Construct

An Open Source Pervasive Systems Platform

Simon Dobson, Paddy Nixon, Lorcan Coyle, Steve Neely, Greame Stevenson and Graham Williamson
Systems Research Group
Adaptive Information Cluster
UCD School of COmputer Science and Informatics
UCD Belfield, Dublin, Ireland
Firstname.Lastname@ucd.ie

Construct differs from other pervasive systems platforms in a number of key respects. It is completely standards-based, using RDF as its data exchange model and ZeroConf for resource discovery. It supports a knowledge-centric model of interaction where clients' actions are driven by queries and triggers about the context of the system. It uses gossiping to maintain a consistent state across a distributed data structure, which maximises robustness and scalability and avoids many problems with hot-spots and hot-paths in communications. Finally, it treats all information sources uniformly as sensors acting as inputs to uncertain reasoning algorithms.

Keywords-component; autonomic, pervasive, context aware, middleware.

I.   INTRODUCTION

Pervasive and autonomic systems share many features in common: they are highly adaptive, involve a large and dynamically-changing population of components and services, must deal with a variety of sensors and information sources delivering partial and uncertain results and must deliver an overall experience which is simultaneously adaptive to changing context but stable enough to present a predictable and scrutable service to users and other systems. Building systems which meet these criteria is challenging, involves considerable infrastructural development work, and requires the designer to understand a wide range of quite subtle issues -- all of which interfere with the development of application-level services.

It is attractive to address as many of the infrastructural issues as possible using middleware, providing a common platform on which to construct more advanced applications and services. A number of such platforms heve been developed: the Context Toolkit, Project Aura, Cooltown, Semantic Space and others. Each has been highly influential, but also presents challenges (of maintenance, openness, availability, scalability or abstraction) for new projects. We have designed Construct to address these issues. The goals of Construct are:

- to provide an open, standards-based platform to encourage and facilitate the development of pervasive and autonomic systems;

- to simplify the development of novel systems by providing a collection of pre-built sensors, modules and services addressing important system features;

- to act as a target for research into adaptive systems design, and into programming language constructs for such systems; and

- to nurture a community of developers who can build on each others' work.

Construct differs from other pervasive systems platforms in a number of key respects. It is completely standards-based, using RDF as its data exchange model and ZeroConf for resource discovery. It supports a knowledge-centric model of interaction where clients' actions are driven by queries and triggers about the context of the system. It uses gossiping to maintain a consistent state across a distributed data structure, which maximises robustness and scalability and avoids many problems with hot-spots and hot-paths in communications. Finally, it treats all information sources uniformly as sensors acting as inputs to uncertain reasoning algorithms.

II.   FUNDAMENTALS

The majority of the codebase is written in Java. At the heart of the infrastructure is a component loader which instantiates the five core services at runtime. As new peers come online they discover other nodes using an implementation of the zeroconf protocol [7]. When entities start-up they also use zeroconf to locate instances of Construct. The result of this is a reference to the discovery service that supplies the manifests for additional local services. Manifests are described using the Web Service Definition Language (WSDL), which provides information on how to connect to and use each service. All data in Construct are modelled using the Resource Description Framework (RDF). RDF is an open-standard that meets our requirement of providing a common language with which to represent information generated within a networked home.

Construct's implementation of the data-manager uses the Jena framework. Jena is a well established tool from the semantic web community that provides a rich interface for manipulating RDF data. The data-manager maintains two models, which contain the entity supplied data and it's associated metadata.

Construct's data-in service exposes a data port that accepts RDF documents in N3 format from data providers. Its data-out service provides support for querying data using the SPARQL and RDQL query languages.

The implementation of a gossiping algorithm in the data distribution service takes the form of a protocol stack with three layers: the data-layer, which exchanges RDF with the data-manager; the gossiping layer, which drives the process; and the network layer which encapsulates the actual sending and receiving of raw packets of bytes. The gossiping subsystem can be decomposed into three components: the core gossiping protocol, the message buffer, and the peer membership manager. The core gossiping protocol is stateless. Gossiping is initiated periodically at each peer by sending their message buffer summary to a randomly selected peer from their contact list. When messages arrive they are identified and processed by type. The gossiping layer handles three types of messages: data messages, message buffer summaries, and individual message requests. Data messages are stored in the message buffer for future gossiping then passed up the protocol stack to the data-manager. Message buffer summaries contain a list of the message IDs present in the remote node's buffer. On comparison with the local message buffer requests for messages that are not held locally are generated. These requests are fulfilled by return. Construct is designed to support an extensible repository of entities and associated ontologies. This includes support for location data obtained from Ubisense, Place Lab, Bluetooth spotters, GPS, hearsay sources in the form of calendar data, and a computer activity monitor. Other data sources include virtual sensors accessing web-based information sources for weather, flights, concert tickets, music reviews, online shopping, news, sport, and television listings. These entities have been written in a number of languages including Python, Ruby, Java, C#, and C++. Applications that operate over the data from these sources employ a variety of output devices such as web pages, mobile phones, wall-mounted displays and Nabaztags.

III. CONCLUSIONS

Construct provides an experimental platform for pervasive systems with a flexible set of features that may make it a candidate for home networking applications: the core system focuses on scalability, openness and extensibility, allowing the community to build home network-oriented sensors and services. The use of gossiping and information replication provide a robust and scalable infrastructure for knowledge-driven embedded intelligence. Our current work focuses on enhancing the sensors and fusion algorithms available within Construct, and providing the facilities needed for long-term deployment and in situ update. In particular we are exploring embedded intelligence, trust models that can control information propagation, proof-carrying code techniques for on the-fly maintenance, and techniques for specifying and analysing self-managed and self-optimising behaviours.

Construct has been released under an open source license and more information can be found at: `http://www.construct-infrastructure.org/`.

REFERENCES

[1] N. Streitz and P. Nixon, "The disappearing computer," Communications of the ACM, vol. 48, no. 3, March 2005.
[2] P. T. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massoulieacute, "Epidemic information dissemination in distributed systems," Computer, vol. 37, no. 5, pp. 60–67, 2004.
[3] P. Eugster, P. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," ACM Computing Surveys, vol. 35, no. 2, pp. 114–131, 2003.
[4] S. Dobson, S. Denazis, A. Fern´andez, D. Ga¨iti, E. Gelenbe, F. Massacci, P. Nixon, F. Saffre, N. Schmidt, and F. Zambonelli, "A survey of autonomic communications," ACM Transactions on Autonomous and Adaptive Systems, vol. 1, no. 2, December 2006, to appear.
[5] P. Nixon, W. Wagealla, C. English, and S. Terzis, Privacy, Security, and Trust Issues in Smart Environments. Wiley, October 2004, ch. Smart Environments: Technology, Protocols and Applications, pp. 220–240.
[6] S. Intille, K. Larson, E. M. Tapia, J. Beaudin, P. Kaushik, J. Nawyn, and R. Rockinson, "Using a live-in laboratory for ubiquitous computing research," in Procedings of The 4th International Conference, PERVASIVE 2006, ser. LNCS. Springer-Verlag Inc., May 2006, pp. 349–365.
[7] E. Guttman, "Autoconfiguration for IP networking: Enabling local communication," IEEE Internet Computing, vol. 5, no. 3, pp. 81–86, 2001.
[8] L. Coyle, S. Neely, G. Rey, G. Stevenson, M. Sullivan, S. Dobson, and P. Nixon, "Sensor fusion-based middleware for assisted living," in Proceedings of The 1st International Conference On Smart homes & heath Telematics (ICOST'2006) "Smart Homes and Beyond". IOS Press, 2006, pp. 281–288.
[9] L. Coyle, E. Balfe, G. Stevenson, S. Neely, S. Dobson, P. Nixon, and B. Smyth, "Supplementing case-based recommenders with context data," in Proceedings of The 1st International Workshop on Case-based Reasoning and Context Awareness at ECCBR 2006., O¨ lu¨deniz/Fethiye, Turkey, September 2006.