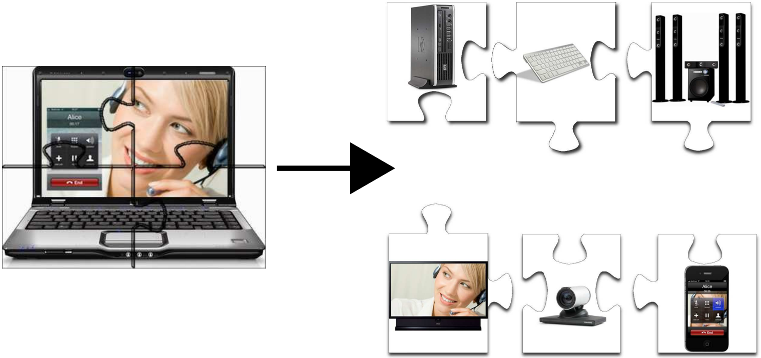
DAMPAT: Dynamic Adaptation of Multimedia Presentations for Application Mobility

Francisco Velázquez, Frank Eliassen

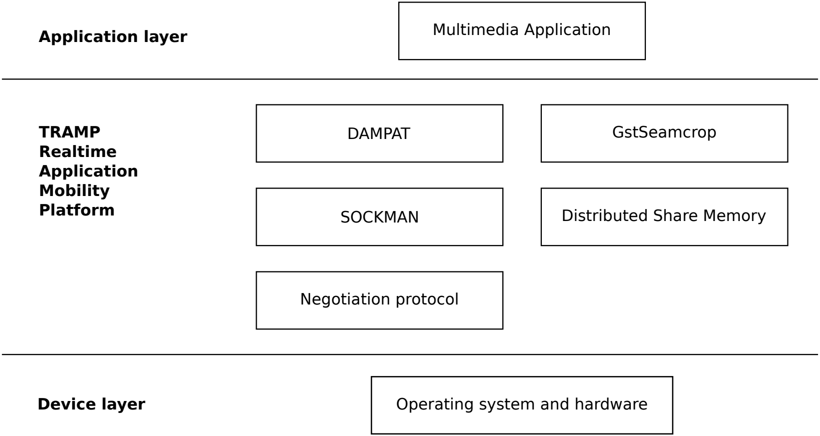
October 21, 2017

Application mobility

Paradigm where users can move parts of their running applications across multiple heterogeneous devices in a seamless manner.



Challenges



Application layer

Multimedia Application

TRAMP
Realtime
Application
Mobility
Platform

DAMPAT

GstSeamcrop

SOCKMAN

Distributed Share Memory

Negotiation protocol

Device layer

Operating system and hardware

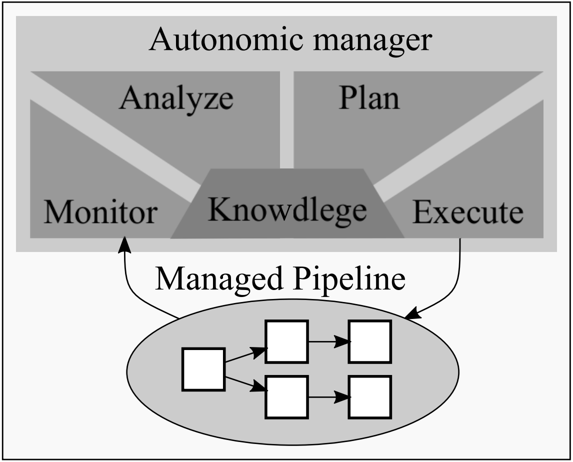
Challenges of adaptive multimedia presentations

- Adaptive systems adapt a subset of scenarios in application mobility
  - Fidelity adaptation
  - Modality adaptation
  - Modality selection
  - Content adaptation
  - Retargeting

DAMPAT: Dynamic Adaptation of Multimedia Presentations in Application Mobility

- Context-aware runtime adaptive system
  - Adapts multimedia pipelines
- Adopts Dynamic Software Product Lines (DSPL)
  - Possible configurations are seen as variability management problem
  - Utility functions to find *best* variant

Monitoring, Analysis, Planning, and Execution (MAPE) adaptation loop



Utility functions

- Best is the variant that produces the highest utility according to the current contextual situation
  - Each component (GStreamer element) provides a utility
- If we want to get the best variant, we have to compare all possible pipeline configurations

Challenges of autonomously creating all possible pipeline configurations

- Control of combinatorial growth due to compositional and parameterization variability
  - Pipeline components
  - Components properties
- Control *valid* path combinations
- Selection of best variant

$$\Upsilon(u,g) = \sum_{j=1}^{l} ut(u.p_j, g.p_j) \cdot u.p_j.we \tag{1}$$

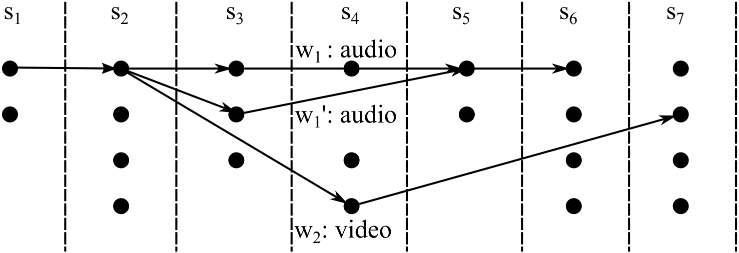
Architectural constraints

- Allows developers to introduce architectural design knowledge
- Enforces directed graphs, and they avoid unnecessary checks of connectors compatibility

Levels of functional stages

pre-processing						retargeter				post processing					
source handler		Input Format Handler			color space converter	adaptation type			Filters		Output Format Handler Format Handler		Sink Handler		
protocol handler	source handler	parser	demuxer	decoder	video converter	modality adaptation	content adaptation	fidelity adaptation	stream selector	mixer	encoder	muxer	payload encoder muxer	session manager	sink handler

Valid path combinations



Binary Reflected Gray Code (BRGC)

Bit strings Subsets	000 $\{0\}$	001 $\{w_2\}$	011 $\{w_1', w_2\}$	010 $\{w_1'\}$	110 $\{w_1, w_1'\}$	111 $\{w_1, w_1', w_2\}$	101 $\{w_1, w_2\}$	100 $\{w_1\}$
Modality counter	$m_{audio} = 0$ $m_{video} = 0$	$m_{audio} = 0$ $m_{video} = 1$	$m_{audio} = 1$ $m_{video} = 1$	$m_{audio} = 1$ $m_{video} = 0$	$m_{audio} = 2$ $m_{video} = 0$	$m_{audio} = 2$ $m_{video} = 1$	$m_{audio} = 1$ $m_{video} = 1$	$m_{audio} = 1$ $m_{video} = 0$
Subgraph $\in G'$	Not valid	g_1	g_2	g_3	Not valid	Not valid	g_4	g_5

Scalability when linking GStreamer pipeline elements

- Unpredictability of time needed for capability negotiation
- No registry to easily know which elements need hardware instantiation
- `query-caps` and `accept-caps`
  - Recursion and no proper implementation of `accept-caps` handler (due to `CAPS` event)

Query measurements

- `capsnego.c`
  - `audiotestsrc`, `adder`, `volume`, `audioconvert`, `identity`
  - `videotestsrc`, `videomixer`, `videoscale`, `videoconvert`, `identity`
- `GST_TRACERS`
- `gsttracer-negotiation-analyzer.py`

Table: Queries when building similar pipelines

Number of comp.	Modality	Total queries	Repeated queries	Response time (ms)
5	audio	16	1	2.3
455	audio	104 041	70 252	28 953.05
5	video	21	3	20.71
455	video	2 721	453	1 782.815

Queries with/without playbin3

- audio: Ogg/Vorbis
- video: WebM/Vorbis/VP8

Input	Elem.	Total queries	Repeated queries	Queries response time (ms)
playbin audio	17	111	28	50
audio	17	107	8	43
playbin video	27	208	112	250
video	27	207	6	161

Work in progress

- Find out how to estimate in a more predictable manner the time needed for building pipelines