52. IWK

Internationales Wissenschaftliches Kolloquium
International Scientific Colloquium

PROCCEDINGS | 10 - 13 September 2007

FACULTY OF COMPUTER SCIENCE AND AUTOMATION

COMPUTER SCIENCE MEETS AUTOMATION

VOLUME II

TECHNISCHE UNIVERSITÄT ILMENAU

Bibliografische Information der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der deutschen Nationalbiografie; detaillierte bibliografische Daten sind im Internet über **http://dnb.ddb.de** abrufbar.

ISBN 978-3-939473-17-6

Impressum

Herausgeber:	Der Rektor der Technischen Universität Ilmenau Univ.-Prof. Dr. rer. nat. habil. Peter Scharff
Redaktion:	Referat Marketing und Studentische Angelegenheiten Kongressorganisation Andrea Schneider Tel.: +49 3677 69-2520 Fax: +49 3677 69-1743 e-mail: kongressorganisation@tu-ilmenau.de
Redaktionsschluss:	Juli 2007
Verlag:	Technische Universität Ilmenau/Universitätsbibliothek Universitätsverlag Ilmenau Postfach 10 05 65 98684 Ilmenau www.tu-ilmenau.de/universitaetsverlag
Herstellung und Auslieferung:	Verlagshaus Monsenstein und Vannerdat OHG Am Hawerkamp 31 48155 Münster www.mv-verlag.de
Layout Cover:	www.cey-x.de
Bezugsmöglichkeiten:	Universitätsbibliothek der TU Ilmenau Tel.: +49 3677 69-4615 Fax: +49 3677 69-4602

Preface

Dear Participants,

Confronted with the ever-increasing complexity of technical processes and the growing demands on their efficiency, security and flexibility, the scientific world needs to establish new methods of engineering design and new methods of systems operation. The factors likely to affect the design of the smart systems of the future will doubtless include the following:

- As computational costs decrease, it will be possible to apply more complex algorithms, even in real time. These algorithms will take into account system nonlinearities or provide online optimisation of the system's performance.
- New fields of application will be addressed. Interest is now being expressed, beyond that in "classical" technical systems and processes, in environmental systems or medical and bioengineering applications.
- The boundaries between software and hardware design are being eroded. New design methods will include co-design of software and hardware and even of sensor and actuator components.
- Automation will not only replace human operators but will assist, support and supervise humans so that their work is safe and even more effective.
- Networked systems or swarms will be crucial, requiring improvement of the communication within them and study of how their behaviour can be made globally consistent.
- The issues of security and safety, not only during the operation of systems but also in the course of their design, will continue to increase in importance.

The title "Computer Science meets Automation", borne by the 52nd International Scientific Colloquium (IWK) at the Technische Universität Ilmenau, Germany, expresses the desire of scientists and engineers to rise to these challenges, cooperating closely on innovative methods in the two disciplines of computer science and automation.

The IWK has a long tradition going back as far as 1953. In the years before 1989, a major function of the colloquium was to bring together scientists from both sides of the Iron Curtain. Naturally, bonds were also deepened between the countries from the East. Today, the objective of the colloquium is still to bring researchers together. They come from the eastern and western member states of the European Union, and, indeed, from all over the world. All who wish to share their ideas on the points where "Computer Science meets Automation" are addressed by this colloquium at the Technische Universität Ilmenau.

All the University's Faculties have joined forces to ensure that nothing is left out. Control engineering, information science, cybernetics, communication technology and systems engineering – for all of these and their applications (ranging from biological systems to heavy engineering), the issues are being covered.

Together with all the organizers I should like to thank you for your contributions to the conference, ensuring, as they do, a most interesting colloquium programme of an interdisciplinary nature.
I am looking forward to an inspiring colloquium. It promises to be a fine platform for you to present your research, to address new concepts and to meet colleagues in Ilmenau.

Professor Peter Scharff
Rector, TU Ilmenau

Professor Christoph Ament
Head of Organisation

Table of Contents

C O N T E N T S

7 New Methods and Technologies for Medicine and Biology

8 Embedded System Design and Application

10 Mobile Communications

11 Education in Computer Science and Automation

52nd Internationales Wissenschaftliches Kolloquium
Technische Universität Ilmenau
10 – 13 September 2007

Nico Pranke, Konrad Froitzheim

The Media Internet Streaming Toolbox

Abstract

MIST[1] - the Media Internet Streaming Toolbox - enables the creation of various live streaming applications in a very flexible and simple way. The MIST-server encodes multiple stream formats using different compression algorithms and sends them through a variety of channels with heterogeneous quality of service (QoS) over the internet. The architecture is designed to scale well with the number of clients. MIST provides the infrastructure to create and run complex flowgraphs that process the media data, encode different streams, and send them over the internet. These objectives are achieved by combining Component Encoding Stream Construction (CESC) [1], dynamically loadable functional components (Compresslets) [2], and automated format negotiation.

1. Introduction

The design of the Media Internet Streaming Toolbox is based on the following approaches, addressing different objectives:

1. Component Encoding Stream Construction (CESC)
2. Dynamically loadable functional components (Compresslets)
3. Automated format negotiation

The main focus of CESC is to support the creation of multiple video stream formats that use conditional replenishment [3] in a scaleable way. The term *Compresslet* was first introduced in [2] and referes to a functional, dynamically loadable component, which implements a specific part of the compression process. The vertices of a *flowgraph* apply the compresslets to process the media data that flows along their edges. The flowgraph is the core of a MIST-based streaming application. It provides the technical description of media processing and stream construction for different given streams. Flowgraphs are created from logical descriptions, so called *usergraphs*, applying automated format negotiation that uses some definitions and principles from [4].

Due to the flexible and scalable graph-based design the toolbox covers a wide

1 http://ente.informatik.tu-freiberg.de/mist/index.php

range of potential applications. Examples include: streaming WebCam servers and corresponding clients; video conferencing/chatting scenarios; fast implementation and evaluation of new compression methods and stream formats.

2. Component Encoding Stream Construction

The CESC-architecture [5] was designed to encode different stream formats and to handle multiple clients simultaneously. Beyond that, it works efficiently and scales well the number of connected clients by exploiting structural similarities in the encoding process.

In order to support the creation of multiple video stream formats in a scaleable way, only those image parts are encoded and transmitted that have changed since the last frame for a particular client has been sent. Since this frame is, due to the heterogeneity of the internet, usually not the same for different clients, a client-dependent extraction of the changed area must be performed by means of a so called generation-map. It stores for each pixel the sequence number of the frame that modified it and thus keeps track of the history of all changes. It allows to extract the changed area for every client independently.

The generation-map is updated with the clock of the source, while the extraction of the area to encode is done with the clock of the particular client. This temporaly decouples the processing of the video and precomputation of components from the actual stream construction. On frame request only the stream construction step has to be done for a particular client using the precomputed components.

3. Compresslets

Compresslets implement specific algorithms of video processing and encoding, for instance the discrete cosine or wavelet transforms and their inverses, colorspace conversion or entropy encoding/decoding. They can have several named *input-pins* to receive their data from upstream and *output-pins* to send their data downstream. Compresslets can have different types, depending on the nature of the service they provide. The different compresslet types are listed in table 1 and follow the classification made in [4]. Compresslets provide partially qualified and not necessarily unique media formats (see section 4) they support for their pins, i.e. they specify only the format parameters that are necessary to perform their service. The information about supported formats is mandatory to perform format negotiation.
To facilitate the cost optimal creation of flowgraphs, every compresslet has to provide a cost factor. Since the true costs in terms of run-time are hard to determine, we use

Type	Description	Input-Pins	Output-Pins
Source	produces data (e.g. a WebCam)	0	1
Sink	consumes data	1	0
Converter	transforms the data from one representation to another but retains the contents	1	1
Filter	manipulates the contents but retains the representation	1	1
Multiplexer	multiplexes 2 or more data streams to one resulting stream	2..*	1
Demultiplexer	demultiplexes an incoming stream into 2 or more resulting streams	1	2..*

Table 1: Compresslet-types.

a qualitative estimation of the true cost.

4. Automated Format Negotiation

We apply a common approach to describe a streaming application by specifying a *media flowgraph* [4, 6], which is a directed, acyclic graph. In our toolbox, it is the technical description of media processing and stream construction for different given streams. The flowgraph is created from so called *usergraphs*, applying an automated format negotiation. Format negotiation and format definition are based on the work done in [4], although we introduced important extensions and modifications to satisfy the constraints of live media streaming applications and CESC.

Usergraphs are directed, acyclic graphs that describe the logic of parts of the streaming application. The elements of the usergraphs are vertices with type source,

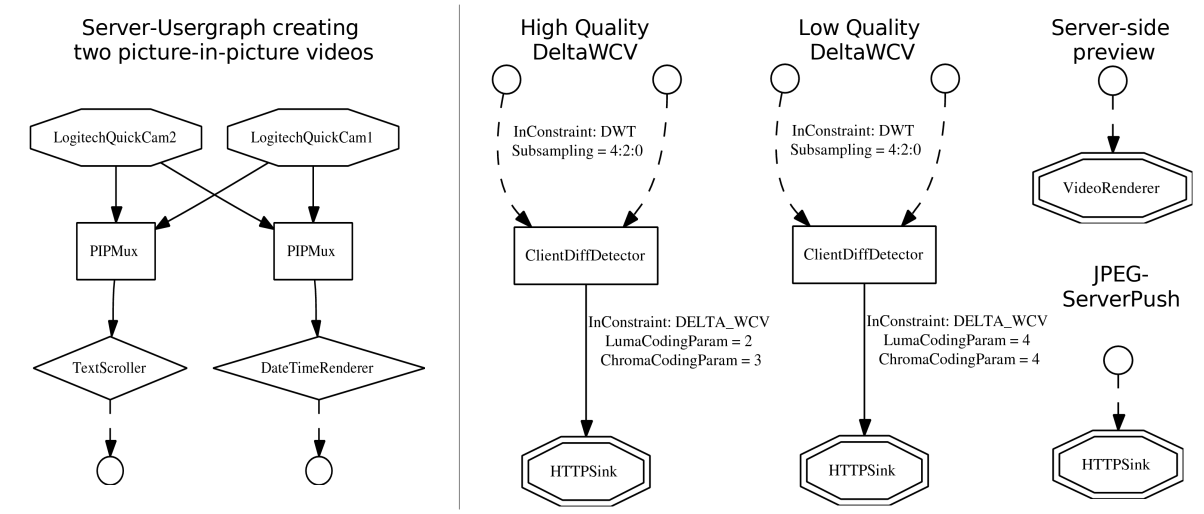

Figure 1: The left part shows a server-usergraph that creates two picture-in-picture videos by combining two sources in butterfly-manner and adding a scrolling text and date/time. The right part shows client-usergraphs that describe the stream construction for particular streams. The desired format and QoS requirements can be specified as a constraint.

sink, filter, multiplexer or demultiplexer. The edge of a usergraph describes a logical connection between an output-pin of its start- and an input-pin of its end-vertex. Their output- and input-formats do not have to be compatible. To technically connect them, format negotiation has to add converter-vertices. There are two possible types of usergraphs. *Server-usergraphs* express how the media preprocessing step has to be done. They contain at least one source- and no sink-vertex. An example of a server-usergraph is shown in figure 1. *Client-usergraphs* describe the client-specific processing of the data (usually the stream-construction). They contain no source- and at least one sink-vertex. Some examples are given in the right part of figure 1. To create a complete, valid flowgraph one server-usergraph and at least one client-usergraph are combined. In doing so, unconnected input-pins of the client-usergraphs are connected with the unconnected output-pins of the server-usergraph.

To perform automated format negotiation it is necessary to use a sophisticated format definition. According to [4], formats are defined by their *classification* and *specification*. The classification divides all possible formats into main categories and consists of *type* and *subtype*[2]. The specification adds parameters[3] that contain a set of values. A format is *unique*, if all its parameters have only one value. The unique formats created from a non-unique format can be obtained by enumerating all combinations of parameters with one value.

If a parameter accepts any value, a wildcard-attribute can be used. Furthermore each value can be attached with an optional quality, which comes into play when more than one format are a valid solution. The quality of a unique format is defined as the mean of the qualities of its parameter values. Contrary to the approach in [4] our primary goal is to find a low cost solution due to the real-time constraints of live-media streaming applications. Only in the case of solutions with equal cost, the quality values are considered to choose the best solution.

To control the negotiation process an additional *constraint format* can be specified for the pins of usergraph-vertices. A format is accepted for the pin only if it matches the given constraint. This enforces the creation of a specific format at the pins of the vertex. To model explicit dependencies between formats of input- and output-pins of a compresslet the concept of an *io-partner* is introduced in [4]: each input format can be associated with an output-format and vice versa. If the format of an input-pin has an io-partner it is accepted only if the io-partner matches the constraint format of its output-pin. We extend this concept to model dependencies

2 e.g. type=VIDEO and subtype=DWT, DCT or RGB32
3 e.g. Resolution=320x240, 640x40; Subsampling=4:2:0, 4:2:2, 4:4:4

between different output-pins of demultiplexers (*output-partner*) and different input-pins of multiplexers (*input-partner*).

The underlying mathematical problem of finding a minimum-weight connected subgraph of a subset of vertices of a graph, the so-called *Steiner tree,* is known to be NP-complete and is even hard to approximate [7]. The number of vertices in the graph depends on the number of available converters and the number of values they provide for their parameters and can add up to several thousands. Since we aim to do format negotiation in less than 100 milliseconds, we do not solve the problem exactly, but rather use a heuristic approach. Our heuristic performs the format negotiation for each edge of the usergraphs and for the edges joining client- and server-usergraphs separately. The negotiation process connects adjacent usergraph-vertices by inserting converter-vertices to convert the format of the output-pin of the start-vertex to the format of the input-pin of the end-vertex of the edge. To decide, whether the pins of two vertices can be connected, their formats have to be compared. They *match* if type and subtype are equal and the value sets of all parameters that exist in both formats have a non-empty intersection. Parameters that exist in only one format are added to the resulting *intersection format*.

To actually perform the format negotiation for a usergraph-edge a *negotiation-graph* is defined, which initially contains only the start- and end-vertices of the edge in question. The Dijkstra-algorithm is then applied to find the shortest path between them. In contrast to the negotiation-graph used in [4] our graph is not constructed completely at the beginning, but is rather extended dynamically during the negotiation process. While processing a vertex *v* with the Dijkstra-algorithm the negotiation-graph is updated in the following way: For all available converter-compresslets the pairs of unique input- and output-formats are enumerated and a new converter-vertex *w* is created for the pair, if the input-format matches the output-format of *v.* The vertex *w* and a new edge *e* connecting *v* and *w* are added to the negotiation-graph, if the resulting output-format of *w* is not already produced by another vertex that has been processed by the algorithm before. The edge *e* is annotated with the intersection format resulting from the output format of *v* and the input format of *w*. Although the complexity of the Dijkstra-algorithm is $O(n \log n + m)$, its run-time can be considerable, since the complete negotiation-graph contains a large number of converter-vertices with a high degree that produce a non-disjoint set of formats. The successive extension of the graph leads to paths that create new formats only, keeps the negotiation-graph small and thus greatly reduces the run-time necessary to create the flowgraph.

Since all output formats of converters on the path originating from the start-vertex of a usergraph-edge represent the same media data just with different formats, these converter-vertices can be used as a potential start-vertex for further usergraph-edges originating from the same vertex. Our heuristic computes the shortest path for a usergraph-edge from each possible start-vertex to the end-vertex of the edge. The start-vertex that results in the shortest path is finally chosen.

The negotiation process is done for each usergraph-edge independently. This leads to the problem of combining all the local solutions to a global one. Two approaches are suggested in [4]. One is to compute all possible solutions for every usergraph-edge and to examine all possible combinations of them to find a valid global solution. This approach is feasible only if the number of solutions per edge is relatively small and the number of valid combinations is small as well. But this can not always be guaranteed. The other suggested approach is to find a "narrowest" edge through which all media data in the graph flows. The parameter values are then propagated up and downstream from this location. But the requirement for the existence of such an edge limits the possible flowgraph layout largely.

Our negotiation process is directed from the source-vertices to the sink-vertices of the usergraphs and is done in breadth first manner. If a solution is found for a usergraph-edge, the resulting format at the input-pin of the end-vertex determines the io-partner format and thus the output-format of the end-vertex. Filter-vertices retain their input-format. So, if the end-vertex is a filter, the io-partner can be manipulated directly by computing the intersection with the input-format to reflect the parameter values, fixed by the input-format. For demultiplexer- and multiplexer-vertices the io-partner format is used as format for the output-pins. The breadth first approach guarantees that all usergraph-edges that influence the output-formats of the start-vertex of an edge have been negotiated before. Parameter values are propageted from the sources to the sinks taking into account the changes they experince by intermediate converters. Thus no unresolvable conflicts can occur and there are no constraints on the layout of the flowgraph.

5. Execution of the flowgraph

One of the most important approaches of CESC is the decoupling of video processing from stream construction. To reflect this, the two-part structure induced by the client-/server-usergraphs is retained for the execution of the flowgraph. The first part of the flowgraph, corresponding to the server-usergraph is executed with the clock of the sources. The source vertices trigger the execution if they can deliver new data. There are two possibillities for the execution of the flowgraph-parts corresponding to the client-usergraphs:

1. they are executed automatically with the clock of the source
2. they are executed asynchronously for a specific client; triggering has to be done by the sink in the corresponding client-usergraph in this case

The first is appropriate for streams without client dependent states sending all the data to every connected client. The latter is appropriate to support video streams that have client dependent states, e.g. streams that apply conditional replenishment. Then a special multiplexer-compresslet, called *ClientDiffDetector* can be used to determine the changed area for a particular client by means of the generation-map. The information about the area to encode is passed downstream the flowgraph along with the media data. Furthermore manually triggered usergraphs can set a maximum frame rate to support different quality of service requirements. The triggering process allows to group clients according to the sequence number of their preceeding frame. The flowgraph part that corresponds to a specific stream and thus the stream construction is executed only once for all clients in the group in this case. It increases scaleability of state dependent streams largely. Detailed measurements of scalability are subject for further research.

6. Example

As an example we connected the client-usergraphs from figure 1 with each of the two unconnected pins of the server-usergraph from figure 1. The generated flowgraph contains additional converter-vertices to do tasks like scaling, colorspace-conversions, discrete cosine and wavelet transforms and entropy encoding. The resulting streaming server creates for each of the two videos produced by the server-usergraph a JPEG-ServerPush stream and two DeltaWCV [8] streams in different qualities. DeltaWCV is a wavelet based video stream that encodes and transmits only the changed blocks with respect to the last frame sent and is far superior to JPEG-ServerPush regarding rate-distortion. Since the DeltaWCV-streams depend on the state of the client, their construction is decoupled by means of the generation-map contained in the ClientDiffDetector-compresslet as introduced in section 2. For JPEG-ServerPush complete frames are always encoded and thus no decoupling is necessary. Both streams are constructed asynchronously on frame request. Furthermore a synchronous client-usergraph containing a VideoRenderersink is added to offer a server-side preview of the resulting videos. This example is currently used to stream videos of our internet-controllable model railroad[4].

7. Results and Further Research Topics

We combined and modified CESC, the usage of compresslets, and automated format negotiation to flexibly create live streaming applications that are highly scalable with respect to the number of connected clients. We applied our implementation to create video streams with different formats and qualities of service. Flexibility is addressed by the usage of usergraphs that are the building blocks of the flowgraph. Scalability is addressed by the usage of a generation-map, the classification of clients with respect to their state and the asynchronous execution of flowgraph parts that relate to the construction of specific streams.

The toolbox greatly benefits from the existence of a large number of compresslets. An emphasis of our further work is thus the implementation of compresslets not only for video but also for audio to support a wider range of applications. Moreover we plan to introduce dynamic vertex-parameters to control the behaviour of the application at run-time. Further work is required to realize particular applications, e.g. we plan a GUI-application to create, manage, and combine usergraphs interactively.

References:

[1] **Kaufman, L.**, *Component Encoding Stream Construction*, Diploma thesis, University of Ulm, 1999.

[2] **Bönisch, H., Froitzheim K**., *Server Side 'Compresslets' for Internet Multimedia Streams*, IEEE Multimedia Systems, ICMCS, Vol. 2 1999: 82-86.

[3] **Mounts, F. W.,** *A video encoding system with conditional picture-element replenishment*, Bell Systems Technical Journal, vol. 48, no. 7, pp. 2545--2554, Sept. 1969.

[4] **Marco Lohse, Philipp Slusallek, and Patrick Wambach,** *Extended Format Definition and Quality-driven Format Negotiation in Multimedia Systems,* Multimedia 2001 - Proceedings of the EUROGRAPHICS Workshop, Manchester, United Kingdom, September 8th-9th, 2001

[5] **Bönisch, H., Froitzheim K.**, *Flexible Server Architecture for Optimal Presentation of Internet Multimedia Stream at the Client,* Proc. SPIE Vol. 3969, p. 232-239, Multimedia Computing and Networking 2000, Klara Nahrstedt; Wu-chi Feng; Eds.: Januar 2000.

[6] **Microsoft DirectShow Architecture,** http://msdn.microsoft.com, 2000

[7] **Chlebik, M. and J.Chlebikova, J.**, *Approximation Hardness of the Steiner Tree Problem on Graphs. Proc. 8th Scandinavian Workshop on Algorithm Theory (SWAT).* Springer-Verlag, pp. 170-179, 2002.

[8] **Pranke, N.**, *Ein wavelet-basierter Video-Codec – Algorithmen und Implementierung,* Diploma thesis, Technical University BergakademieFreiberg, 2002

Nico Pranke, Konrad Froitzheim

Technische Universität Bergakademie Freiberg,

Institut für Informatik, Bernhard-von-Cotta-Strasse 2, 09596 Freiberg

Phone: +49 (0) 3731 / 39 – 3016

Fax: +49 (0) 3731 / 39 – 22 98

E-mail: pranke@informatik.tu-freiberg.de, frz@informatik.tu-freiberg.de

---

4 http://rr.informatik.tu-freiberg.de