Component-wise Mapping of Media-needs to a Distributed Presentation Environment

Loe Feijs, Jun Hu
Department of Industrial Design
Eindhoven University of Technology
Den Dolech 2, 5600MB Eindhoven, The Netherlands
{l.m.g.feijs, j.hu}@tue.nl

Abstract

Whereas formal specification and verification have shown value by improving reliability and trustworthiness of traditional industrial systems, we made a contribution by applying them to the field of distributed multimedia presentations in an Ambient Intelligence context. We investigate a mapping problem in which media needs are to be satisfied using given presentation resources. The goal of the investigation is to see whether Broy's stream-based component framework can be used to model media-related interfaces and constraints in an elegant way. The formalization will serve as a framework for the development of an automated mapper that can handle real media needs and real presentation resources. It combines the well-known notations of Z with an underlying concurrency theory. We show that not only verification issues can be handled such as bandwidth and delay constraints, but also architecture-level issues such as network structural media-type compatibilities.

1 Introduction

In the vision of Ambient Intelligence, technology becomes invisible, embedded in our natural surroundings, present whenever we need it, enabled by simple and effortless interactions, attuned to all our senses, adaptive to users and context and autonomously acting [1]. Interactive media presentations will no longer be folded in a flat screen with 5.1 speakers, but instead, will be distributed into the user's environment. The environment with distributed and networked interfaces, functioning as interactive theater, will engage people in more immersive experiences. In a previous study done in Philips Electronics, we have shown how to structure the content and system to present interactive media to a distributed environment that consists of not only screens and speakers, but also lights and robots [6, 7]. It was done in the context of EU funded R&D projects (NexTV [11] and ICE-CREAM [9]). (Figure 1 shows two users interacting with the ICE-CREAM demonstration, exhibited in IBC 2003 [8]).

Figure 1. ICE-CREAM theater at IBC 2003

One of the problems is the variety of such environments - thinking of how different people arrange their living rooms. This has been a long-existing problem with regard to web browsers. In order to give the same look-and-feel on a single web page to the users, the poor authors often have to work very hard to use all kinds of tricks, and when this fails, write different versions for different browsers. On the other hand, web authors are lucky - if they only consider Internet Explorer, they will possibly cover 4/5 of the audience, and if they are kind enough to consider Mozilla users, they will almost cover them all. We are not that lucky. We can not assume that there are only Explorer-like living rooms and Mozilla-like living rooms. They are different, in terms of available components and connections. It is impossible to have one version to fit them all. Instead we should have only one abstract media description for all living rooms, then map the media needs to every physical room, to create the experience for end users as intended as possible. Similar mapping problems are studied informally in [10].

We aim at the formalization of such mapping problems. Each mapping problem is defined by a set of media needs and a set of presentation resources. The media needs are abstract description of the media content as well as its requirements for the physical components and their connections, including for example audiovisual players, robotic interactors and lights. These physical components are presentation resources. We employ Broy's stream-based component framework [2]. The following issues must be addressed:

1. component interfaces for control and for events;
2. user interfaces: touch screens, switches and buttons;
3. throughput requirements and network delays;
4. embedding of presentation components into devices;
5. standardization of specifications (keeping the math away from the media developers).

The formalization will serve as a helpful framework for the development of an automated mapper that can handle real media needs and real presentation resources.

The mapping is to find the presentation resources for a set of media so that each media content is properly presented. In a typical implementation a mapping is a process that at run-time deals with control and events, and that has access to a number of reserved presentation resources. A presentation could be built by a "mapping server", which behaves as a *factory* [3], launching presentations. Of course the mapping server takes a mapping problem (= media needs + presentation resources) as its input, solves the mapping problem, makes the reservations, and then builds and launches the presentation. In fact, a presentation could be structured as a main process by the mapper server, upwards serving the media needs, and several distributed and auxiliary processes downwards employing the local resources from the devices in the environment. Typically, a distributed presentation is built by a local server (also a factory). Although one may expect that one such server per device would be sufficient, this is actually not the case; the reason is that several mutually incompatible software abstractions exist upon the the same device (for example, the Real player and the Microsoft media player on a Windows PC).

2 Preliminaries

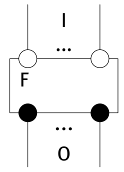
To get started we consider a few simple media needs:

- high-resolution video presentation
- low-resolution video presentation
- dancing behavior
- up-down interactor

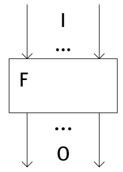
which have to be obtained from the following resources:

- *cable*: upc cable 801(10Mb/s)
- *the_sign*: www.clips.com/the_sign.mpg (10Kb/s)
- *happy_puppy*: www.dance.com/happy_puppy.dhb (1Kb/s)
- *updown*: /interactors/updown.exe (1b/s).

To model the media needs and presentation resources, we follow the approach of Broy [2] based on timed streams. We have some drawing conventions of our own, which is derived from ADL Darwin [4, 5]. We use



instead of



(See [2], p.7).

In an interactive presentation system, it is necessary to distinguish user interface channels from data streaming channels. For this we add another convention, namely output channels ■ modeling real, physical presentations such as sounds, video frames and robotic movements, and input channels □ modeling the components with which the user may interact with the system, such as a GUI interface on a screen and physical buttons on a remote control.

Broy demands that a set S of types be given. We put

$$S = \{PAL, MPG, DBH, IBH, \mathbb{R}\}$$

for PAL streams, MPEG streams, Dancing BeHavior, Interaction BeHavior and "things in the $\mathbb{R}$eal world"(for example, buttons, screens and robots).

In Broy's approach there is a discrete time frame representing time as an infinite chain of time intervals of equal finite duration. We take 1 second for that duration. This allows us to formally represent the facts for example that there are 25 video frames per second in a high-resolution video stream and that each frame takes a number of bits.

We assume functions on data:

$$\begin{array}{rcll} bits & : & PAL \to \{0,1\}^* & \text{... also for } MPG, DBH \\ pr & : & PAL \to \mathbb{R} & (pr \text{ for presentation}) \\ pr & : & MGP \to \mathbb{R} & (pr \text{ for presentation}) \\ pr & : & DBH \to \mathbb{R} & (pr \text{ for presentation}) \\ pr & : & IBH \to \mathbb{R} & (pr \text{ for presentation}). \end{array}$$

and obtain lifted versions of pr at timed-stream level:

$$\begin{array}{rcll} pr' & : & PAL^* \to \mathbb{R}^* & \text{by } (pr'(p)).i = pr(p.i) \\ pr'' & : & (PAL^*)^\infty \to (\mathbb{R}^*)^\infty & \text{by } (pr''(x)).t = pr'(x.t) \end{array}$$

and so on. The formal model has some structure of its own (Figure 2).

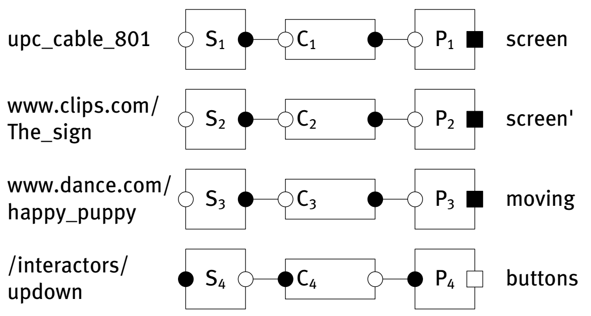


Figure 2. Formal model structure

In order to map this we must assume devices and connections. Figure 3 shows a possible configuration with the following devices:

- *STB*: set-top box;
- *modem*: internet modem;
- *hi_res*&*PIP*: high-resolution monitor with picture-in-picture function;
- *lo_res*: low-resolution (lo_res) monitor;
- *level*2*_robot*: "level-2" robot;
- *RC*: remote control for the low-resolution monitor

and the following connections:

- $STB - modem$ 100b/s duplex;
- $STB - hi\_res$ 100Mb/s duplex;
- $modem - lo\_res$ 1Mb/s duplex;
- $modem - robot$ 10Kb/s duplex;
- $RC - lo\_res$ 100b/s duplex.

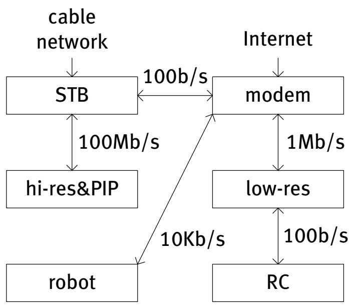


Figure 3. A possible configuration

There are two possible ways of presenting the dancing behavior: either by an animated character on the *PIP* of the *hi_res* screen, or by the physical robot together with the *RC* providing user input. But the $STB - modem$ bandwidth is not enough for the *PIP* to do it. So the robot must dance and the *RC* must provide the up-down interaction.

3 Specification

We need some abbreviations. For timed stream z and $n_1, n_2 \in \mathbb{N}$,

$$rate(z, n_1, n_2) == \forall t : \mathbb{N}\backslash\{0\} \bullet \#(z.t) = n_1 \\ \wedge \forall i : \mathbb{N} \bullet 0 < i \leqslant \#(z.t) \Rightarrow \#bits(z.t.i) = n_2$$

with the intuition that e.g. $rate(z, 100, 100\text{K})$ means that 100 frames fit into once second and that 100K bits go into each frame (Figure 4).

For timed streams x and x', and $\Delta \in \mathbb{N}$,

$$delay(x, x', \Delta) == \forall t : \mathbb{N}\backslash\{0\} \bullet x.t = x'.(t + \Delta)$$

and for $m \in \mathbb{N}$,

$$maxdelay(x, x', m) == \\ \exists \Delta : \mathbb{N} \bullet \Delta < m \wedge delay(x, x', \Delta)$$

The rate information is put into the source devices S_1, S_2, S_3 and the sink S_4.

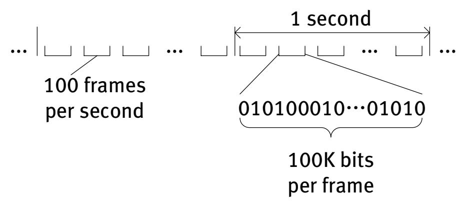


Figure 4. $rate(z, 100, 100K)$

S_1
in $cable : PAL$
out $z : PAL$
$z = cable$
$rate(z, 100, 100\text{K})$

S_2
in $the\_sign : MPG$
out $z : MPG$
$z = the\_sign$
$rate(z, 2, 5K)$

S_3
in $happy\_puppy : DBH$
out $z : DBH$
$z = happy\_puppy$
$rate(z, 10, 100)$

S_4
in $z : IBH$
out $updown : IBH$
$z = updown$
$rate(z, 1, 1)$

The relation between media formats and the real world is described in $P_1, ..., P_4$.

P_1
in $w : PAL$
out $screen : \mathbb{R}$
$screen = pr''(w)$

P_2
in $w : MPG$
out $screen' : \mathbb{R}$
$screen' = pr''(w)$

P_3
in $w : DBH$
out $moving : \mathbb{R}$
$moving = pr''(w)$

P_4
in $button : \mathbb{R}$
out $w : IBH$
$button = pr''(w)$

Matters of delay are modelled in the channels.

C_1
in $z : PAL$
out $w : PAL$
$maxdelay(z, w, 2)$

C_2
in $z : MPG$
out $w : MPG$
$maxdelay(z, w, 3)$

C_3
in $z : DBH$
out $w : DBH$
$maxdelay(z, w, 4)$

C_4
in $z : IBH$
out $w : IBH$
$maxdelay(z, w, 4)$

Finally the whole system is specified by:

$$\begin{aligned} SYSTEM = {} & (S_1 \circ C_1 \circ P_1) \setminus \{z, w\} \\ \| {} & (S_2 \circ C_2 \circ P_2) \setminus \{z, w\} \\ \| {} & (S_3 \circ C_3 \circ P_3) \setminus \{z, w\} \\ \| {} & (P_4 \circ C_4 \circ S_4) \setminus \{z, w\} \end{aligned}$$

which has a syntactic interface of (I, O), where

$$I = \{cable, the\_sign, happy\_puppy, button\}$$
$$O = \{screen, screen', moving, updown\}$$

4 Presentation resources

The next step is to formalize the presentation resources, i.e. the components and the network connections. We assume that the components cause delays and that the connections give rise to bandwidth restrictions. The structure of the resulting model is shown in Figure 5. Clearly we will need renaming later, for example, $[the\_sign/url_1]$ (for modem) and $[cable/channel]$ (for STB).

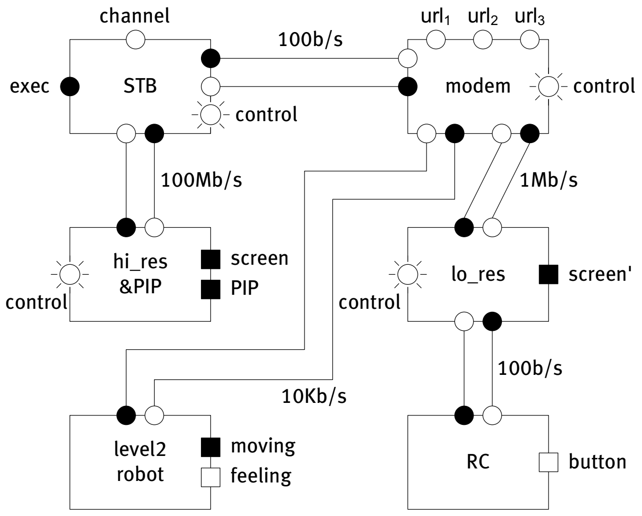


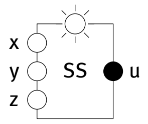
Figure 5. Structure of the resource model

Somehow the components must be configured to perform the required routing of streams. Although the current mapping-example does not require it, in general a component must be able to combine two incoming streams and put them on a single output port, or conversely, split what comes in to produce separate outgoing streams. In other words, some components must contain a switch.

A component containing a switch must have an extra channel of a special type, a control channel, accepting so-called *commands*, from a set $\mathbb{C}$. So from now on,

$S = \{PAL, MPG, DBH, IBH, \mathbb{R}, \mathbb{C}\}$

To get some experience in modelling switches, we try a simplified component, the Simple Switch *SS*:



$(x, y, z : DBH)$

Any selection from x, y and z can be combined and offered on u, provided the rates fit. To adapt to the rate of u, which is assumed to be fixed, dummy data must be stuffed.

We assume the existence of disjoint copies of the set DBH, which we denote DBH_1, DBH_2, etc. We also assume conversion functions $d_1 : DBH \rightarrow DBH_1$, $d_2 : DBH \rightarrow DBH_2$, etc. and the corresponding inverses, e.g. $d_1^{-1} : DBH_1 \rightarrow DBH$. They give rise to lifted versions, e.g. $d_1' : DBH^* \rightarrow DBH_1^*$. These assumptions allow us to specify streams being merged, e.g. x and y merged into u:

$u©DBH_1 = d_1''(x) \wedge u©DBH_2 = d_2''(y)$

In other words, the mechanism of disjoint copies is used to model tagging of sub-streams conveniently in an abstract way. (In Broy's theory, $S©x$ means the stream obtained from x by deleting all its messages that are not elements of the set S.)

We introduce abbreviations:

$$maxrate(x, n) == \forall t : \mathbb{N}\backslash\{0\} \bullet \sum_{i=1}^{\#(x.t)} \#bits(x.t.i) < n$$

$$maxrate(x, y, n) == \forall t : \mathbb{N}\backslash\{0\} \bullet \sum_{i=1}^{\#(x.t)} \#bits(x.t.i) + \sum_{i=1}^{\#(y.t)} \#bits(y.t.i) < n$$

Although a real network connection carries bits, we use a more abstract model to reflect the general-purpose nature of connections:

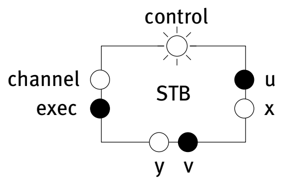
$$\begin{aligned} \mathbb{B} = & \bigcup_{i=0,1,2\ldots} DBH_i \\ \cup & \bigcup_{i=0,1,2\ldots} PAL_i \\ \cup & \text{ etc.} \qquad \text{(all data types used).} \end{aligned}$$

where $DBH_0 \triangleq DBH$ (So this can be used for the simple case where tagging is not really needed).

```
SS
in   $x, y, z : DBH$
in   $s : \mathbb{C}$
out  $u : \mathbb{B}$
----------------------------------------
$s.1.1 = xy2u \Rightarrow$
    $maxrate(x, y, 100)$
    $\wedge\ delay(d_1'(x), u©DBH_1, \Delta_{SS})$
    $\wedge\ delay(d_2'(x), u©DBH_2, \Delta_{SS})$
$s.1.1 = yx2u \Rightarrow$
    $maxrate(x, y, 100)$
    $\wedge\ delay(d_2'(x), u©DBH_2, \Delta_{SS})$
    $\wedge\ delay(d_1'(x), u©DBH_1, \Delta_{SS})$
$s.1.1 = x2u \Rightarrow$
    $maxrate(x, 100)$
    $\wedge\ delay(d_1'(x), u©DBH_1, \Delta_{SS})$
etc.  $(y2u, z2u, xz2u, \ldots)$
```

On the basis of this example, we consider the real components, assuming the real delay occurs in the components as show in the following table. This means that a constraint such as $delay(x, u, \Delta_{SS})$ has to be rewritten as $delay(d_1''(x), u©DBH_1, \Delta_{SS})$. We assume that the delay is fixed, independent of the load and the precise routing. Otherwise, more sophisticated schemes can be devised if necessary. Note that $©'$ is the lifted version of ©. The rate constraints are modelled as if they belong to the input ports.

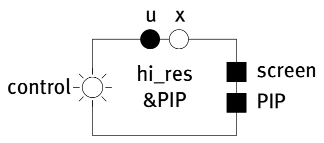
component	delay	value
STB	Δ_{STB}	1
modem	Δ_{modem}	1
hi_res	Δ_{hi_res}	1
lo_res	Δ_{lo_res}	1
level2_robot	Δ_{level2_robot}	0
RC	Δ_{RC}	0



STB

in $channel : PAL$
in $control : \mathbb{C}$
in $x, y : \mathbb{B}$
out $exec : IBH$
out $u, v : \mathbb{B}$

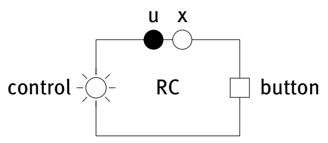
$control.1.1 = channel2v \Rightarrow$
$\quad maxrate(channel, 100M)$
$\quad \wedge\ delay(channel, v©PAL, \Delta_{STB})$
$control.1.1 = x2exec \Rightarrow$
$\quad maxrate(x, 100K)$
$\quad \wedge\ delay(x©DBH, exec, \Delta_{STB})$
$control.1.1 = channel2v\_x2exec \Rightarrow$
$\quad maxrate(channel, 100M)$
$\quad \wedge\ maxrate(x, 100K)$
$\quad \wedge\ delay(channel, v©PAL, \Delta_{STB})$
$\quad \wedge\ delay(x©DBH, exec, \Delta_{STB})$
etc.



hi_res&PIP

in $x : \mathbb{B}$
in $control : \mathbb{C}$
out $u : \mathbb{B}$
out $screen, PIP : \mathbb{R}$

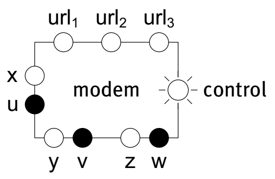
$control.1.1 = x2screen \Rightarrow$
$\quad maxrate(x, 100M)$
$\quad \wedge\ delay(pr''(x©PAL), screen, \Delta_{hi\_res\&PIP})$
$control.1.1 = x2screen\_PIP \Rightarrow$
$\quad maxrate(x, 100M)$
$\quad \wedge\ delay(pr''(x©PAL), screen, \Delta_{hi\_res\&PIP})$
$\quad \wedge\ delay(pr''(x©DBH), PIP, \Delta_{hi\_res\&PIP})$
etc.



RC

in $x : \mathbb{B}$
in $control : \mathbb{C}$
in $button : \mathbb{R}$
out $u : \mathbb{B}$

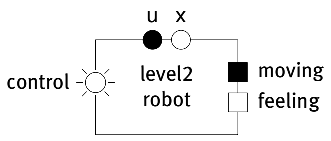
$control.1.1 = button2u \Rightarrow$
$\quad maxrate(u, 100)$
$\quad \wedge\ delay(button, pr''(u©IBH), \Delta_{RC})$



modem

in $url_1, url_2, url_3, x, y, z : \mathbb{B}$
in $control : \mathbb{C}$
out $u, v, w : \mathbb{B}$

$control.1.1 = url_1 2v \Rightarrow$
$\quad maxrate(url_1, 56K)$
$\quad \wedge\ delay(url_1, v, \Delta_{modem})$
$control.1.1 = z2u \Rightarrow$
$\quad maxrate(z, 1M)$
$\quad \wedge\ delay(z, u, \Delta_{modem})$
$control.1.1 = url_2 2w \Rightarrow$
$\quad maxrate(url_2, 56K)$
$\quad \wedge\ delay(url_2, w, \Delta_{modem})$
etc.

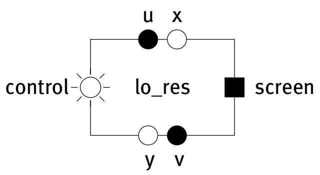


level2_robot

in $x : \mathbb{B}$
in $feeling : \mathbb{R}$
in $control : \mathbb{C}$
out $moving : \mathbb{R}$
out $u : \mathbb{B}$

$control.1.1 = x2moving \Rightarrow$
$\quad maxrate(x, 10K)$
$\quad \wedge\ delay(pr''(x©DBH), moving, \Delta_{level2\_robot})$
$control.1.1 = feeling2u \Rightarrow$
$\quad maxrate(u, 10K)$
$\quad \wedge\ delay(feeling, pr''(u©IBH), \Delta_{level2\_robot})$

$$
\begin{array}{l}
control.1.1 = x2moving\_feeling2u \Rightarrow \\
\quad maxrate(x, 10K) \\
\quad \wedge\ maxrate(u, 10K) \\
\quad \wedge\ delay(pr''(x \copyright DBH), moving, \Delta_{level2\_robot}) \\
\quad \wedge\ delay(feeling, pr''(u \copyright IBH), \Delta_{level2\_robot})
\end{array}
$$



$lo\_res\&PIP$

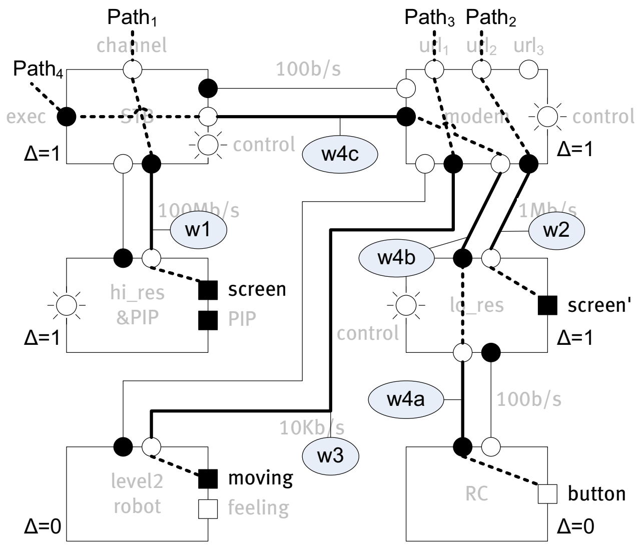
$$
\begin{array}{ll}
\textbf{in} & x, y : \mathbb{B} \\
\textbf{in} & control : \mathbb{C} \\
\textbf{out} & u, v : \mathbb{B} \\
\textbf{out} & screen : \mathbb{R}
\end{array}
$$

$$
\begin{array}{l}
control.1.1 = x2screen \Rightarrow \\
\quad maxrate(x, 1M) \\
\quad \wedge\ delay(pr''(x \copyright MPG), screen, \Delta_{lo\_res}) \\
control.1.1 = y2u \Rightarrow \\
\quad maxrate(y, 100) \\
\quad \wedge\ delay(y, u, \Delta_{lo\_res}) \\
control.1.1 = x2screen\_y2u \Rightarrow \\
\quad maxrate(x, 1M) \\
\quad \wedge\ maxrate(y, 100) \\
\quad \wedge\ delay(pr''(x \copyright MPG), screen, \Delta_{lo\_res}) \\
\quad \wedge\ delay(y, u, \Delta_{lo\_res}) \\
\text{etc.}
\end{array}
$$

5 Mapping

Next we propose a mapping. Abstractly, a mapping is a set of paths through a given network, one path for each of the four chains specified in *SYSTEM*. At a concrete level, the paths are established by feeding appropriate switching commands into *STB*, *modem* and *lo_res*. In general there is a freedom in choosing these paths, for example if alternative routing through the network exists. Even the choice of which screen or interactive input is to be used is not a priori fixed. The four paths are shown in Figure 6. In an informal way it is easy to check that the delay and bandwidth constraints are not violated and the mapping is feasible.

To formalize the connections we use renaming by adapting port names to "wire names". The wire names are also shown in Figure 6. Note that *w4a*, *w4b* and *w4c* together form a path from *RC* via *lo_res* and *modem* to *STB*. The other wire names are not used. Unused ports must be hidden and ports that go to the external world must be renamed.



	Media needs	$\sum \Delta$	Required $\sum \Delta$
$path_1$	cable	2	$\leqslant 2$
$path_2$	the_sign	1	$\leqslant 3$
$path_3$	happy_puppy	1	$\leqslant 4$
$path_4$	updown	3	$\leqslant 4$

Figure 6. Four paths and "wire names"

$$
\begin{array}{l}
STB' = STB[cable/channel, \\
\qquad updown/exec, w4c/x, w1/v] \\
\quad \ll control : channel2v\_x2exec \\
\quad \backslash\{control, x, y\} \\
hi\_res\&PIP' = hi\_res\&PIP[w1/x] \\
\quad \ll control : x2screen \\
\quad \backslash\{control, u, PIP\} \\
RC' = RC[w4a/u] \\
\quad \ll control : button2u \\
\quad \backslash\{control, x\} \\
lo\_res' = lo\_res[w4a/y, w4b/u, w2/x, screen'/screen] \\
\quad \ll control : x2screen\_y2u \\
\quad \backslash\{control, v\} \\
modem' = modem[happy\_puppy/url_1, the\_sign/url_2, \\
\qquad w4c/u, w3/v, w4b/z, w2/w] \\
\quad \ll control : url_1 2v\_url_2 2w\_z2u \\
\quad \backslash\{control, url_3\} \\
level2\_robot' = level2\_robot[w3/x] \\
\quad \ll control : x2moving \\
\quad \backslash\{control, u, feeling\}
\end{array}
$$

Finally the whole system implementation is described by

$$
\begin{array}{l}
SYSTEM' = \{STB' \otimes hi\_res\&PIP' \otimes RC' \otimes modem' \\
\qquad \otimes level2\_robot\}
\end{array}
$$

which has the syntactic interface (I', O'), where

$$
\begin{array}{l}
I' = \{cable, the\_sign, happy\_puppy, button\} \\
O' = \{screen, screen', moving, updown\}
\end{array}
$$

6 Correctness

The next questions is: how to express the formal correctness of the implementation?

Broy proposes three ideas of refinement: property refinement, glass box refinement, and interaction refinement. The glass box refinement is a classical concept of refinement typically used to decompose a system with a specified black box behavior into a distributed system architecture in the design phase, which seems appropriate in our case. The general form of a glass box refinement is

$$F_1 \oplus F_2 \oplus \cdots \oplus F_n \subseteq F$$

(In Broy's theory (see [2]), there is also another form for state machines, which we don't need here). The relation $\subseteq$ on component behaviors is defined by the rule that $\hat{F} \subseteq F$ stands for the proposition $\forall x : \overset{\dagger}{I} \bullet \hat{F}.x \subseteq F.x$, where

$$F : \overset{\dagger}{I} \rightarrow \mathcal{P}(\overset{\dagger}{O}), \hat{F} : \overset{\dagger}{I} \rightarrow \mathcal{P}(\overset{\dagger}{O}).$$

Also recall the definition of $\oplus$ for $F_1 \oplus F_2$

$$(F_1 \oplus F_2).x = \{y \upharpoonright O : y \upharpoonright I = x \upharpoonright I$$
$$\wedge\, y \upharpoonright O_1 = F_1(x \upharpoonright I_1) \wedge y \upharpoonright O_2 = F_2(x \upharpoonright I_2)\}$$

Let $RATE'$ be given by

$$\boxed{\begin{array}{l} rate(x.cable, 100, 100\text{K}) \\ rate(x.happy\_puppy, 10, 100) \\ rate(x.the\_sign, 2, 5K) \\ rate(x.button, 1, 1) \end{array}}$$

Note that this is essentially the same as $S_1, ..., S_4$ are saying about their z ports.

Consider an arbitrary x satisfying $RATE'$ where $x \in \overset{\dagger}{I'}$ where I' is $\{cable, ...\}$. x is a channel valuation $x : I' \rightarrow (M^*)^\infty$ such that

$$\forall i : I' \bullet x.i \in (type(i)^*)^\infty$$

where $M = \bigcup_{s \in S} s = PAL \cup MPG \cup DBH \cup IBH \cup \mathbb{R}$

Since I' has only four elements, we can write this out by assuming

$$\begin{array}{ll} x : cable \mapsto x_u & (x_u \in (PAL^*)^\infty) \\ x : happy\_puppy \mapsto x_d & (x_d \in (DBH^*)^\infty) \\ x : the\_sign \mapsto x_c & (x_c \in (MPG^*)^\infty) \\ x : button \mapsto x_b & (x_b \in (\mathbb{R}^*)^\infty) \end{array}$$

Now this x satisfies $RATE'$, that is, x_u has 100 frames per second, each frame being 100K bits, etc. (And similarly, x_d: 10, 100 respectively, x_c: 2, 5K respectively and x_b: 1, 1 respectively.)

Let $y \in SYSTEM'.x$ where $y \in \overset{\dagger}{O'}$ where O' is $\{screen, screen', moving, updown\}$ which we write out by assuming

$$\begin{array}{ll} y : updown \mapsto y_i & (y_i \in (IBH^*)^\infty) \\ y : screen \mapsto y_s & (y_s \in (\mathbb{R}^*)^\infty) \\ y : screen' \mapsto y_{s'} & (y_{s'} \in (\mathbb{R}^*)^\infty) \\ y : moving \mapsto y_m & (y_m \in (\mathbb{R}^*)^\infty) \end{array}$$

The correctness requirement is

$$RATE' \wedge SYSTEM' \subseteq SYSTEM$$

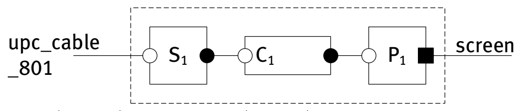
because $SYSTEM'$ only specifies maximal rates where as $SYSTEM$ has already fixed the rates.

The obvious proof strategy is to take an arbitrary x satisfying $RATE'$ and consider a $y \in SYSTEM'.x$, i.e.

$$y \in (STB' \oplus hi\_res\&PIP' \oplus \cdots) \setminus \{w_1, w_2, ...\}).x$$

and then check that $y \in SYSTEM.x$, which essentially boils down to checking *maxdelay* constraints and checking the essential transformations of the form pr'' for each of the specification paths, e.g. when going from *cable* to *screen*, or when going from *the_sign* to *screen'*.

As an example we check that y_s satisfies all constraints of $SYSTEM.x$. Since $SYSTEM$ falls apart in 4 unconnected parts, it is enough to check that $y_s = y.screen$ satisfies the the constraints of $(S_1 \circ C_1 \circ P_1) \setminus \{z, w\}$:



$$\exists\, z : (PAL^*)^\infty \bullet \exists\, w : (PAL^*)^\infty \bullet$$
$$z = x.cable \wedge rate(z, 100, 100\text{K})$$
$$\wedge\, maxdelay(z, w, 2) \wedge y.screen = pr''(w)$$

which is equivalent to:

$$rate(x.cable, 100, 100\text{K})$$
$$\wedge\, maxdelay(pr''(x.cable), y.screen, 2)$$

(since pr'' works frame-wise.)

What we know about y_s comes from $RATE' \wedge SYSTEM'$ which for $RATE'$ means $rate(cable, 100, 100\text{K})$. For $SYSTEM'$, its meaning is more complicated. First, note that

$$SYSTEM' : \overset{\dagger}{I} \rightarrow \mathcal{P}(\overset{\dagger}{O})$$

also, $SYSTEM' = (STB' \oplus hi\_res\&PIP' \oplus \cdots)$, where

$$STB' = STB[cable/channel, w1/v, ...]$$
$$\ll control : channel2v\_x2exec$$
$$\setminus \{control, x, y\}$$
$$hi\_res\&PIP' = hi\_res\&PIP[w1/x]$$
$$\ll control : x2screen$$
$$\setminus \{control, u, PIP\}$$

Note that for y_s it suffices to focus on STB' and $hi\_res\&PIP$ and forget about all other channels than w_1 where $\oplus$ can be replaced by $\circ$, i.e.

$$SYSTEM' = hi\_res\&PIP' \circ STB'$$

We summarize the assertions from $SYSTEM'$ by performing the renamings and keeping only the relevant clauses in view of the chosen control command. STB' says:

$$maxrate(x.cable, 100\text{M})$$
$$\wedge\, delay(x.cable, w_1 \copyright PAL, \underbrace{\Delta_{STB}}_{1})$$

$hi\_res\&PIP'$ says

$$maxrate(w_1, 100\text{M})$$
$$\wedge\, delay(pr''(w_1 \copyright PAL), y.screen, \underbrace{\Delta_{hi\_res\&PIP}}_{1})$$

from $rate(x.cable, 100, 100\text{K})$ we decide that for all t

$$\#(x.cable.t) = 100$$

and for each i, the i-th frame has

$$\#bits(x.cable.t.i) = 100\text{K}$$

therefore

$$\sum_{i=1}^{100} \#bits(x.cable.t.i) = 10\text{M}$$

so the *maxrate* requirement of STB' saying

$$\sum_{i=1}^{\#(cable.t)} \#bits(cable.t.i) < 100\text{M}$$

is not contradicted.

Next we check the delays. We combine the two one-step delay assertions, first by noting that pr'' works frame-wise:

$$\begin{cases} delay(pr''(x.cable), pr''(w_1 \circledcirc PAL), 1) \\ delay(pr''(w_1 \circledcirc PAL), screen, 1) \end{cases}$$

and secondly by adding the delays:

$$delay(pr''(x.cable, y.screen, 2)$$

which satisfies the clause from *SYSTEM* that

$$maxdelay(pr''(x.cable), y.screen, 2)$$

as required. The other *SYSTEM* constraints can be checked in a similar manner.

7 Concluding remarks

The case study shows how Broy's framework can deal with special interfaces for control and events at an abstract level and with real-world interfaces. It can also deal with bandwidth requirements and network delays very well. The case study does not have a sophisticated performance model, but it is plausible that certain models can be made using the same modelling style (for example if the delay is a function of the bit rate).

The case study shows one specification and one configuration of presentation resources. Abstractly, the mapping between the two is a set of paths through the network. Concretely, a mapping is a set of control commands, to be given to those components that have switching capabilities. Maximal rate and delay requirements can be checked formally, although the calculations are not surprising. Media types are described by sets such as *PAL* and *MPG*, this implies that type compatibilities are handled formally too.

The model created focuses on the stream-based aspects; in this type of distributed systems these aspects are most important at the architectural level. At a protocol level, we expect that reactive behavior is most important, and complementary state-machine based models could be used [12].

The following research questions remain as options for future research:

1. How to model dynamic aspects such as changing configurations and servers that build presentations according to the "factory" paradigm;
2. How to scale-up the approach when more and more technicalities have to be modelled (while keeping the math away from media developers);
3. How to specify the general class of "mapping problems" (instead of a single instance, as we did now).

Acknowledgment

The authors would like to thank Emile Aarts and Maddy Janse from Philips Research Eindhoven for their support on the work and Yuechen Qian for his comments on the early draft of this paper.

References

[1] E. Aarts and S. Marzano. *The New Everyday View on Ambient Intelligence*. Uitgeverij 010 Publishers, 2003.

[2] M. Broy. A logical basis for component-based systems engineering. In M. Broy and R. Steinbr uggen, editors, *Calculational System Design*. IOS Press, 1999.

[3] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. *Pattern-Oriented Software Architecture, A System of Patterns*. John Wiley & Sons, Inc., Chichester, UK, 1996.

[4] W. Eixelsberger and H. Gall. Describing software architectures by system structure and properties. In *COMPSAC '98 - 22nd International Computer Software and Applications Conference*, pages 106–111, Vienna, Austria, 1998. IEEE Computer Society.

[5] L. M. G. Feijs and Y. Qian. Component algebra. *Science of Computer Programming*, 42(2-3):173–228, 2002.

[6] J. Hu and L. M. G. Feijs. An adaptive architecture for presenting interactive media onto distributed interfaces. In *The 21st IASTED International Conference on Applied Informatics (AI 2003)*, pages 899–904, Innsbruck, Austria, 2003. ACTA Press.

[7] J. Hu and L. M. G. Feijs. An agent-based architecture for distributed interfaces and timed media in a storytelling application. In *The 2nd International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1012–1013, Melbourne, Australia, 2003.

[8] IBC. The world of content creation, management and delivery, 2003. Available from: http://www.ibc.org.

[9] ICE-CREAM. The ice-cream project homepage, 2003. Available from: http://www.extra.research.philips.com /euprojects /icecream/.

[10] C. Kray, A. Krüger, and C. Endres. Some issues on presentations in intelligent environments. In E. Aarts, R. Collier, E. van Loenen, and B. de Ruyter, editors, *First European Symposium on Ambient Intelligence (EUSAI)*, pages 15–26, Veldhoven, The Netherlands, 2003. Springer.

[11] NexTV. The nextv project homepage, 2001. Available from: http://www.extra.research.philips.com /euprojects /nextv.

[12] A. Ulrich and H. König. Specification-based testing of concurrent systems. In A. Togashi, T. Mizuno, N. Shiratori, and T. Higashino, editors, *Joint International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols (FORTE 1997)*, pages 7–22. Chapman & Hall, 1997.