PROCEEDINGS

ACM Multimedia '95

San Francisco, California, November 5-9, 1995

Sponsored by the ACM SIG MULTIMEDIA, SIGCHI, SIGGRAPH, SIGMIS, SIGBIO, SIGCOMM, SIGIR AND SIGOIS, in cooperation with ACM SIGAPP, SIGCAPH, SIGMOD and SIGOPS

The Association for Computing Machinery, Inc.
1515 Broadway
New York, NY 10036

...In proceedings of the ACM Multi-media, 1995 (San Francisco, CA, USA, November 5-9, 1995) ACM, New York, 1995, pp. 23-35.

Ordering Information

Nonmembers
Nonmember orders placed within the U.S. should be directed to:

Addison-Wesley Publishing Company
Order Department
Jacob Way
Reading, MA 01867
Tel: +1 800 447 2226

Addison-Wesley will pay postage and handling on orders accompanied by check. Credit card orders may be placed by mail or by calling the Addison-Wesley Order Department at the number above. Follow-up inquiries should be directed to the Customer Service Department at the same number. Please include the Addison-Wesley ISBN number with your order:

A-W ISBN 0-201-87774-0

Nonmember orders from outside the U.S. should be addressed as noted below:

Europe/Middle East

Addison-Wesley Publishing Group
Concertgebouwplein 25
1071 LM Amsterdam
The Netherlands
Tel: +31 20 6717296
Fax: +31 20 6645334

Germany/Austria/Switzerland

Addison-Wesley Verlag Deutschland GmbH
Hildachstrasse 15d
Wachsbleiche 7-12
53111 Bonn
Germany
Tel: +49 228 98 515 0
Fax: +49 228 98 515 99

United Kingdom/Africa

Addison-Wesley Publishers Ltd.
Finchampstead Road
Wokingham, Berkshire RG11 2NZ
United Kingdom
Tel: +44 734 794000
Fax: +44 734 794035

After January 1, 1996:
Addison-Wesley Longman Publishers, Ltd.
Longman House
Burnt Mill
Harlow, Essex CM202JE
United Kingdom
Tel: +44 1279 623 623
Fax: +44 1279 431 059

Asia

Addison-Wesley Singapore Pte. Ltd.
15 Beach Road
#05-02/09/10 Beach Centre
Singapore 0718
Tel: +65 339 7503
Fax: +65 339 9709

Japan

Addison-Wesley Publishers Japan Ltd.
Nichibo Building
1-2-2 Sarugakucho
Chiyoda-ku, Tokyo 101
Japan
Tel: +81 33 291 4581
Fax: +81 33 291 4592

Australia/New Zealand

Addison-Wesley Publishers Pty. Ltd.
6 Byfield Street
North Ryde, N.S.W. 2113
Australia
Tel: +61 2 878 5411
Fax: +61 2 878 5830

Latin America

Addison-Wesley Iberoamericana S.A.
Boulevard de las Cataratas #3
Colonia Jardines del Pedregal
Delegacion Alvaro Obregon
01900 Mexico D. F.
Tel: +52 5 660 2695
Fax: +52 5 660 4930

Canada

Addison-Wesley Publishing (Canada) Ltd.
26 Prince Andrew Place
Don Mills, Ontario M3C 2T8
Canada
Tel: +416 447 5101
Fax: +416 443 0948

ACM Members
A limited number of copies are available at the ACM member discount. Send order with payment in US dollars to:

ACM Order Department
P.O. Box 12114
Church Street Station
New York, N.Y. 10257

OR

ACM European Service Center
Avenue Marcel Thiry 204
1200 Brussels
Belgium

Credit card orders from U.S.A. and Canada:
+1 800 342 6626

New York Metropolitan Area and outside of the U.S.:
+1 212 626 0500
Fax +1 212 944-1318
Email:acmpubs@acm.org

Please include your ACM member number and the ACM Order number with your order.

ACM Order Number: 433952
ACM ISBN: 0-89791-751-0

vic: A Flexible Framework for Packet Video

Steven McCanne
University of California, Berkeley
and Lawrence Berkeley Laboratory
mccanne@ee.lbl.gov

Van Jacobson
Network Research Group
Lawrence Berkeley Laboratory
van@ee.lbl.gov

ABSTRACT

The deployment of IP Multicast has fostered the development of a suite of applications, collectively known as the MBone tools, for real-time multimedia conferencing over the Internet. Two of these tools — *nv* from Xerox PARC and *ivs* from INRIA — provide video transmission using software-based codecs. We describe a new video tool, *vic*, that extends the groundbreaking work of nv and ivs with a more flexible system architecture. This flexibility is characterized by network layer independence, support for hardware-based codecs, a conference coordination model, an extensible user interface, and support for diverse compression algorithms. We also propose a novel compression scheme called "Intra-H.261". Created as a hybrid of the nv and ivs codecs, Intra-H.261 provides a factor of 2-3 improvement in compression gain over the nv encoder (6 dB of PSNR) as well as a substantial improvement in run-time performance over the ivs H.261 coder.

KEYWORDS

Conferencing protocols; digital video; image and video compression and processing; multicasting; networking and communication.

1 INTRODUCTION

Over the past few years, a collaborative effort in the network research community has produced a suite of tools for multimedia conferencing over the Internet [16, 25, 26, 39, 42]. The driving force behind these tools is Deering's IP Multicast [11], a technology which extends the traditional IP routing model for efficient multipoint packet delivery. The incremental deployment of IP Multicast has been realized by building a (temporary) virtual multicast network on top of the existing Internet, which Casner has dubbed the Multicast Backbone, or MBone [6].

Multimedia '95, San Francisco, CA USA
0-89791-751-0/95/11

The first applications to provide video over the MBone were the Xerox PARC Network Video tool, *nv*, and the INRIA Video Conferencing System, *ivs*. While these two systems share the goal of supporting low bit-rate multicast video over the Internet, their approaches are markedly different. Ivs is an integrated audio/video conferencing system that relies exclusively on H.261 [45] for video compression. By adopting a standardized algorithm, ivs can interoperate with a large installed base of H.320 video codecs as an H.320-compliant bit stream is easily generated from an H.261 packet stream by introducing H.221 framing in software [20].

In contrast, nv is a "video-only" application that utilizes a custom coding scheme tailored specifically for the Internet and targeted for efficient software implementation [17]. Because of its low computational complexity, the nv codec can run much faster than an H.261 codec. Even though H.261 has better compression performance than nv, nv is more often used by the MBone community because of its better run-time performance.

Inevitably, in pioneering work such as nv and ivs, restrictions must be imposed on the design process to facilitate experimentation. For example, the ivs design assumes video is represented as 4:1:1-decimated CCIR-601 YUV, while nv assumes 4:2:2 decimation. Extending nv to support H.261 or ivs to support nv-style compression would require non-trivial design changes. Also, since both systems are based on software compression, their video capture models were designed around uncompressed video. Extending either to support hardware-based compression engines would be relatively difficult.

In this paper, we describe a third model for a packet video application, realized in the UCB/LBL video conferencing tool, *vic*. Vic builds upon the lessons learned from ivs and nv by focusing on flexibility. It is an extensible, object-oriented, application framework that supports

- multiple network abstractions,
- hardware-based codecs,
- a conference coordination model,
- an extensible user interface, and
- diverse video compression algorithms.

Moreover, we have combined Frederick's insights from the nv codec with the compression advantages and standards

compliance of H.261 in a novel scheme which we call *Intra-H.261*. Intra-H.261 gives significant gain in compression performance compared to nv and substantial improvement in both run-time performance and packet-loss tolerance compared to ivs.

Vic was originally conceived as an application to demonstrate the Tenet real-time networking protocols [14] and to simultaneously support the evolving "Lightweight Sessions" architecture [24] in the MBone. It has since driven the evolution of the Real-time Transport Protocol (RTP) [40]. As RTP evolved, we tracked and implemented protocol changes, and fed back implementation experience to the design process. Moreover, our experience implementing the RTP payload specification for H.261 led to an improved scheme based on macroblock-level fragmentation, which resulted in a revised protocol [44]. Finally, the RTP payload specification for JPEG [13] evolved from a vic implementation.

In the next section, we describe the design approach of the MBone tools. We then discuss the essentials of the vic network architecture. The network architecture shapes the software architecture, which is discussed in the following section. Finally, we discuss signal compression issues, deployment and implementation status.

2 COMPOSABLE TOOLS VS. TOOLKITS

A cornerstone of the Unix design philosophy was to avoid supplying a separate application for every possible user task. Instead, simple, one-function "filters" like *grep* and *sort* can be easily and dynamically combined via a "pipe" operator to perform arbitrarily complex tasks. Similarly, we use modular, configurable applications, each specialized to support a particular media, which can be easily composed via a *Conference Bus* to support the variety of conferencing styles needed to support effective human communication. This approach derives from the framework proposed by Ousterhout in [33], where he claims that large systems are easily composed from small tools that are glued together with a simple communication primitive (e.g., the Tk *send* command). We have simply replaced his *send* primitive with a well-defined (and more restrictive) Conference Bus protocol. Restricting the protocol prevents the evolution of sets of tools that rely on the specifics of each other's internal implementations. In addition to vic, our conferencing applications include the Visual Audio Tool (vat) for audio [26], a whiteboard (wb) for shared workspace and slide distribution [25, 15], and the Session Directory (sd) for session creation and advertisement [23].

This "composable tools" approach to networked multimedia contrasts with the more common "toolkit framework" adopted by other multimedia systems [10, 31, 37, 38]. Toolkits provide basic building blocks in the form of a code library with an application programming interface (API) to that library providing high-level abstractions for manipulating multimedia data flows. Each distinct conferencing style requires a different application but the applications are typically simple to write, consisting mostly of API calls with style-dependent glue and control logic.

The toolkit approach emphasizes the programming model and many elegant programming mechanisms have resulted from toolkit-related research. To simplify the programming model, toolkits usually assume that communication is application independent and offer a generic, least-common-denominator network interface built using traditional transport protocols.

In 1990 Clark and Tennenhouse [8] pointed out that multimedia applications could be simplified and both application and network performance enhanced if the network protocol reflected the application semantics. Their model, Application Level Framing (ALF), is difficult to implement with toolkits (where application semantics are deliberately "factored out") but is the natural way to implement "composable tools". And ALF-based, media-specific tools offer a simple solution to multimedia's biggest problem — high rate, high volume, continuous media data streams. Since the tools are directly involved in processing the multimedia data flows, we can use ALF to tune all the performance-critical multimedia data paths within the application and across the network.

In addition to performance, flexibility is gained by composing simple tools rather than using a monolithic application built on top of some API. Since each tool deals directly with its media stream and sends only low-rate reports like "X has started/stopped sending" on the Conference Bus, the coordination agent necessary to implement a particular conferencing scenario can be written in a simple interpreted language like *Tcl* [34]. This allows the most volatile part of the conferencing problem, the piece that melds audio, video, etc., into a coordinated unit that meets particular human needs and expectations, to be simple and easy to evolve. It also ensures that the coordination agents are designed orthogonal to the media agents, enforcing a mechanism/policy separation: media tools implement the mechanism by which coordination tools impose the policy structure appropriate for some particular conferencing scenario, e.g., open meeting, moderated meeting, class, seminar, etc.

3 NETWORK ARCHITECTURE

While the freedom to explore the communications protocol design space fosters innovation, it precludes interoperability. Since the MBone was created to study multicast scaling issues, interoperability is especially important. Multicast use at an interesting scale requires that a large group of people spread over a large geographic region have some reason to send and receive data from the group. One good way to achieve this is to develop interoperable applications that encourage widespread use.

3.1 RTP

To promote such interoperability, the Audio/Video Transport Working group of the Internet Engineering Task Force (IETF) has developed RTP as an application level protocol for multimedia transport. The goal is to provide a very thin transport layer without overly restricting the application designer. The protocol specification itself states that "RTP is intended to be malleable to provide the information required by a particular application and will often be integrated into the application processing rather than being implemented as a separate layer." In the ALF spirit, the semantics of sev-

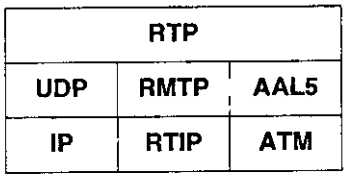

Figure 1: RTP and the Protocol Stack

eral of the fields in the RTP header are deferred to an "RTP Profile" document, which defines the semantics according to the given application. For example, the RTP header contains a generic "marker" bit that in an audio packet indicates the start of a talk spurt but in a video packet indicates the end of a frame. The interpretation of fields can be further refined by the "Payload Format Specification". For example, an audio payload might define the RTP timestamp as a audio sample counter while the MPEG/RTP specification [22] defines it as the "Presentation Time Stamp" from the MPEG system specification.

Because of its ALF-like model, RTP is a natural match to the "composable tools" framework and serves as the foundation for vic's network architecture. Since RTP is independent of the underlying network technology, vic can simultaneously support multiple network protocols. Figure 1 illustrates how RTP fits into several protocol stacks. For IP and IP Multicast, RTP is layered over UDP, while in the Tenet protocols, it runs over RMTP/RTIP [2]. Similarly, vic can run directly over an ATM Adaptation Layer. In all these cases, RTP is realized in the application itself.

RTP is divided into two components: the data delivery protocol, and the control protocol, RTCP. The data delivery protocol handles the actual media transport, while RTCP manages control information like sender identification, receiver feedback, and cross-media synchronization. Different media of the same conference-level session are distributed on distinct RTP sessions.

Complete details of the RTP specification are provided in [40]. We briefly mention one feature of the protocol relevant to the rest of the paper. Because media are distributed on independent RTP sessions (and because vic is implemented independently of other multimedia applications), the protocol must provide a mechanism for identifying relationships among media streams (e.g., for audio/video synchronization). Media sources are identified by a 32-bit RTP "source identifier" (SRCID), which is guaranteed to be unique only within a single session. Thus, RTP defines a canonical-name (CNAME) identifier that is globally unique across all sessions. The CNAME is a variable-length, ASCII string that can be algorithmically derived, e.g., from user and host names. RTCP control packets advertise the mapping between a given source's SRCID and variable-length CNAME. Thus, a receiver can group distinct RTP sources via their CNAME into a single, logical entity that represents a given session participant.

In summary, RTP provides a solid, well-defined protocol framework that promotes application interoperability, while its ALF philosophy does not overly restrict the application design and, in particular, lends itself to efficient implementation.

4 SOFTWARE ARCHITECTURE

The principles of ALF drove more than the vic network architecture; they also determined the overall software architecture. Our central goal was to achieve a flexible software framework which could be easily modified to explore new coding schemes, network models, compression hardware, and conference control abstractions. By basing the design on an objected-oriented ALF framework, we achieved this flexibility without compromising the efficiency of the implementation.

ALF leads to a design where data sources and sinks within the application are highly aware of how data must be represented for network transmission. For example, the software H.261 encoder does not produce a bit stream that is in turn packetized by an RTP agent. Instead, the encoder builds the packet stream fragmented at boundaries that are optimized for the semantics of H.261. In this way, the compressed bit stream can be made more robust to packet loss.

At the macroscopic level, the software architecture is built upon an event-driven model with highly optimized data paths glued together and controlled by a flexible Tcl/Tk [34] framework. A set of basic objects is implemented in C++ and are coordinated via Tcl/Tk. Portions of the C++ object hierarchy mirror a set of object-oriented Tcl commands. C++ base classes permit Tcl to manipulate objects and orchestrate data paths using a uniform interface, while derived subclasses support specific instances of capture devices, display types, decoder modules, etc. This division of low-overhead control functionality implemented in Tcl and performance critical data handling implemented in C++ allows for rapid prototyping without sacrifice of performance. A very similar approach was independently developed in the VuSystem [29].

4.1 Decode Path

Figure 2 roughly illustrates the receive/decode path. The elliptical nodes correspond to C++ base classes in the implementation, while the rectangular nodes represent output devices. A Tcl script is responsible for constructing the data paths and performing out-of-band control that might result from network events or local user interaction. Since Tcl/Tk also contains the user interface, it is easy to present control functionality to the user as a single interface element that might invoke several primitive control functions to implement its functionality.

The data flow through the receive path is indicated by the solid arrows. When a packet arrives from the network, the Network object dispatches it to the Demuxer which implements the bulk of the RTP processing. From there, the packet is demultiplexed to the appropriate Source object, which represents a specific, active transmitter in the multicast session. If no Source object exists for the incoming packet, an upcall into Tcl is made to instantiate a new data path for that source. Once the data path is established, packets flow from the source object to a decoder object. Hardware and soft-

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.