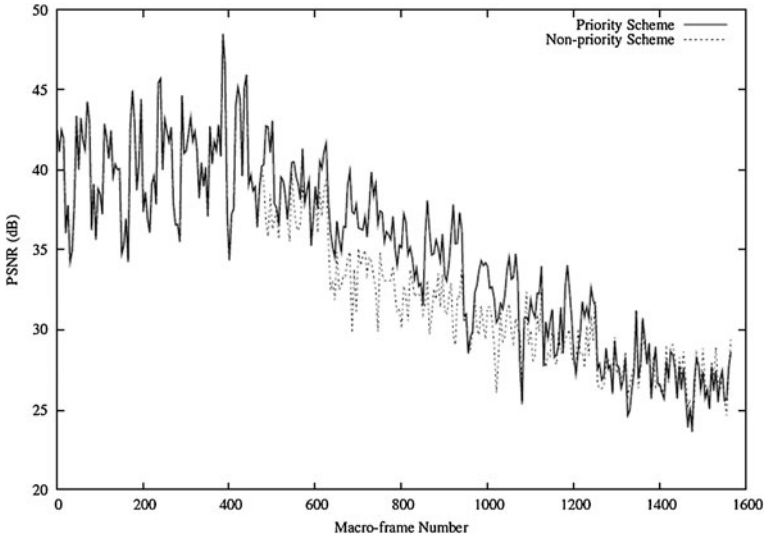

Fig. 10 Overall rendered quality for priority and non-priority bandwidth management schemes in TEEVE [80]

with priority and non-priority schemes are approximately the same) since at this point the system gets to a state with only 1–2 streams per bundle and with very low frame rate, hence there is not much information to differentiate over.

4.3 Delay management and synchronization in TEEVE

TEEVE system acquires delays in all three locations of the end-to-end system, the sender-side, network and receiver-side as shown in Fig. 11. Capturing devices (3D cameras) are the main sources of sender-side delays (D_{sender}) in TEEVE. The camera threads grab images from the camera buffers, and perform some pre-processing including rectification, moment computation, background subtraction, and edge extraction. After pre-processing, the images are copied to the computing threads. Instead of dividing the images to only two halves as in the original algorithm, TEEVE divide the images into multiple pieces for computing for parallel processing. It uses mesh-based 3D reconstruction [70] with reconstruction time

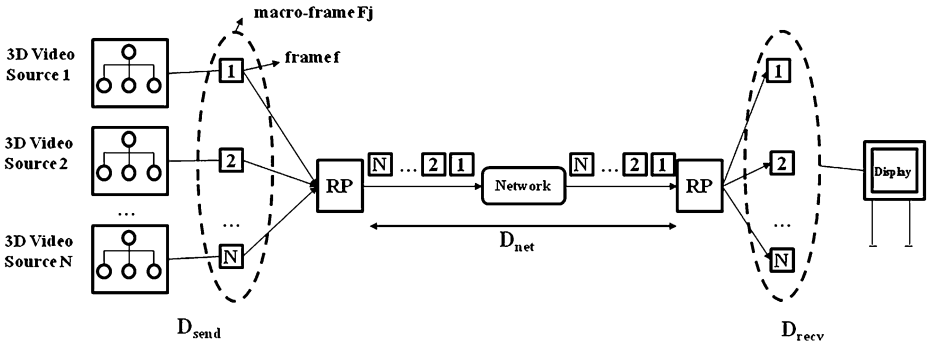

Fig. 11 End-to-end delay for TEEVE system

under 30*ms*. It offers a significant improvement of reconstruction time over point-could based approach used before in [75].

The renderer at the receiver-side selects a subset of most contributing streams to the user-selected view by computing the contribution factor *CF* of each stream with reference to the given user view [75] as discussed in "Multiple Flows Adaptive Bandwidth Management" and "Multi-stream synchronization delay management". TEEVE supports a selection of dynamic number of streams with *CF* larger than a threshold TH_{imp} with TH_{imp}= 0, thus the selected streams cover 180° of the scene and the receiver-side delay ($D_{receiver}$) incurred by 3D video rendering is about 37*ms*.

In TEEVE, the network delay (D_{net}) includes the **time interval** from the RP (gateway) *starting time of sending the first frame* 1 belonging to a macro-frame until the *end time of receiving the last frame N* belonging to the same macro-frame as shown in Fig. 11.

The end-to-end delay (D_{e2e}) of a macro-frame F_j includes then the sum of sender-side delays (D_{sender}), network delays (D_{net}) and receiver-side delays ($D_{receiver}$). The measurements between Illinois and Berkeley indicate that the end-to-end delay is around 200*ms* [Yang2010] as shown Fig. 12. The Fig. 12 shows two curves representing (a) a scheme with frames 1..*N*. of a macro-frame F_j are being scheduled from cameras using a token-ring scheme (traffic shaping occurs), and (b) a scheme with frames 1..*N* of a macro-frame F_j are being scheduled from cameras as soon as possible (no traffic shaping). We can see that traffic shaping helps the end-to-end delay and TEEVE achieves an end-to-end delay of 200 *ms* versus around 250–270 ms without traffic shaping. Note that this delay was sufficient to conduct extensive tele-immersive dance and sports activities [63, 76].

The current version of TEEVE includes one audio stream and multiple video streams, and the audio stream needs to be synchronized with multiple highly correlated video streams capturing different views of the same scene at different positions. Because of the multi-tier architecture in 3DTI, a sync skew between any two streams in one tier can be propagated to the next tier. Moreover, users can change views at any time [80]; hence a sync skew can be introduced between the audio streams and the new view. To solve these

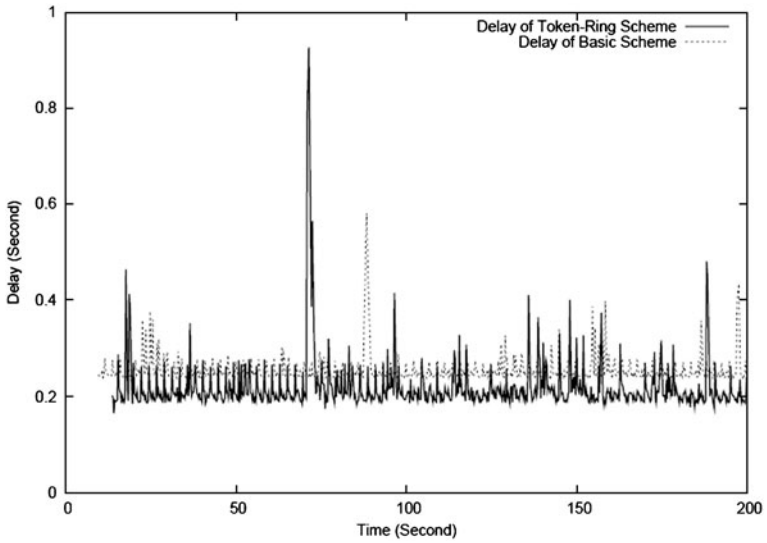

Fig. 12 Marco-frame end-to-end delay between Illinois and Berkeley [79]

issues, TEEVE adapts TSync [25], also discussed in "Multi-stream synchronization delay management", which can effectively minimize the multi-stream multi-source sync skews in TEEVE over Internet. It performs cooperative frame rate allocation with consideration of this synchronization issue. It can successfully bind the audio-visual sync skew within a recommended maximum 80 ms threshold [66].

5 Conclusion

Advanced DIME systems are emerging and with them new challenges such as (a) how to deal with large scale of devices deployed in them and (b) how to achieve real-time interactions and guarantees in distributed settings are also emerging. Solutions towards the above mentioned challenges lie in deploying appropriate QoS and resource management algorithms, protocols, and approaches. In this paper, we have surveyed existing DIME-relevant QoS and resource management techniques for two important resources, CPU computing and networking resources. Their appropriate coordination and organization leads the DIME performance improvement, hence towards interactivity and scale. The survey also provides taxonomy and comparison of the various QoS/resource management techniques since we expect that the future DIMEs will vary greatly in their capabilities, demands and setups.

We have presented one futuristic DIME system, the TEEVE 3DTI system, which has utilized various resource management techniques to understand performance limits we are facing when deploying adaptive QoS and resource management techniques over COTS components. The results clearly show that these types of DIME 3DTI systems are feasible for applications such as *Tele-immersive Dance (TED)* [63].

In the future, we expect that as more advanced DIME systems emerge, (a) COTS components get faster using some of the surveyed QoS and resource management techniques (i.e., techniques that are currently in research systems will move into the general purpose systems to allow for broad DIME usage), (b) hardware components, such as accelerators, compression cards, synchronizers, multi-GPUs, multi-core CPUs, become part of DIMEs in cost-effective manner, (c) Int-Serv and/or Diff-Serv like network services become available over different sizes of networks (LANs, MANs, WANs), and (d) advanced QoS and resource techniques and abstractions emerge to satisfy the new DIME demands.

References

1. Agarwal V, Rejaie R (2005) Adaptive multi-source streaming in heterogeneous peer-to-peer networks. In: Proceedings of the 12th Annual Multimedia Computing and Networking (MMCN '05)
2. Antoniades D, Athanatos M, Papagiannakis A, Markatos E, Dovrolis C (2006) Available bandwidth measurement as simple as running wget. In: Proceedings of Passive and Active Measurement (PAM'06)
3. Baccichet P, Bagni D, Chimienti A, Pezzoni L, Rovati F (2005) Frame concealment for H. 264/AVC decoders. IEEE Trans Consum Electron 51:227–233
4. Baker H, Bhatti N, Tanguay D, Sobel I, Gelb D, Goss M, Culbertson W, Malzbender T (2005) Understanding performance in Coliseum, an immersive videoconferencing system. ACM Trans Multimed Comput Commun Appl, vol 1
5. Banachowski SA, Brandt SA (2002) The BEST scheduler for integrated processing of best-effort and soft real-time processing. In: Proceedings of Multimedia Computing and Networking (MMCN'02)

6. Boutremans C, Boudec JYL (2003) Adaptive joint playout buffer and fec adjustment for internet telephony. In: Proceedings of 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'03), pp 652–662
7. Braden R, Zhang L, Berson S, Herzog S, Jamin S (1997) Resource ReSerVation Protocol (RSVP)—Version 1 Functional Specification, RFC 2205
8. Brandt SA, Banachowski S, Lin C, Bisson T (2003) Dynamic integrated scheduling of hard real-time, soft real-time and non-real-time processes. In: Proceedings of the 24th IEEE Real-Time Systems Symposium (RTSS'03), pp 396–407
9. Castro M, Druschel P, Kermarrec A-M, Rowstron AIT (2002) Scribe: a large-scale and decentralized application-level multicast infrastructure. IEEE J on Selected Areas in Communications (SAC'02) 20 (8):1489–1499
10. Castro DM, Kermarrec P, Nandi A-M, Rowstron A, Singh A (2003) SplitStream: high-bandwidth multicast in cooperative environments. J Operating Systems Review 37(5):298–313
11. Chang S, Zhong D, Kumar R (2001) Real-time content-based adaptive streaming of sports videos. Proceedings of IEEE Workshop on Content-based Access of Image and Video Libraries, In
12. Chu H-H, Nahrstedt K (1999) CPU service classes for multimedia applications. In: Proceedings of IEEE International Conference on Multimedia Computing and Systems (ICMCS'99), vol 1
13. Chu Y, Rao SG, Zhang H (2000) A case for end system multicast. In: Proceedings of ACM Annual Conference of the ACM Special Interest Group on Measurement and Modeling of Computer Systems (SIGMETRICS'00)
14. Cisco telepresence. http://www.Cisco.com/TelePresence
15. Daniilidis F, Mulligan J, Mckendal R, Majumder A, Kamberova G, Schid D, Bajcsy R, Fuchs H (1999) Towards the holodeck: an initial testbed for real-time 3D teleimmersion. In: Proceedings of Annual Conference of the ACM Special Interest Group on Computer Graphics and Interactive Techniques (SIGGRAPH'99)
16. Deering S (1988) Multicast routing in internetworks and extended LANs. In: Proceedings of Annual Conference of the ACM Special Interest Group on Data Communication (SIGCOMM'88), pp 55–64
17. Fernandez-Escribano G, Kalva H, Cuenca P, Orozco-Barbosa L (2006) Speeding-up the macroblock partition mode decision in MPEG-2/H.264 transcoding. In: Proceedings of IEEE Conference on Image Processing (ICIP'06), pp 869–872
18. Gautier L, Diot C (1998) Design and evaluation of mimaze, a multi-player game on the internet. In: Proceedings of IEEE Multimedia Systems Conference, pp 233–236
19. Goldenstein S (1999) Time warping of audio signals. In: Proceedings of IEEE Conference on Computer Graphics, pp 52–57
20. Hellerstein JL (1993) Achieving service rate objectives with decay usage scheduling. IEEE Trans Softw Eng
21. Hosseini M, Georganas ND (2003) Design of a multi-sender 3D videoconferencing application over an end system multicast protocol. In: Proceedings of the eleventh ACM international conference on Multimedia (MM'03)
22. HP Halo System http://h71028.www7.hp.com/enterprise/us/en/halo/index.html
23. Huang C-M, Lin C-W, Yang C-C, Chang C-H, Ku H-H (2009) An SVC-MDC video coding scheme using the multi-core parallel programming paradigm for P2P video streaming. In: Proceedings of IEEE International Conference on Computer Science and Application (ICCSA'09)
24. Huang Y-L, Shen Y-C, Wu J-L (2009) Scalable computation for spatially scalable video coding using NVIDIA CUDA and multi-core CPU. In: Proceedings of ACM Multimedia (MM'09)
25. Huang Z, Wu W, Nahrstedt K, Arefin A, Rivas R (2010) TSync: a new synchronization framework for multi-site 3D tele-immersion. In: Proceedings of International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'10)
26. Hwang FK, Richards DS (1992) Steiner tree problems. Networks 22:55–89
27. IEEE 1588 standard (2008) Precise time synchronization as the basis for real time applications in automation.
28. ITU-G.114 (2003) One-way transmission time.
29. Jain M, Dovrolis C (2002) Pathload: a measurement tool for end-to-end available bandwidth. In: Proceedings of the 3rd Passive and Active Measurements (PAM '02)
30. Jain M, Dovrolis C (2003) End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput. IEEE/ACM Trans Netw
31. Jeffay K, Smith FD, Moorthy A, Anderson J (1998) Proportional share scheduling of operating system services for real-time applications. In: Proceedings of Real-Time Systems Symposium (RTSS'98), vol. 0
32. Jia X (1998) A distributed algorithm of delay-bounded multicast routing for multimedia applications in wide area networks. IEEE/ACM Transaction on Networking 6:828–837

33. Jones MB, Leach PJ, Draves RP, Barrera I (1995) Modular real-time resource management in the rialto operating system. In: Proceedings of the Fifth Workshop on Hot Topics in Operating Systems (HotOS'95)
34. Kompella VP, Pasquale JC, Polyzos GC (1993) Multicast routing for multimedia communication. IEEE/ACM Trans Netw
35. Kostic D, Rodriguez A, Albrecht J, Vahdat A (2003) Bullet: high bandwidth data dissemination using an overlay mesh. ACM SIGOPS Operating Systems Review 3(5)
36. Kurillo G, Vasudevan R, Lobaton E, Bajcsy E (2008) A framework for collaborative real-time 3D teleimmersion in a geographically distributed environment. In: Proceedings of IEEE International Symposium on Multimedia (ISM'08)
37. Lee H (1997) A proportional-share scheduler for multimedia applications. In: Proceedings on International Conference on Multimedia Computing and Systems (ICMCS'97)
38. Lee X, Zhang Y (1996) A fast hierarchical motion-compensation scheme for video coding using block feature matching. IEEE Transaction on Circuits and Systems for Video Technology 6:627–635
39. Liang YJ, Faber N, Girod B (2003) Adaptive playout scheduling and loss concealment for voice communication over IP networks. IEEE Transaction on Multimedia 5(4):532–543
40. Little T (1993) A framework for synchronous delivery of time-dependent multimedia data. Multimedia Systems 1(2):87–94
41. Liu T, Choudary C (2004) Real-time content analysis and adaptive transmission of lecture videos for mobile applications. In: Proceedings of the 12th annual ACM international conference on Multimedia (MM'04), pp 400–403
42. Liu L-K, Feig E (1996) A block-based gradient descent search algorithm and block motion estimation in video coding. IEEE Transaction on Circuits and Systems for Video Technology 6(4):419–421
43. Liu CL, Layland JW (1973) Scheduling algorithms for multiprogramming in a hard real-time environment. J ACM 20(1):46–61
44. Liu T, Nelakuditi S (2004) Disruption-tolerant content-aware video streaming. In: Proceedings of ACM Multimedia (MM'04)
45. Liu J, Niu Z (2004) An adaptive receiver buffer adjust algorithm for VoIP applications considering voice characters. In: Proceedings of 10th Asia-Pacific Conference on Communications and 5th International Symposium on Multi-Dimensional Mobile Communication, pp 597–601
46. Liu H, Zarki M (2006) An adaptive delay and synchronization control scheme for Wi-Fi based audio/video conferencing. Springer Wireless Networks 12(4):511–522
47. Microsoft NetMeeting, http://www.microsoft.com.
48. Nahrstedt K, Qiao L (1997) Stability and adaptation control for lip synchronization skews. Technical Report, University of Illinois
49. Nahrstedt K, Steinmetz R (1995) Resource management in multimedia systems. IEEE Computer 28(5):52–65
50. Nahrstedt K, Chu H, Narayan S (1998) QoS-aware resource management for distributed multimedia applications. J on High-Speed Networking 8(3):227–255
51. Narbutt M, Kelly A, Murphy L, Perry P (2005) Adaptive VoIP playout scheduling: assessing user satisfaction. IEEE Internet Computing 9(4):28–34
52. Nguyen H, Rivas R, Nahrstedt K (2009) iDSRT: Integrated dynamic soft real-time architecture for critical infrastructure data delivery over wlan. In: Proceedings of International ICST conference on Heterogeneous Networking for Quality, Reliability, Security, and Robustness (QShine'09)
53. Nichols K, Blake S, Baker F, Black D (1998) Definition of the differentiated services field (DS Field) in the IPv4 and IPv6 headers, RFC 2474
54. Nieh J, Lam MS (2003) A smart scheduler for multimedia applications. ACM Trans Computer Systems 21(2):117–163
55. POSIX (1992) Realtime extension for portable operating systems (posix 1003.4). Technical Report
56. Rajkumar R, Juvva K, Molano A, Oikawa S (1998) Resource kernels: A resource-centric approach to real-time and multimedia systems. In: Proceedings of the SPIE/ACM Conference on Multimedia Computing and Networking (MCN'98), pp 150–164
57. Ramjee R, Kurose J, Towsley D, Schulzrinne H (2004) Adaptive playout mechanisms for packetized audio applications in wide-area networks. Proceedings of 13th IEEE Annual Joint Conf on Networking for Global Communication (GLOBECOM'04) 2:680–688
58. Rivas R, Arefin A, Nahrstedt K (2010) Janus: a cross-layer soft real-time architecture for virtualization. In: Proceedings of the 4th International Workshop on Virtualization Technologies in Distributed Computing (VTDC'10)
59. Russinovich M (2007) Inside the windows vista kernel: Part 1. Technet Magazine

60. Saroiu S, Gummadi P, Gribble S (2002) SProbe: a fast technique for measuring bottleneck bandwidth in uncooperative environments. In: Proceedings of IEEE International Conference on Computer Communications (INFOCOM'02)
61. Sat B, Huang Z, Wah BW (2007) The design of a multi-party VoIP conferencing system over the internet. In: Proceedings of IEEE International Symposium on Multimedia (ISM'07)
62. Savage S (1999) Sting: a TCP-based network measurement tool. In: Proceedings of USENIX Symposium on Internet Technologies and Systems (SITS'99)
63. Sheppard R, Kamali M, Rivas R, Tamai M, Yang Z, Wu W, Nahrstedt K (2008) Advancing interactive collaborative mediums through tele-immersive dance (TED): a symbiotic creativity and design environment for art and computer science. In: Proceedings of ACM International Conference on Multimedia (MM'08)
64. Shin I, Lee I (2003) Periodic resource model for compositional real-time guarantees. In: Proceedings of the 24th IEEE International Real-Time Systems Symposium (RTSS'03)
65. Skype. http://www.skype.com
66. Steinmetz R (1996) Human perception of jitter and media synchronization. IEEE Journal on Selected Areas in Communications 14(1):61–72
67. Stoica I, Abdel-Wahab H (1995) Earliest eligible virtual deadline first: A flexible and accurate mechanism for proportional share resource allocation. Old Dominion University, Technical Report
68. Tham JY, Ranganath S, Ranganath M, Kassim AA (1998) A novel unrestricted center biased diamond search algorithm for block motion estimation. IEEE Transaction on Circuits and Systems for Video Technology 8(4):369–377
69. Tommasi F, Molendini S (2000) Some extensions to enhance the scalability of the RSVP protocol. Internet Draft
70. Vasudevan R, Lobaton E, Kurillo G, Bajcsy R et al (2010) A methodology for remote virtual interaction in teleimmersive environments. In: Proceedings of the first annual ACM SIGMM Conference on Multimedia Systems (MMSys'10), pp 281–292
71. Vickers BJ, Albuquerque C, Suda T (2000) Source-adaptive multilayered multicast algorithms for real-time video distribution. IEEE/ACM Transaction of Network 8(6):720–733
72. Wah BW, Lin D (1999) Transformation-based reconstruction for real-time voice transmissions over the Internet. IEEE Transaction on Multimedia 1(4):342–351
73. Waldspurger C (1995) Lottery and stride scheduling: Flexible proportional-share resource management. Dissertation, MIT
74. Wang L, Terzis A, Zhang L (1999) RSVP refresh overhead reduction by state compression, Internet Draft
75. Wu W, Yang Z, Nahrstedt K (2008) Implementing a distributed 3D tele-immersive system. In: Proceedings of IEEE International Symposium on Multimedia (ISM'08)
76. Wu W, Yang Z, Nahrstedt K (2008) A study of visual context representation and control for remote sport learning tasks. In: Proceedings of AACE World Conference on Educational Multimedia, Hypermedia and Telecommunications (ED-MEDIA'08)
77. Wu W, Arefin A, Rivas R, Yang Z, Sheppard R, Nahrstedt K (2009) Quality of experience in distributed interactive multimedia environments: Toward a theoretical framework. In: Proceedings of ACM Multimedia (MM'09)
78. Xin J, Vetro A, Sun H (2004) Efficient macroblock coding-mode decision for H.264/AVC video coding. In: Proceedings of Picture Coding Symposium
79. Yang Z, Cui Y, Yu B, Liang J, Nahrstedt K, Jung SH, Bajscy R (2005) TEEVE: The next generation architecture for tele-immersive environments. In: Proceedings of IEEE International Symposium on Multimedia (ISM'05)
80. Yang Z, Yu B, Nahrstedt K, Bajcsy R (2006) A Multi- stream adaptation framework for bandwidth management in 3D tele- immersion. In: Proceedings of International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'06)
81. Yeung A, Liew SC (1997) Multiplexing video track using frame-skipping aggregation technique. Proceedings of International Conference on Image Processing (ICIP'97) 1:334–337
82. Zhu Q, Garcia-Luna-Aceves J (1995) A source-based algorithm for delay-constrained minimum-cost multicasting. In: Proceedings of IEEE International Conference on Computer Communications (INFOCOM'95)

Klara Nahrstedt is a full professor at the University of Illinois at Urbana-Champaign, Computer Science Department. Her research interests are directed toward Multimedia Distributed Systems and Networking, Peer-to-peer Streaming Systems, Quality of Service (QoS) and Resource Management in Internet and Mobile Wireless Networks, Real-Time Security in Wireless Networks, and Tele-Immersive Systems. She is the recipient of the Early NSF Career Award, the Junior Xerox Award, the IEEE Communication Society Leonard Abraham Award for Research Achievements, the University Scholar Award (2008–2009) and the Humboldt Research Award. She was the editor-in-chief of the ACM/Springer Multimedia Systems Journal (2000–2007), associate editor of ACM Transactions on Multimedia Computing, Communications and Applications since 2005, associate editor of IEEE Transactions on Information Forensics & Security since 2005, general co-chair of ACM Multimedia 2006, general chair of ACM NOSSDAV 2007, general chair of IEEE PerCom 2009, and Ralph and Catherine Fisher Professor (2002–2010). She was elected to serve as the chair of the ACM SIG Multimedia (2007–2011). Klara Nahrstedt received her BA in mathematics from Humboldt University, Berlin, in 1984, and M.Sc. degree in numerical analysis from the same university in 1985. She was a research scientist in the Institute for Informatik in Berlin until 1990. In 1995 she received her PhD from the University of Pennsylvania in the Department of Computer and Information Science. She is the member of ACM and IEEE Fellow.

Ahsan Arefin is a PhD student in Computer Science at the University of Illinois at Urbana-Champaign. His research interests include Management and Analysis of Large-Scale Distributed Systems focusing Distributed Multimedia Systems, Quality of Service Management, Large-scale Monitoring and Data Mining. He is currently a member of SIGMM educational committee. Ahsan Arefin received his B.Sc. in Computer Science and Engineering from Bangladesh University of Engineering and Technology (BUET), Bangladesh in 2006. He received various scholarships including merit list from Government of Bangladesh and awards for programming and design competitions during his graduate and undergraduate studies.

Raoul Rivas is a PhD student in Computer Science at University of Illinois at Urbana-Champaign. His research interests are in the area of Operating Systems and Multimedia Systems including Virtualization Techniques, Quality of Service, Soft-Real-time Scheduling, Power Management and Heterogeneous Architectures. He has received various scholarships including a merit based scholarship from Secretaría de Educación (government board of education) in Mexico and the Intel Undergraduate Research Scholarship in 2006. Raoul Rivas received a B.S. in Computer Science from the University of Illinois at Urbana-Champaign in 2007, and a B.S in Systems Engineering from Universidad Panamericana in Mexico City, Mexico in 2005.

Pooja Agarwal is currently a PhD student in Computer Science at University of Urbana-Champaign. She received her undergraduate degree BTech in Computer Science and Engineering from Manipal Institute of Technology, India. Before joining to PhD, she worked as an intern at Indian Institute of Technology at Kanpur, India. Her research interests include different networking and overlay aspects in Distributed Interactive Multimedia Systems focusing on Bandwidth Management, Delay Management and Peer-to-peer Streaming.

Zixia Huang is currently a PhD candidate at University of Illinois at Urbana-Champaign, Computer Science department. His research areas include Multimedia Streaming, Cloud Computing, Mobile Networks and Parallel Computing. He received his bachelor degree in Electronics and Information Engineering from Shanghai Jiao Tong University in 2006 and his M.Sc. degree in electrical and computer engineering from University of Illinois in 2009. He has been the recipient of several awards including the Global Electric (GE) Scholarship and the Pan Wen-Yuan Student Award.

Wanmin Wu completed her undergraduate studies at Zhejiang University in 2004. Her research interests include Multimedia Processing and Communication, Distributed Systems, Human-Computer Interface, and Tele-Immersion. She is currently a PhD student in Computer Science at University of Illinois at Urbana-Champaign. She was selected for IBM Watson Emerging Leaders in Multimedia in 2008. She has received various scholarships and fellowships including Yee Memorial Fund Fellowship for year 2010–2011.

Zhenyu Yang is working as an assistant professor in school of Computing and Information Sciences at Florida International University. His research interests include Networking and Distributed systems with a focus on Communicative, Collaborative and Multimedia Systems involving Quality-of-Service Management, Overlay and Peer-to-peer Networks, Content Delivery Network, Wireless Networks, 3D Collaborative Environments, Ubiquitous Computing, Human-Computer Interface/Interaction, and Development of Multimedia Applications particularly to promote collaborative and interdisciplinary research. He completed his PhD from the department of Computer Science at University of Illinois at Urbana-Champaign.