The Breathe System

Peter Seymour

2 October 2008

1 Introduction

The Breathe System is a realisation of an abstract description for ubiquitous computing. By computing we mean all activities that are regarded as usefully performed by computing gadgets and processes in society today. This is different from a study of computation that seeks only to produce a theoretical view of computers and not their human-oriented use in a wider sense.

The system is constructed out of various concepts each of which can have many different concrete or physical implementations. Some will be best represented by dedicated or general electronic gadgets or perhaps in a form not currently considered. What is specified though is the interaction between these concepts and a detailed description of their activities.

It would not be practical to be forced to describe every aspect of computing just to solve a local problem. So while the Breathe System is a potential total system it is not necessary to use it in this way. Existing technologies can be incorporated by only studying their behaviours. For instance a printer can be usefully used without requiring a detailed analysis of its internals. However, should such a breakdown be required it can be supplied within the system.

2 Goals

The end result is a universal method for describing computing that catalogues existing technologies and solutions while enabling new features to be integrated. As concepts are implemented they can be clearly documented and re-used resulting in cleaner abstractions. The traditional view of the field focuses on the divide between hardware and software but this is largely arbitrary and avoided here. It is possible to provide software, hardware or even alternative implementations (such as human computers) so long as the description is adhered to.

A second goal is to make the description meaningful to humans. Some of the complexity in computing is of an essential nature deriving from its place in mathematics. We can see this in the models used to drive computations in the Breathe System, they are completely abstract. However, a far more potent source of complexity arises through bending computing towards human needs and whims. This is reflected in the system by the intricate arrangement of concepts and interactions between them as we see in societies around the world.

A third goal is to be able to separate the model of computation from the resulting computing system. By allowing different models to co-exist means that computational problems can be solved in the most appropriate way without impacting the overall architecture.

3 Concepts

This paper explores the concepts one-by-one giving a few details of each. For more detailed information see the earlier sections of [1]. There are two ways in which the system can be viewed: top-down and bottom-up.

3.1 Top-down

When we consider the top-down view we see a directed hypergraph as shown in Figure 1. The nodes are computing resources called cells and the connections form a web of directional communication. The problems in this domain are of organisation and location. By repeatedly partitioning the resources they become more easily nameable and reflect the society in which they exist.

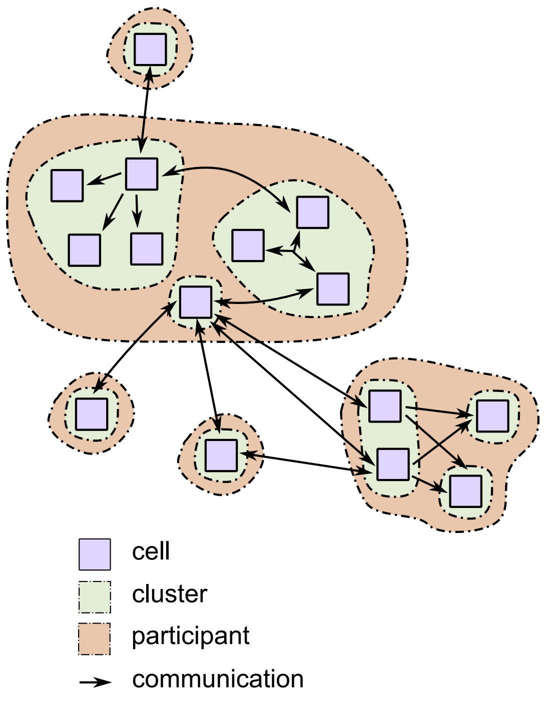

Figure 1: The Breathe hypergraph

This view would correspond to how we might see today's networks of the Internet, phone system, mail system or even conversations in a crowded room. Each of the concepts is explored in a little more detail.

3.2 Participants

The evolution of this hypergraph is driven by some motivation to achieve something. This motivation may be from an individual acting alone or some form of organisation, these are the participants. The set of participants partitions the system into logical subsets consisting of their cells.

3.3 Clusters

Each participant has a set of aims for what they wish to achieve and may group resources accordingly. These aims further partition the sub-system defined by a participant into clusters of like resources. For instance all the telephones in a company, a distributed database or even all teachers in a school.

3.4 Cells

These are units of computation power and actually perform calculations. They are the logical nodes of the hypergraph. Note that they do not need to directly correspond to any physical object for instance a collection of virtual machines would suffice. A cell will compute and as a result produce and receive communications with other cells. Examples are electronic devices, emulation software or perhaps a person.

3.5 Connections

Connections are communication paths between cells i.e. edges in the hypergraph. These are non-transitive meaning that a connection from A to B and one from B to C does not imply a connection from A to C. Each connection is directed with one or more sources and one or more targets where these sets intersect we have bi-directional communication.

3.6 Networks

As cells are partitioned into clusters, connections are partitioned into networks. For each network there is a name and a method for communication that is shared by all connections. Networks are connected (there is a path between all cells on the network) so distinct networks may share the same name even if their methods differ, see Figure 2. Each cell may appear on multiple networks. A method determines how information is communicated along a connection. For instance sending bytes of data, posting letters in the mail or perhaps verbal communication. The topology allows for simultaneous communication (talking at the same time) as well as broadcast (talking to a group).

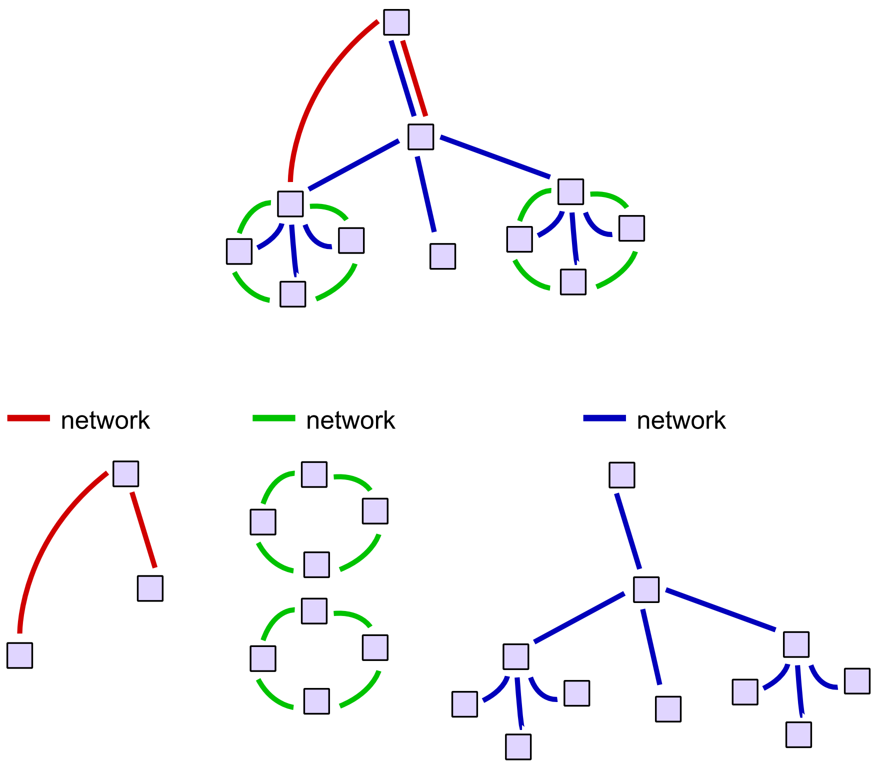

Figure 2: Breathe Networks

3.7 Bottom-up

We look now at cells in isolation from their environment, see Figure 3. In this direction we see the driving force behind the computations that are present within the cell. Unlike the top-down approach this view corresponds more closely to abstract computation and takes a more mathematical view of the world.

3.8 Model Instances

A model of computing is required to describe processes (a sequence of computations). Instances of such a model describe the actual computations that will take place. The distinction between model and instance is that of a programming language and a program or a natural language and a conversation transcript. An application or delivery of a transcript would correspond to a running model instance.

Although a model instance describes a single path of execution (serial computations) it may simulate the behaviour of multiple parallel processes. A model

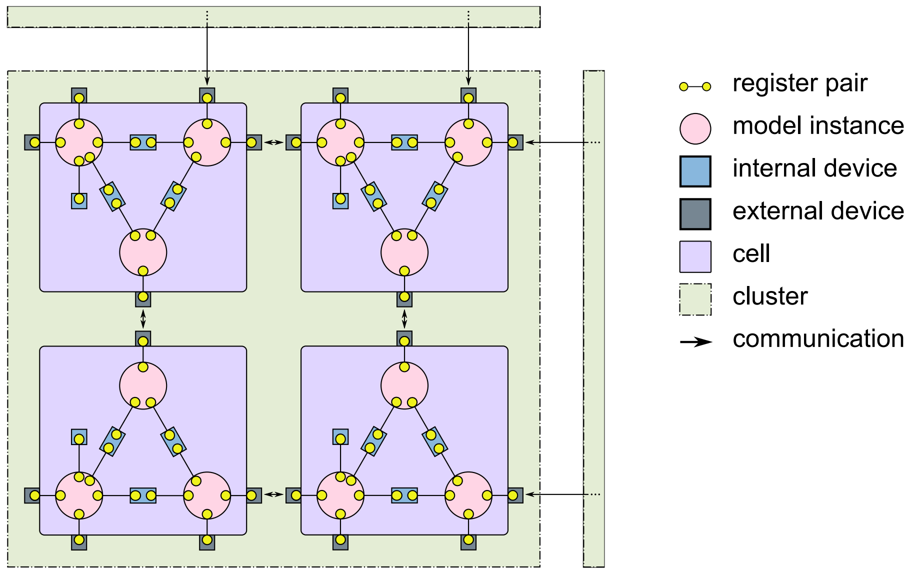

Figure 3: Detailed cells in a cluster

instance specifies communication with its external environment through registers and signals made available by its host cell. For more information on registers and signals see [1] and [2] where [2] describes a specific but general purpose model. Any model that subscribes to the interface implied (see [1]) by the registers and signals of the host cell is sufficient.

3.9 Devices

There are two types of devices but a model instance interacts with them through the same mechanism, namely synchronisation, see [1] and [2]. A device is an abstraction of some other area of computing for instance a physical device or another cell on a network. They represent all that is known to a model instance about its environment and allow systems to be fitted together without describing the complete picture. Devices may or may not have their internal workings described within the Breathe System as is convenient. When a model instance starts running it binds to all the devices it will need.

A device may be active or inactive although available. If a device becomes inactive (for example unplugging a printer) the registers will still be bound but outward synchronisation will have no effect and inward synchronisations will see previous register states. This is useful for handling disruptions and for speculative binding to multiple similar devices in order to find the best active one to use.

3.9.1 External Devices

An external device is bound to one and only one model instance, however, a cell may present more devices to serve multiple model instances. These devices correspond roughly to physical devices. Examples are keyboards, monitors, printers or a post box for posting mail.

A piece of software such as an operating system supporting an implementation of a cell could be an external device. In such a situation the operating system can present itself as an external device along with other software devices.

3.9.2 Internal Devices

An internal device can be bound to multiple model instances but exists uniquely withing a cell.

As multiple running instances attempt to bind to an internal device the device prepares a new set of registers and a signal channel specifically for that instance. A simple internal device may allow communication between model instances in the same cell for instance a pipe or to allow sharing of an external device.

References

[1] “A Compendium Of Devices”, P. Seymour, 2008.

[2] “The U*-model”, P. Seymour, 2008.