Towards Device-Blending: Model and Challenges

Harinder Seera, Seng Wai Loke and Torab Torabi
Department of Computer Science and Computer Engineering
La Trobe University, Australia
Email: {s.loke, t.torabi}@latrobe.edu.au, hpseera@students.latrobe.edu.au

Abstract

This position paper proposes a device-blending architecture for aggregating device functionality via inter-device peer-to-peer relationships, effectively forming a multi-device "distributed computer", as a collective for processing tasks for users. The system would provide users with more freedom when using the devices by hiding low level details of device interconnections and automating such connectivity. We describe the concept of device-blending, point out benefits of the architecture and examine implementation and software issues related to the model. We contend that inter-device relationships, as we introduce in this paper, forms an important platform for subsequent inter-device collaborations in fulfilling user tasks.

1. Introduction

In near future every home environment will be populated by different kinds of computing devices and each device varying in energy resources, memory capabilities, processing power and different means of user interaction. For example, we may have single task devices (example Toaster, washing machine) and multipurpose devices such as Computer, Mobile phone, Electrolux Screen Fridge [14], NCR microwave bank [14]. As pointed out by Mark Weiser [1],"the real power of the concept [of Ubiquitous Computing] comes not from any of these devices; it emerges from the interaction of all of them." Therefore, one of the core challenges is how the user will know how these devices will interact with each other and how s/he can configure these devices for his/her own liking. Another challenge is how to uses devices which have similar functions as others when those devices do not work and the final challenge is building applications that are generic enough to be used by most of the devices in an environment.

Different devices use different protocols, for example UPnP [2], Jini [3], Bluetooth [4], Salutation [5], SLP [6] and most of the devices only interact with devices of similar protocols but not with devices of different protocols. To solve this problem researchers have proposed different systems such as Interplay [7], EasyLiving [8], Cogma [9] , Infostick[10], Touch and Connect [11].

Interplay [7] is a middleware which allows users to simply select what activities they want to perform at home by means of pseudo sentence. This middleware requires a centralized server (current implementation uses TV as the control point). It is implemented in Java, it uses Owl (web ontology language) and RDF (resource description framework) to define the device and task description schemas. It currently has plug-ins to support UPnP protocols with AV (Audiovisual), content and printing services. This system has the following issues: Firstly, if the centralized system crashes then devices can not interact. Second, it can only be used in a home environment and not outside and lastly, the system has only been tested with UPnP devices.

EasyLiving project [8] is a "Microsoft research that is concerned with the development of architecture and technologies for intelligent environments." The EasyLiving system consists of an InConcert middleware, geometric modelling, sensing capabilities and service description. The InConcert middleware provides asynchronous message passing, machine independent addressing and XML- based message protocols so that programs can handle offline and queued operation more naturally.

Geometric modelling [8] "provides a general geometric service for ubiquitous computing, focusing on in-home or in-office tasks in which there are myriad I/O, perception and computing devices supporting multiple tasks." Sensing capabilities (sensors) are used as the source for the geometric model. Currently, this system has problems with the number of connection between services, which if increases, causes the polling to cease. The current lookup table is not robust enough to handle thousands of services continuously. The current system is not extensible enough to track the users between different disjoint spaces and the current system does not let user create and edit automatic behaviours.

Cogma [9] stands for Cooperative Gadget for mobile appliances. It has the following features [14]:

- de-centralized system,
- lightweight middleware,
- allows dynamic code/state transfer,
- simultaneously use two or more different type of network-link, and
- autonomous discovery mechanism of other nodes simplicity of management and communication mechanism of mobile software

This system currently doesn't handle the home devices which have already been installed.

Infostick[10] allows the user to transfer information between different information appliances or physical objects. It acts as information carrier. For Infostick to work, each object/device needs to have a visual marker which allows Infostick to give appropriate action to user that he can perform based on what that object is. The drawbacks with this system are: if the marker gets washed away/removed than the Infostick can not recognize the appliance or the object and also the user needs to know where the marker is located.

Touch and Connect [11] is a simple management framework which connects two networked devices by simply touching them. This framework does not require a server and consists of a lock mechanism which prevents incorrect connections caused by the users. Currently, devices that use this system may have a single button or multiple buttons on it to perform the task and this is one of the drawbacks as it requires the user to know all the steps involved when pressing these buttons. Also, the connect ability-relationship and the application between two device types are defined beforehand.

Based on the issues related to the above systems, this position paper introduces the concept of device-blending, together with its formalization (a *mathematical model) and architecture* for the home environment. The mathematical model shows how different devices interact in the environment, which devices do not interact and what kind of inter-device relationships exists between these devices. The device-blending architecture uses the mathematical model as the conceptual foundation for the implementation of a decentralized system that all the devices can use.

The rest of the paper is organised as follows. Section 2 outlines some of the scenarios that are common in the home environment. Section 3 outlines the key inter-device relationships which are the building blocks for the device blending mathematical model as well as our architectural design for the home environment. It also outlines the issues related to the mathematical model and the architecture. Section 4 concludes the paper with future directions.

2. Future Home Computer Scenarios and requirements

This section gives three different scenarios in a home environment and show different choices that a user can make.

In the first scenario, *Harry finishes typing a document using his laptop and now he wants to print the document, he finds out that his laptop is not connected to the printer but his desktop computer is. Harry has two choices to make:*

- *He can copy his document to the desktop computer*
- *He can connect the printer to the laptop*

In the second scenario, *Harry is in a sitting room and he wants to check his bank details and the only input device that he has with him is mobile phone which is not connected to the internet. So the only option is to log on to a device which has Internet connection in order to check his bank details, for example, a personal PC.*

In the last scenario, *Harry is watching, "the Secret window" movie on his computer and due to some hardware malfunction, his monitor stops working, so Harry can either:*

- *Wait till he gets a new monitor or gets his old one fixed*
- *Use his DVD player to watch his movie*

From the above three scenarios, we believe the following issues are important when designing devices for the home environment.

- Devices should have a knowledge about devices that have similar services
- Devices need not have a direct connection with other devices to use their services
- Most of the devices should be able to perform a user task
- User need not have deep knowledge about how devices are connected in order for him/her to take advantages of the services they provide

In this paper, we propose a mathematical model that

- allows the user to know how many devices interact in an environment, how they are interacting and what devices are not interacting, and
- serves as a building block for the decentralized device-blending architecture.(more on the device blending architecture in section 3.3)

3. Device-Blending

In this section we discuss the key technical concept of device blending and a mathematical model for device blending. Before we define what is device blending is we define inter-device relationships which are the building block for device blending.

The assumptions that we make for the model are as follows:

- Each device in the environment consists of services.
- A device can have none, one or more than one network connections.
- A device can make use of the services of another device to which it is connected.

3.1 Key Concepts – Inter device Relationships

In the real world or the environment we see different kinds of relationships that exist between human beings or objects. For example, A PhD student has to have at least two supervisors (main and co supervisor). So based on this we can have different possibilities:

- The main supervisor is absent for the meeting and therefore the co supervisor **substitutes** him
- Both supervisors are present and in that case each one can **enhance** the other by elaborating on an idea or on a suggestion

If we consider the devices as objects which interact with other objects based on some relationship, we can have the following inter-device relationships using the metaphor of relationships that exist in the real world.

Substitute (sb): A device is said to substitute another device if there exists a similar service that both devices provide and in the case where one of them cannot provide that service then the other one can. For example, both the TV and computer monitor are used for viewing movies; so in the case where the computer monitor stops working then the TV can substitute the monitor.

Dependent/Bridge (br): A device is said to be a bridge between two or more device when it is able to relay or pass a message/service back and forth between devices which are not directly(physically) connected together. For example, a computer can be a bridge between a mobile phone and a printer.

Enhance (eh): A device is said to enhance another device if it can provide a service which can be used by another existing service for its betterment or to enhance user pleasure. For example, an extra woofer & sub woofer can be used along with existing speakers for more enjoyment but they are not necessary to have.

Complement (cm): A device is said to complement another device, if one device's service can use another device's service and vice/versa to complete a given task.

Enable (cm): A device is said to enable another device, if the use of service in one device enables a service provided by another device. For example, when you open the door the alarm goes off.

Disable (ds): is similar to enable except it is the opposite. For example, when you leave your house, the kettle should automatically stop boiling the water.

Therefore we can say a device is blend-able with the environment if it satisfies the following criteria:

- A network **connection** exist between it and any other device and it **is recognizable** by that device
- There exists some services which can **complement/enhance/substitute/bridge/enable/disable** the existing services in the environment

Note that we do not make underlying assumptions about the actual network technology- the model is more general than that. We have a relationship model for devices having the blending properties.

Let D denote the set of devices in an environment.
D= {d1, d2, d3,…,dn} and Let R denote the set of blending relations that exist in an environment R= {eh, en, cm, sb, br, ds}.

Let F denote a device ecology involving devices in D having relationships from the possible relations in R. Therefore, we have F, defined over D and R, as follows:
$F(D,R) \subseteq (DxR'xD \cup Dx\{br, br^*, \tilde{br}\}xDxD)$
where $R' = R \backslash \{ br, br^*, \tilde{br} \}$, i.e. bridging relations involve three devices whereas the other relations are binary. The device ecology F(D, R) can be represented using what we call an inter-device relationship matrix (IDR matrix, for short).

Each inter-device relationship can be sub categorized as r, r* and $\tilde{r}$. The r stands for a uni-directional relationship between devices. The r* means that the relationship is bi-directional based on a single service that the devices provide to each other, and $\tilde{r}$ means that there exists a bi-directional relationship between devices but this relationship is based on different services they provide to each other and not the same service. For example, consider that, the computer monitor can substitute the TV but not vice versa. In this scenario we have a uni-directional substitution relationship (sb). If both devices can substitute each other for the same service than we have a sb* relationship; if one can substitute the other but on different services, then we have a $\tilde{sb}$ relationship.

Let D be a set of devices and R blending relations. Then an environment E is defined as a set of device ecologies F1, F2 ..., Fn over D and R. We have
E = {F1(D,R), F2(D,R), … , Fn(D,R)}, where Fi ≠ Fj for i ≠j.

IDR matrix is a matrix that shows different types of relationships that exist between different devices in an environment. The IDR matrix looks as follows (for example):

	D1	D2	D3	…	Dn
D1			R3	…	
D2	R1				
D3		R2			
…					
Dn					

Figure 1: IDR matrix

Bridging relationships involve three devices and are represented by entries in two cells (one for the two devices being bridged and another for the bridge device).
D1, D2, D3, Dn are the devices in the environment (D1, D2, D3… Dn ∈ D). R1, R2, R3 are different relationships that exist between the devices (R1, R2, R3 ∈ R). In an IDR matrix the row shows the direction in which the relationships exist.

For example, figure 2 is an IDR matrix for an environment E consisting of 2 devices A and B.

	A	B
A		
B	sb	

Figure 2: IDR matrix for 2 devices

From the matrix, we can write down the device relationship F(D,R) which in this case is, say, F({A,B},{sb}) = {BsbA}, we will often write "BsbA" or "B sb A" to denote "(B,sb,A)". Also, E= {{BsbA}}.

The following two rules are used in the mathematical model for simplifying the expressions:

- ***Joining rule:*** If there exists a device D1 which has relationship R1 with device D2 that is D1R1D2 and R2 with device D3 that is D1R1D3 and if R1=R2=R' where R1, R2, R' ∈ R and D1,D2,D3 ∈ D then we have D1 R' (D2, D3)

- ***Distributive rule:*** Suppose D1R1D2D3 and D1R2D2D4, and R1=R2=R' then from the joining rule we have D1R'(D2D3,D2D4) and therefore since D2 is common in (D2D3,D2D4) we write D1R'D2(D3,D4). Similarly if we had D1R'(D3D2,D4D2) we write D1R'(D3,D4)D2.

  Note that from the above distributive property D1R'D2(D3,D4) ≠ D1R'(D3,D4)D2 because the way devices are written in the equation tells us in which direction the relationship exists. This is shown in figure 3 below. The figure on the left represents D1R'(D3,D4) D2 and the figure on the right represents D1R'D2(D3,D4).

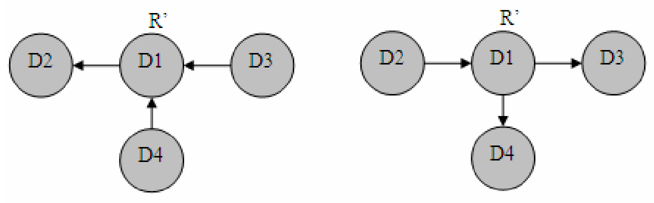


Figure 3: Different meanings for similar kinds of representations in mathematical model

3.2 Mathematical Model - A Case Study

Consider the following case study in figure 4 which is taken from the paper called “Challenge of Invisible Computing” written by G. Boriello [13]. The above concept of device blending is not restricted to the home environment it can be used anywhere. In this case our environment will be the (imaginary) boundary that surrounds the devices.

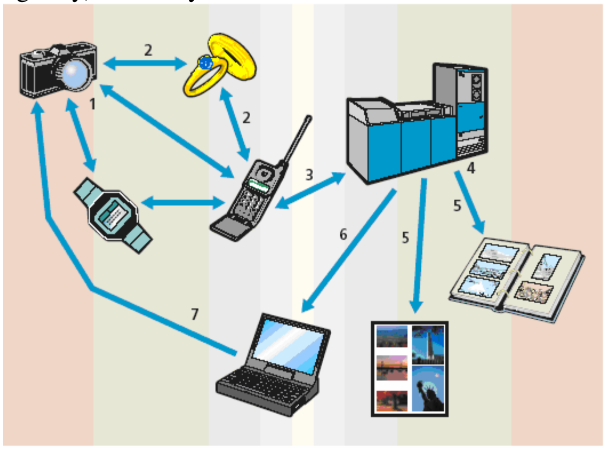


Figure 4: Digital photography in a world of invisible computing [13]

The arrows represent the direction in which the information or the data packets flow can be transferred. The following is the explanation taken from the Invisible Computing paper [13], “*After the user takes a picture, (1) the camera forwards the data to a body server (shown as a cell phone), or to an interim carrier (shown as a wristwatch) and then to the body server. Along the way, (2) the picture picks up the photographer’s identity from a personal item the user is wearing (such as ring). (3) The cellular connection lets the picture data travel into the information infrastructure (shown as a mainframe computer), where (4, 5) it uses computational resources to find the services (in this case, a photo album and printed pictures) to which the user is registered. Finally, (6) an acknowledgement that the services have received the data goes to the user’s PC, which (7) later informs the camera that it can reuse the space occupied by that photograph.*”
Based on the explanation given we have the following different relationships that exist between these devices.

Ring: ring provides either the camera or the body server with the user identification so that the photos can be transferred to the appropriate user account. If the camera is used by its owner than there is no need for user identification but if it is used by a different user than in the user identification is needed since the information will be transferred to a different user account. Hence, the ring **enhances** the camera and mobile phone by providing user identification which can be used as an extra step for data security.

Watch: it is considered as an interim carrier in case the body server is busy so that the camera can send the photos to it and then later on pass it to the body server. Here, the watch is acting as a **bridge** between the camera and the body server.

Mobile phone: Mobile phone acts as a **bridge** between the camera and the main computer, that is, there is no direct connection between the main computer and the camera but there exists an indirect connection via the mobile phone. It is used for transferring the data as well as the acknowledgement from the main computer.
Laptop: Laptop is used as a **bridge** between the main computer and the camera for letting the camera know the data has been received or not received.

Main Computer (Server): It is used as a **bridge** between the mobile phone, laptop, photo album and printer.

Figure 5 shows the above relationships.

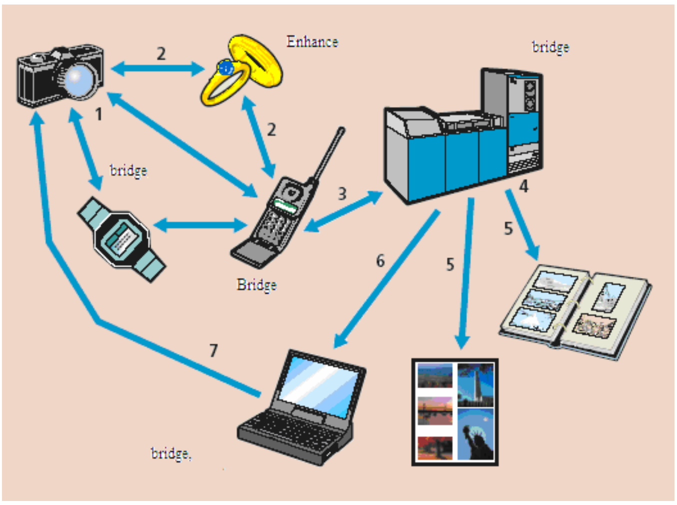


Figure 5: Relationships between different devices

Let R be ring, C be camera, M be mobile, S be the server, W be watch, L be laptop, A be the album and P be the printer.

From the above figure we have the following relationships, as expressed in our notation:
1: RehC
2: RehM
3: Mbr*CS
4: Wbr*CM
5: Mbr*WS
6: SbrML
7: SbrMA
8: SbrMP
9: LbrSC

We can use a matrix which represents the different relationships described above, as follows (in figure 6):

	R	C	M	S	W	L	P	A
R		eh	eh					
C								
M		br*		br*,br*	br*			
S			br,br,br			br	br	br
W		br*	br*					
L		br		br				
P								
A								

Figure 6: Matrix representation of devices and their relationship

From the matrix, we can now simplify the relationships by combining expressions, and so, we now have:

1: R eh(C, M)
2: M br(C, W) S
3: S br(ML, MA, MP)= S br M (L,A,P)
4: W br* CM

5: L br SC
From 1 and 2 we replace M in 1 by M in 2 so we have

6: R eh(C, M br*(C, W)S)

If we replace S and W in 6 with S and W in 3 and 4 respectively we have
7: R eh(C,M br*(C,(W br*CM))(Sbr M(L,A,P)))

And lastly replacing L in 7 with 5 we have the following equation

8: R eh(C,M br*(C,(W br*CM)) (Sbr M((L br SC),A,P)))
Therefore, environment E = {F1(D,R)} where F1(D,R) is effectively:
R eh(C,M br*(C,(W br*CM)) (Sbr M((L br SC),A,P)))
i.e. In this case, the set of relationships collapses into one complex relationship by following the Web of relationships (this is a *strongly blended environment*).
Then, given only one device ecology in E, E is also effectively
R eh(C,M br*(C,(W br*CM))(Sbr M((L brSC),A,P)))

From the above equation, we notice that all the devices in the environment have at least one relationship with another device and there is no device that does not have a relationship; hence, we can say that we have a **fully blended environment.** If we have a device which has no relationship with any other device in the environment, then we will have a **partially blended environment, i.e.**

Fully blended Environment: an environment is said to be a fully blended environment if there exists at least one blending relationship for each device in the environment.

Partially blended Environment: an environment is said to be a partially blended environment if there exists a device in the environment which does not have a blending relationship with other devices in the environment.

Uses of the mathematical model

The expressions formalize the relationships among devices (i.e., the device blending). There are at least four uses of above model:

- One of the basic uses of this model is as a basis to show the user how many devices in the environment are connected, how they are connected (since the relationships as captured in an IDR matrix, can be depicted in a graphical format) and what relationships exist between the devices.
- Another use of this model is to enable the user to tell the devices how to interact, i.e. the user devices an IDR matrix and then let's the system attempt to form the connections as prescribed in the matrix.
- This model also lets user know which devices are using more resources and which ones are not. These resources can be represented as the inter-device relationship. For example, let A, B be two devices such that device A substitutes device B based on service X. That means B is using A's resources and device B is also using the resources from other devices without providing its own resources to other devices. This information can be used by the user to reconfigure his/her devices so that they provide optimum resources to its friends.
- Companies that manufacture more than one device (or product) can use this model to design the interaction between all their devices (or products) when they are located in the same environment. Via device-blending, we, therefore, enable devices to "pile up" at a place, to form a more effective collection of blended devices as devices are added.

3.3 Device Blending and Tasking Architecture

The device blending architecture is divided into three layers each component in the layer has a clear specific responsibility as show in the Figure 7. These three layers provide the modularity between different processes in a device as well as extensibility to the model. The basic layer is the Blending layer that stores the information of devices which provide services to their "friends". The Task layer does the composition/decomposition of a task and stores information about user preferences.

The final layer is the device layer which is similar to the Seamless Device Integration layer in Interplay [7]. This layer provides the connectivity, discovery of devices; storage of content or aggregate contents and keeps track of device functionality.

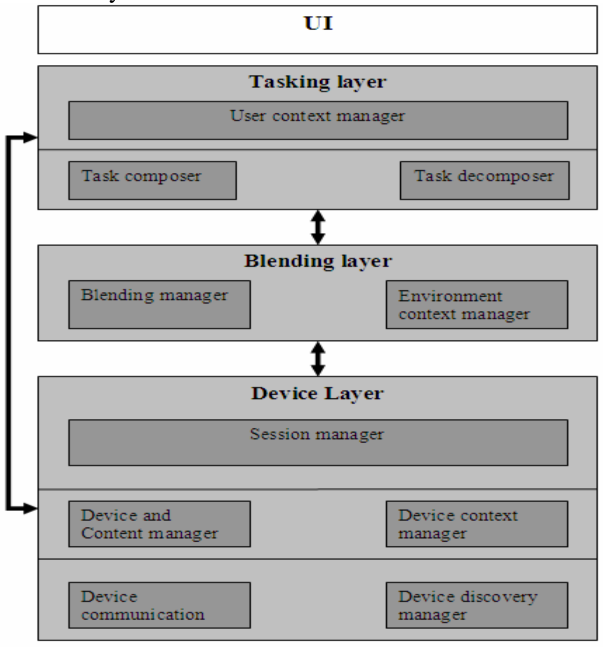
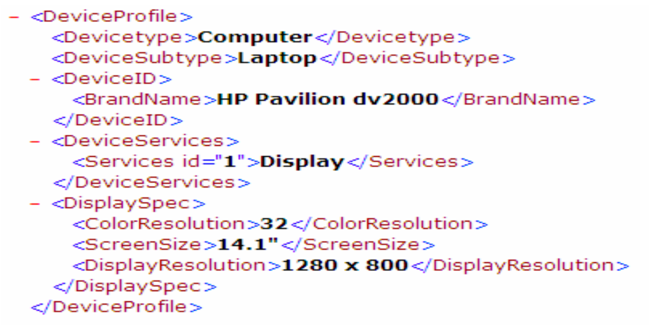


Figure 7: Device blending Architecture

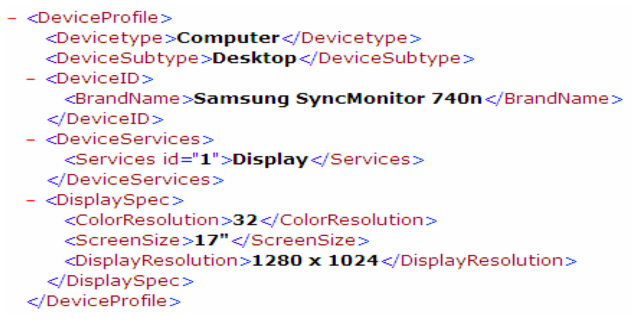
Blending Layer

- **Blending manager.** The blending layer manager's sole purpose is to store the information about devices which share a blending property (or relationship) with other devices. To perform this task, for each device, the blending manager collects the profile of the devices which are connected to the device and compares each device profile with this device's profile and assigns a blending property between (or among) them if their exist one. For example, consider the following device profile for a desktop PC and a PDA respectively.



```
- <DeviceProfile>
    <Devicetype>Computer</Devicetype>
    <DeviceSubtype>Laptop</DeviceSubtype>
  - <DeviceID>
      <BrandName>HP Pavilion dv2000</BrandName>
    </DeviceID>
  - <DeviceServices>
      <Services id="1">Display</Services>
    </DeviceServices>
  - <DisplaySpec>
      <ColorResolution>32</ColorResolution>
      <ScreenSize>14.1"</ScreenSize>
      <DisplayResolution>1280 x 800</DisplayResolution>
    </DisplaySpec>
  </DeviceProfile>
```

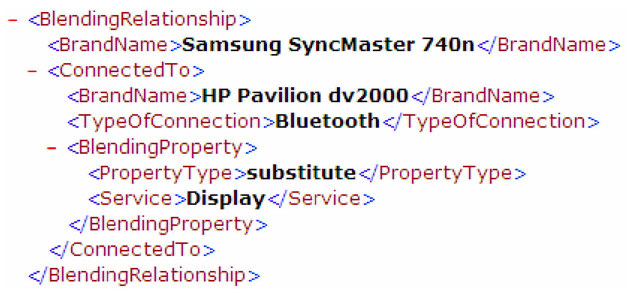
Figure 8: HP laptop device profile



```
- <DeviceProfile>
    <Devicetype>Computer</Devicetype>
    <DeviceSubtype>Desktop</DeviceSubtype>
  - <DeviceID>
      <BrandName>Samsung SyncMonitor 740n</BrandName>
    </DeviceID>
  - <DeviceServices>
      <Services id="1">Display</Services>
    </DeviceServices>
  - <DisplaySpec>
      <ColorResolution>32</ColorResolution>
      <ScreenSize>17"</ScreenSize>
      <DisplayResolution>1280 x 1024</DisplayResolution>
    </DisplaySpec>
  </DeviceProfile>
```

Figure 9: Samsung desktop device profile

The blending manager compares the devices' services in the profile of each of the devices and finds that there exists a common service called "Display", which implies that these devices can substitute each other based on the Display service. Figure 10 shows a sample XML file of how the blending manager will store this information.



```
- <BlendingRelationship>
    <BrandName>Samsung SyncMaster 740n</BrandName>
  - <ConnectedTo>
      <BrandName>HP Pavilion dv2000</BrandName>
      <TypeOfConnection>Bluetooth</TypeOfConnection>
    - <BlendingProperty>
        <PropertyType>substitute</PropertyType>
        <Service>Display</Service>
      </BlendingProperty>
    </ConnectedTo>
  </BlendingRelationship>
```

Figure 10: An XML file with the device blending relationships

Task Layer

- **User Context manager:** the sole purpose of the user context manager is to store the user preferences and when a need arises, use the user preference to complete the user task. Context manager reduces the user- device interaction. Consider a scenario where the user wants to print a document. The user doesn't need to go through all the process of selecting which printer s/he wants the document to be printed or how far the user needs to go to pick up the document (Location), but instead the task is sent over the network and the system figures out how to do it. If there is an ambiguity then the system refers to the user context manager to get the user preferences of that given task on order to complete the task, instead of going back to the user. For example, a document can be printed on two different printers, so the system can refer to the user context manager and see what the preference are. The preference could be stated as rules such as "If the document has a figure or a diagram then use the colour printer else user the printer which is free." Such default behaviours represented with rules reduces the need for user intervention but the user might still be called in as and when required and indicated by the rules.
- **Task composer:** The task composer is responsible for composing the task description in the system. This task is then advertised by the device so that devices which are connected to it can check if they can perform the whole task or part of the task. For example, a user types a document in a mobile phone and tells the system to save it in a server and print it as well. In this case the task is then passed through different devices (in a peer to peer approach via peers "friends") till it reaches the server. Then, the server checks that it can perform the saving task but not the printing. The server performs the saving task but passes the remaining task, that is, "printing the document" to its friend devices till it reaches a printer which can print the document. We envision that this is done automatically, so that, effectively, this is a constrained (by knowledge of about peers) peer-to-peer search for a device that can do part of or the whole task.
- **Task decomposer:** The task decomposer works with the task composer when a device receives the task, the task decomposer checks out the functionality requirements of the task with the functionality of the device. If it matches and the device is available to perform the task (the task decomposer checks with the session manager for the availability of the device) then the device performs the task else it passes on the task to its neighbour device.

Device Layer

- **Session manager:** the session manager keeps track of which devices are using what services of a device or which device is offering services to other devices so that if a new request comes via the task decomposer it checks the status of the services and based on that it will either accept the task or it will not.
- **Device and content manager:** the device and content manager manages the content in a device, groups the content based on certain criteria, keeps track of all the devices that are connected and acts as a temporary storage system for data that needs to be passed on to other devices in case those devices are busy.
- **Device context manager:** the device context manager stores the information of the environment so that based on the environment context some services provided by the device can be used or they can not. For example, a mobile phone goes on silent mode if it notices that there is a lot of noise in a room or it can increase the volume of the ring tone.

The benefits of our automatic device blending architecture can be viewed as follows:

- **Easy Human-Device Interactions.** The device blending architecture provides a simple and convenient way of providing an uninterrupted service to a user wherever he/she is at home. The user does not need to worry about how the devices are going to interact to complete a task; s/he just expects the task to be completed or a response if the task is not completed. For example, the user need not worry if his/her laptop is not connected to printer; as long as there exists an indirect network connection between the laptop and the printer the task should be completed. For example, figure 9 and 10 show two different connections that may exist between the laptop and the printer.

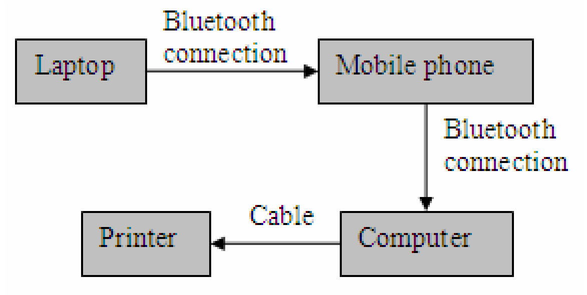


Figure 11: Mobile phone and Computer providing the indirect connection between Laptop and Printer

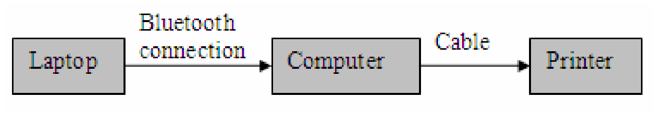


Figure 12: Computer providing the indirect connection between Laptop and Printer

- **De centralized system:** Since the system is de centralized if even one of the devices (in a given collection) is not working, other tasks could be completed which are not related to the non functioning device. With a centralized system, if the server or the central system stops working than the user can not perform any task.

This architecture not only helps the user but also helps companies that have more than one product that they manufacture. For example, a company like Sony has many products and as part of product design, they can make a relationship model for their products, that is, if these products come into an environment, the model specifies how they are going to interact with each other. In this case, products add value to one another i.e. an accumulation products in a place will result in a more powerful "user task processor"; adding products to a place will result in a new relationships being formed between the new products and the existing products at the place, yielding a more powerful multi-device task processor for the user. For example, a Sony mobile phone can be an Interim Carrier between Sony Camera and a Server. Here, the mobile phone acts as a bridge between Camera and Server.

3.4 Implementation and Software Infrastructural Issues

There are four implementation issues for the device blending architecture which we are currently working on:

- *Device driver*: Since the architecture is a distributed architecture, a criterion has to be defined about which device needs to store/download the device driver of a device. For example Harry bought a new smart clock [12] which has a Bluetooth connection. His computer also has a Bluetooth connection. In this case, do both the devices need to get the drivers or only the computer needs to have it? This issue could be solved by checking which device has resources to do it. For example, if the smart clock does not have a connection to the internet but computer does than there is no need for the clock to download the driver since it can be done by the computer
- *Device profile matching and device relationship generation:* Another issue that we are looking in to is how to determine from the device profile which services complement, enhance, substitute, enable and disable each other. Bridging two or more devices doesn't require the checking of the profiles of the devices since the bridging devices are there just to relay the information from one device to the other.
- *Device ambiguity*: For example, which printer to select, if there is more than one printer available to print a document? This problem could be solved by associating context or rules with the task for example you can specify "use only the colour printer".
- *User involvement:* The third issue leads to another issue, and that is when to use the user requirement to complete a task? Does the task need to check the user requirement before it is performed or to use it later when it runs out of options?

These issues are not the only considerations we are exploring; we are also looking into software infrastructures which can support this concept. The requirements here include:

- **Maintainability:** we imagine the home environment to be dynamic in the sense that some devices will be added or removed from the environment over time and therefore the relationships that have been established needs to be updated. This problem could be solved by using protocols such as UPnP, and Jini.
- **Usability:** not only that device can connect to other devices but how much information it needs to have so that a user can use the services provided by these devices easily.
- **Resource usage conflict resolution**. we are also looking into the issue of resource usage conflicts. For example, the TV in John's room is not working so he comes downstairs to watch his movie in the sitting room and Alice was watching her movie in her room and she comes down to cook and watch the remaining part of the movie but she finds John is watching the movie. So who should be given the preference to finish watching the movie? One manual solution is to let John and Alice talk it out, but could system based solutions solve easy conflicts automatically?

4. Conclusion

In this paper, we have described the motivation, a formalization and an architectural design of our device-blending model. The model allows the user to design his/her environment which consists of different devices and these devices can interact with other devices based on inter-device relationships. It also lets the user know which devices do not interact with other devices and which devices are using more or less resources (in the sense of being involved in many relationships and using many services or having many of its services being used by other devices).

The idea is that, given a collection of devices, through a process of what we call device-blending, relationships (of the types we specified) are automatically formed (via device profile matching as we noted) among the devices. Such inter-device relationships are represented via the expressions we introduced, and such expressions can be depicted graphically or via an IDR matrix (and shown to the user in such a format). These IDR expressions can also be used for formal, automated analysis – to discover important devices, for example, which forms bridges among smaller collections of devices. Alternatively, the user can specify such IDR expressions and have the system (i.e., the set of devices) attempt to form such relationships, or given a generated IDR expression, the user modifies it and submits that to the system to reconfigure the system.

Our proposed device blending architecture consists of three layers: task, blending and device layer. Such a layered architecture provides a separation of concerns – note that the architecture is distributed and peer-to-peer, with each peer having such layers (perhaps to different complexities according to their computational capabilities).

Our notion of blending is, hence, at a level of abstraction above low-level device connectivity and discovery, and after a collection of devices have formed relationships with one another (i.e. have blended with one another), they can be collectively tasked by users - the user is effectively issuing tasks to, or being served by, a collection of blended devices.

A novelty of the notion of device-blending is that such inter-device relationships are above the level of network connectivity, and provides a general abstract description of how devices might or can work together, independently of particular domain-specific tasks. Our previous work [15] was on a centralized workflow metaphor for coordinating devices; the work here takes a novel decentralized peer-to-peer approach instead. After relationships have been formed among the devices, i.e. after device-blending, the devices can then be tasked and work with one another to fulfil user tasks – the metaphor here considers human relationships where tasks are typically performed effectively over an existing web of human relationships. We are working on the task language for the user to specify such tasks over a collection of blended devices.

The above mentioned issues are part of our future work but also we will be looking into making our mathematical model more general and looking into capturing connectivity details. For example, the model does not explain what kind of connection exists between different devices, or record the criteria for assigning the relationship (i.e., remember the results of matching the device profiles).

We will also be looking into what is a feasible way of using this architecture in a device. If a separate chip with the architecture build in it is needed or whether installing device blending software is enough.

5. References

[1] M.D. Weiser. The Computer for the 21st Century. Scientific American, 265(3):66-75, September 1991

[2] UPnP, http://www.upnp.org/

[3] JINI, http://www.sun.com/software/jini/

[4] Bluetooth, http://www.bluetooth.com/

[5] Salutation, http://www.salutation.org/

[6] SLP, http://openslp.org/doc/html/IntroductionToSLP/index.html

[7] Messer, A. Song, H. Kumar, P. Phuong Nguyen Kunjithapatham, A. Sheshagiri, M. 2006. Interplay: a middleware for integration of devices, services and contents in the home networking environment. In *Proceedings of the 3rd IEEE international conference Consumer Communications and Networking Conference*. (CCNC 8-10 Jan. 2006). pp. 1083 – 1087

[8] Brumitt, B. and Shafer, S. 2001. Better Living Through Geometry. *Personal Ubiquitous Computing*. 5, 1 (Jan. 2001), pp. 42-45

[9] Kawaguchi, N., Cogma: A middleware for cooperative smart appliances for ad hoc environment. In Proceedings of the International Conference on Mobile Computing and Ubiquitous Networking (ICMU2004), pp.146–151, 2004.

[10] Kohtake, N., Rekimoto, J., and Anzai, Y. 1999. InfoStick: An Interaction Device for Inter-Appliance Computing. In *Proceedings of the 1st international Symposium on Handheld and Ubiquitous Computing* (Karlsruhe, Germany, September 27 - 29, 1999). pp. 246-258.

[11] Iwasaki, Y., Kawaguchi, N., Inagaki, Y. 2003. Touch-and-connect: a connection request framework for ad-hoc networks and the pervasive computing environment. In *Proceedings of 1st IEEE International conference on Pervasive Computing and Communications*. (PerCom 23-26 March 2003). pp. 20 - 29

[12] Smart Personal Object Technology (SPOT), DotNet Developer's Journal, http://dndj.sys-con.com/read/46614.htm

[13] G. Borriello, "The Challenges to Invisible Computing," *IEEE Computer, Integrated Engineering column*, vol. 33, no. 11, pp. 123-125, Nov. 2000.

[14] Huang, A. C., Ling, B. C., and Ponnekanti, S. 1999. Pervasive computing: what is it good for?. In *Proceedings of the 1st ACM international Workshop on Data Engineering For Wireless and Mobile Access* (Seattle, Washington, United States, August 20 - 20, 1999). pp. 84-91.

[15] Loke, S.W. Service-Oriented Device Ecology Workflows. Proceedings of the International Conference on Service-Oriented Computing, (eds.) M. Orlowska, S. Weerawarana, M.P. Papazoglou, and J. Yang, Trento, Italy, December 2003, Springer-Verlag, Lecture Notes in Computer Science 2910, pp. 559-574.