Self-aware Middleware for Smart Camera Networks

Herwig Guggi
Pervasive Computing Group
Institute of Networked and Embedded Systems
Klagenfurt University
Email: herwig.guggi@aau.at

***Abstract*— This paper presents a middleware architecture for smart camera networks that is based on a pipe-and-filter model augmented with self-awareness and self-expression. These properties enable the system to move from a procedural design methodology towards a self-aware system which is able to inherently react on changing conditions from the environment.**

***Index Terms*—Smart camera networks; Middleware; Self-awareness; Self-expression**

I. Background

During the last years, hardware devices have become more and more powerful. Smart cameras are one example of this trend. They combine image-sensing, processing and networking. While single cameras can be used to trigger events and support a human observer in a surveillance system [1] or perform vehicle detection and speed estimation [2], distributed smart camera networks offer an even higher benefit. They can be used to detect obstacles to avoid collisions [3] or perform a cooperative tracking with local image analysis [4].

Designing, implementing and deploying applications for distributed execution is much more complex than for single device systems. On general-purpose platforms, distributed applications are often based on a middleware system which provides services for networking and data transfer [5].

While there are systems that support a developer of a distributed application to access sensor data like ICE middleware [6], there is no reasonable system-level software available that is able to support cooperation and collaboration within a distributed sensor network.

Adaptive middleware systems can be used to implement applications that adapt to current system requirements. Rahnama et. al. [7] describe such an adaptive system and compare it to a non-adaptive implementation. Another example for is shown by Hurtado et. al [8]. Their middleware system is based on the Open Service Gateway Interface (OSGi). This interface provides a universal publish-subscribe based protocol which is used to dynamically load and unload components during runtime. The described systems [7] and [8] do not modify parameters of single components to adapt to new conditions. They instead exchange components and modify the behaviour of the system.

II. Middleware structure

The idea of the proposed middleware is as described by Lewis et al. [9] *"to move from a procedural design methodology wherein the behaviour of the computing system is pre-programmed or considered beforehead (i.e., at design time), towards a self-aware system where this is not required and the system adapts to its context at run-time."*

To achieve this goal, a standard approach is extended by self-awareness and self-expression-properties as defined by Parsons et al. [10]: " *To be Self-Aware a node must contain total information about its internal state along with enough knowledge of its environment to determine the current state of the system as a whole. It may either be focused on its own state or the environments state at any time, but not both at once. In a systems where some nodes may have more importance each node must also be aware of its own importance within the system.* [...] *A node is said to have Self-Expression if it is able to assert its behaviours upon either itself or other nodes, this behaviour is based upon a nodes sense of its personality*"

The developer implements the functionality using the pipe-and-filter concept. This concept is typically used in sensor network applications. The middleware, if necessary, splits the software to sub-sets of pipes-and-filters and deploys them to one or more hardware devices. The middleware splits this software into sub parts in a way that the overall system performance is maximized and deploys the filters to the available hardware devices.

To go a step towards a self-aware system, two additional filters have been added to each hardware device. They provide self-awareness and self-expression mechanisms. More details are given in section III.

The self-awareness and self-expression principles can also be implemented on a lower level of the system. All filters that are used within the software can be self-aware on their own. More details are given in section IV.

III. Self-aware resource utilisation

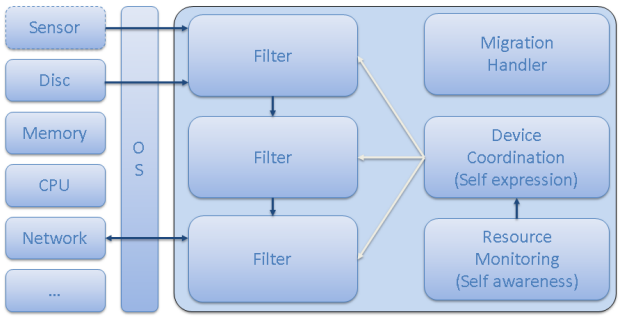

Fig. 1. Pipes and filters on a single device dark arrows: data-flow, light arrows: control information

Figure 1 visualizes the part of the software that is executed on a single hardware-device. The elements named "*Filter*" are the parts of the software which is implemented by the developer using the pipe-and-filter concept. The filter named "*Device Coordination*" is responsible for the coordination of the pipes and filters on this specific device. Its job is to modify the pipe-and-filter configuration to meet changed requirements. These modifications include the deployment of additional pipes and filters and, if necessary, the delegation of functionality to other devices. The element named "*Resource Monitoring*" implements self-awareness properties. These self-aware properties might provide data like the CPU utilisation, memory utilisation and available network bandwidth. The "*Device Coordination*" can show self-expressive behavior by taking into account knowledge of the internal system state which is provided by the "*Resource Monitoring*" element.

IV. Self-aware filters

A self-aware filter extends the standard filter from the pipes-and-filter concept by self-awareness and self-expression mechanisms. Each filter in our approach has a list of input filters (filters that provide data to the current filter) and output filters (filters that receive data from the current filter). After initialisation, the filters work as intended. During the execution of a dynamic application like in sensor networks, it will happen that one filter requires a parameter to be modified (e.g., a change in resolution for image data). This request is backwarded to the filter which provides the data and is autonomously processed by that filter. The detection of this requirement relates to the self-awareness principle. After detection, the filter itself checks if it is able to meet the new requirements and in case that it is possible, the parameters are automatically tuned. This automatic re-configuration of the parameter represents the self-expressive principle.

Let us assume an imaging application consisting of two filters on two devices, one image capture filter and one image display filter. In the initial setup, the images would be captured and displayed in full resolution. At some point the user might reduce the window size of the image display. To be resource efficient, the image should already be transmitted in the reduced resolution. A state of the art middleware would require the developer to handle this event and tune the parameter of all filters (in our case only from the image capture filter) to adapt to this modification.

In the case of our proposed self-aware filter, the display filter would automatically detect the modification of the display area and inform its input filter (in our case the image capture filter) that a lower image resolution would be sufficient. This filter recognises the request for a modification of a parameter and reacts on this request by automatically modifying this parameter. After the modification, images with a reduced resolution will be forwarded. This approach allows distributed automated optimisation of filter parameters to requirements. This automatic adaption is optimizing the overall performance of the system as the amount of data is reduced as close to the data source as possible. This means that no data is transmitted via the network which would be dropped at the destination. This self-aware behavior automatically reduces the required bandwidth to a minimum.

V. Conclusion

The middleware architecture presented in this paper extends the state-of-the-art pipes-and-filter concept by self-aware and self-expressive properties. These properties are implemented in two levels of the system. First the individual filters and second the device coordination. Self-aware filters enable fast and efficient context-based modification of filter parameters. Self-aware device coordination supports the developer with decisions on filter migration or delegation of functionality to other devices. This enables the system to react inherently on changing conditions from the environment.

Acknowledgment

The research leading to these results has received funding from the European Union Seventh Framework Programme under grant agreement no 257906. The author would like to thank Bernhard Rinner and Lukas Esterle for valuable comments.

References

[1] M. Bramberger, J. Brunner, B. Rinner, and H. Schwabach, "Real-time video analysis on an embedded smart camera for traffic surveillance," in *Real-Time and Embedded Technology and Applications Symposium, 2004. Proceedings. RTAS 2004. 10th IEEE*, may 2004, pp. 174 – 181.

[2] D. Bauer, A. N. Belbachir, N. Donath, G. Gritsch, B. Kohn, M. Litzenberger, C. Posch, P. Schön, and S. Schraml, "Embedded vehicle speed estimation system using an asynchronous temporal contrast vision sensor," *EURASIP J. Embedded Syst.*, vol. 2007, pp. 34–34, January 2007.

[3] H. Guggi and B. Rinner, "Distributed smart cameras for hard real-time obstacle detection in control applications," in *Distributed Smart Cameras (ICDSC), 2011 Fifth ACM/IEEE International Conference on*, aug. 2011, pp. 1 –6.

[4] M. Bramberger, A. Dobl, A. Maier, and B. Rinner, "Distributed embedded smart cameras for surveillance applications," *Computer*, vol. 39, p. 2006, 2006.

[5] D. C. Schmidt, "Middleware for real-time and embedded systems," *Commun. ACM*, vol. 45, pp. 43–48, June 2002.

[6] ICE Middleware. [Online]. Available: http://www.zeroc.com/

[7] H. Rahnama, P. Kramaric, A. Sadeghian, and A. Shepard, "Self-adaptive middleware for the design of context-aware software applications in public transit systems," in *Proceedings of the 13th international conference on Ubiquitous computing*, ser. UbiComp '11. New York, NY, USA: ACM, 2011, pp. 491–492.

[8] S. Hurtado, S. Sen, and R. Casallas, "Reusing legacy software in a self-adaptive middleware framework," in *Adaptive and Reflective Middleware on Proceedings of the International Workshop*, ser. ARM '11. New York, NY, USA: ACM, 2011, pp. 29–35.

[9] P. R. Lewis, A. Chandra, S. Parsons, E. Robinson, K. Glette, R. Bahsoon, J. Torresen, and X. Yao, "A Survey of Self-Awareness and Its Application in Computing Systems," in *Proc. Int. Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, 2011.

[10] S. Parsons, R. Bahsoon, P. R. Lewis, and X. Yao, "Towards a Better Understanding of Self-Awareness and Self-Expression within software systems," University of Birmingham, School of Computer Science, UK, Tech. Rep. CSR-11-03, Apr. 2011.