A Self Configurable Topology-Aware Network for Smart Materials

Tadashi Yanagihara§, Hiroshi Sakakibara‡, Ryo Ohsawa‡, Masao Ideuchi†,
Naohiko Kohtake†, Iwai Masayuki†, Kazunori Takashio†, Hideyuki Tokuda†‡

†Graduate School of Media and Governance
‡Faculty of Environmental Information
§Keio Research Institute
Keio University, Shonan Fujisawa Campus
Endou 5322 Fujisawa, Kanagawa 252-8520, JAPAN
{willow, skk, ryo, ide, nao, tailor, kaz, hxt}@ht.sfc.keio.ac.jp

Abstract

In this paper, we present an architecture to create extemporaneous networking on u-Texture. u-Texture is a tile shaped smart material which can be used to construct Smart Furnitures[13]. Each u-Texture reacts according to what the entire Smart Furniture is shaped like. This reaction greatly affects the networking within a u-Texture. We therefore must allow dynamic arrangements within the network when a u-Texture is detached from or attached to another Smart Furniture. We present our tactics to implement such an self configurable topology-aware network to support applications executed on the entire Smart Furniture.

1 Introduction

Within these recent years, we have witnessed enormous advances in technologies for sensors and wireless communication. As a result of this rapid technological progress, we have succeeded in promising a post PC era, where computers are embedded into small devices and made invisible to the user. This sort of computing is known today as *"Ubiquitous Computing"*[14].

The basic idea behind the invisible computer is making the computer adjust to the user's activities. This is the exact opposite of today's computer paradigm, where the user is forced to adjust towards the services provided on the computer. Making the computer invisible would at the same time mean the computer is more aware of the real world than every before. We can expect the computer to keep track of the user and their belongings to provide automated services.

Various efforts from different projects have been made to realize the invisible computer. One popular approach is amplifing everyday household appliances or furnitures with computation power. For example, "Roomware"[12] has created DynaWall, CommChair, and InteracTable where each furniture is equipped with sensors and touch panel displays. This approach enables collaborative work between multiple users and provides automatic recognition of the user's belongings. The main goal of this project is providing a concurrent workflow for groupwork.

We have extended the idea of amplifying legacy furniture by creating "u-Texture"[4][7], a self-organizable versatile square-shaped material for multipurpose usage. u-Texture is equipped with multiple sensors, actuators, and computation power and is used to extemporaneously create Smart Furniture. It is also distinctive because it is equipped with wired and wireless network interface and makes full use of them to connect and transfer data between u-Textures.

This one-for-all approach can benefit more in comparison with previous research in 3 ways: Allowing the user to create more customized furniture, reusing the same material for different furnitures which is less expensive and space conservative, and providing a more concurrent operation.

2 What is u-Texture?

u-Texture is a square shaped board equipped with an interactive touch panel display, built-in speaker, an RFID reader, an acceleration sensor, infrared sensors, R232C interfaces, and wired and wireless network interfaces. Each u-Texture can directly connect to another u-Texture to form a more complex system (called "Smart Furniture") by help of a multi-joint which can connect up to 4 u-Textures. The infrared sensors and R232C interfaces are used to automatically detect the incoming u-Texture, and the acceleration sensor is used to measure its own gradient. The display and

speakers are used for actuation, and network interfaces are for communication.

Figure 1 shows u-Texture being used as 4 types of Smart Furniture: Smart Table, Smart Shelf, Smart Stand, and Smart Wall. Each furniture share the same properties: They can display, sense, computate, and communicate with the user, other objects and other u-Textures.

Figure 1. u-Texture can form various Smart Furnitures

For example, the Smart Wall provides the ability to show high resolution images across multiple u-Textures, and a simple whiteboard application to emulate a real one. The Smart Shelf provides a smart surface to detect objects layed upon it, and launch the appropriate application for it. For example, if someone were to lay a CD case on it, a CD Player application would be launched. The Smart Table can be used for both occasions (as a smart surface and a mega display), therefore is able to use both applications. In this sense, u-Texture can form into different shapes free of restrictions by detaching a u-Texture and attaching it to another. And of course, u-Texture can also be used simply alone, with just the same features as the examples mentioned above.

3 Issues in u-Texture Networking

Before moving on, we would like to present our goals set for u-Texture. We believe these goals can share common visions with more networked applications and appliances which are to be found in future ubiquitous environments. The goals are solving the following three issues: Dynamic Routing, Network Addressing and Naming, and Wired/Wireless Interface Selection.

3.1 Dynamic Routing

u-Texture includes 4 wired RJ45 connectors on each side to connect to another u-Texture, and a 802.11b/g network interfaces to provide service roaming between txterior networks. Data is transfered in a multihop manner, jumping from u-Texture to u-Texture to reach its final destination. Therefore, we would need someway of routing data around the entire u-Texture network.

It can be said a large part of ubiquitous networking will be based on existing internet protocols, such as the widely known Internet Protocol (IP). This is because the current ubiquitous computing environment is based on technology deprived from this area, and as well, must offer seamless access to traditional internet services.

However, this happens to come into question when applied with u-Texture. This is because there are at least 5 network interfaces within a u-Texture, which would mean the application programmer would have to keep in mind which network interface to bind their service with. Also, traditional routing protocols such as RIP[3] or OSPF[5] require routing information beforehand. This would interfere with our goal to create a zero configuration environment, and thus require an autonomous network.

On the otherway around, we can eliminate the need for Layer 3 routing and stick with Layer 2 networking by the use of bridge. The discussion divides where we should take scalability versus simplicity. We will discuss more on this with results from our experiments later on.

3.2 Network Addressing and Naming

There are many visions towards how the future network would look like, but all visions have one thing in common; It will be more distributed. Our vision is not an exception.

As we have mentioned before, u-Texture can be used alone or attached to another u-Texture to form various shapes. This means a u-Texture must operate independently from other u-Texture. However, we find this a problem because we cannot rely on common methods such as DHCP or DNS to construct a distributed network ID and naming addressing.

Also, because IP was built with no intends towards being “topology-aware”, there is no telling on how far apart two nodes are from each other. In a u-Texture, this sort of information is essential because each u-Texture must recognize the entire Smart Furniture’s shape to determine which furniture it is shaping.

However, it would be nonsense to fiddle with IP in this case; As we have mentioned before, most technology in ubiquitous computing is inherited from the internet. Any drastic changes to this sort of technology would break the compatibility with such applications.

3.3 Multiple Interface Selection

This is a topic which had its time in discussion within network QoS, but we believe worthwhile to revisit at this certain era. We have found a specific network topology which we would like to refer to as "Crowd-type Network". This is an network topology which is where many network interfaces, especially wireless network interfaces, are highly concentrated and deliver different amounts and variety of data. We mention this as a "Crowd", because it is difficult to achieve a steady and fast throughput in this situation. Anyone who has worked with 802.11 cards knows the performance in throughtput not only drops drastically, but also the jitter between the maximum/mininmum throughput and packet loss becomes a major problem.

4 Solutions

While developing u-Texture, we have discovered problems which we believe many other ubiquitous systems will face in the near future. We would like to share theses problems and possible ways to evade in throughout this paper.

4.1 Simplified Wired Mesh Routing

In our early version of u-Texture, we have commited heavy testing to see if we could rely on just wireless network connectivity alone. As we will mention later on in the paper, we soon found that the throughput would drop drastically as we add more and more u-Textures to the Smart Furniture, so drastic that only a small percentage of the data could even be sent.

We have introduced wired network connectivity for a steady and fast transmission. Considering routing between these wired network interfaces, we need to create a multihop network where packets are forwarded from u-Texture to u-Texture until the data reaches its final destination. This is to create a wired mesh between u-Texture to create a single hard wired network topology.

In our current version of u-Texture, we have binded all 4 of the wired network interfaces as a bridge and creating a psuedo network device to bind the IP address to. This way, the network is more simpler and requires no configuration. We would like to note that we rely on a loop prevention protocol to prohibits loops in the network which would result in a packet storm. However, we have referred to loop prevention protocols such as STP to prevent this. Although STP does take time to settle when enabled, it still is effective enought to keep excessive traffic down.

Figure 2 shows how data is transfered from u-Texture to u-Texture. The data in this case is divided and sent separately to different u-Textures (The example is a multiwindow video application). The data can be sent in a multihop manner and forwarded via wired connection or directly via wireless connection toward its destination. We current use the wired network for intranetworking (for routing traffic within the private network of u-Texture), and the wireless network for roaming or sending data for internetworking (in opposite as in sending data to hosts in global networks).

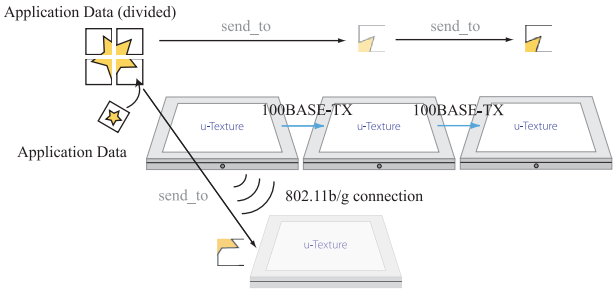

Figure 2. Figure of Routing within u-Texture

4.2 Automatic Topology-Aware Addressing and Naming

Considering Automated ID Addressing and Naming, we must fulfill three different requirements while keeping in mind that we cannot refer to any previous centralized mechanisms. The three requirements are:

- Automatic Allocation of Network ID Addresses
- Network Notification
- Topology-Aware Naming

We would like to address these issue respectively.

Automatic Allocation of Network ID Addresses

We have currently adapted AutoIP[15] to automatically assign an ID to the bridge psuedo interface mentioned earlier. AutoIP depends on the broadcast address to avoid collison, but since a u-Texture is one whole bridge network, this will not become of any problem. On the other hand, AutoIP is known not to scale: It can handle up to only 169.254.0.0/16 (or 255) IP addresses. We believe it is rare to have more than 10 u-Texture in one structure, and that AutoIP is enough to get the job done. Even if there were any such situations, we would be facing more problems than network: power consumption, CPU overheating and the durability of the casting holding the structure up are to name a few.

Network Nofification

Once the network has been set ready for broadcasting, the next step is to notify the u-Texture's ID to others. Each

u-Texture sends an XML file describing itself and its neighbors to the broadcast address. Figure 3 shows a example of the XML file. This represents 1 of 2 u-Textures lined up vertically to form a Smart Wall.

This XML files consists of information such as the u-Texture's network ID, its neighbors and where they are located, and its gradient. In the upper part of our example, we find this u-Texture's network ID is 192.168.1.9, another u-Texture is located on its left, and that its gradient is vertical. Next, we find information on the neighboring u-Texture; Its network ID is 192.16.1.5, it has a neighbor to its right (which is the original u-Texture we just mentioned), and lined vertical.

These files are collected and parsed, then stored locally on each u-Texture. This way, the u-Texure can figure out how many u-Textures are within the entire network, and which u-Textures is next to which. Afterwards, this information is used when transmitting data to another u-Texture.

```
<utexture>
  <externalization class="jp.ac.keio.utexture.NextEvent"
   time="1109022525719" nodeid="192.168.1.9">

   <member name="ownID" type="string" value="192.168.1.9"/>
   <member name="ownEdge" type="string" value="LEFT"/>
   <member name="ownAngle" type="string" value="ACCEL_DOWN"/>

   <member name="nextIDs" type="string[]">
    <element>192.168.1.5</element>
    <element>NULL</element>
    <element>NULL</element>
   </member>
   <member name="nextEdges" type="string[]">
    <element>RIGHT</element>
    <element>NULL</element>
    <element>NULL</element>
   </member>
   <member name="nextAngles" type="string[]">
    <element>ACCEL_DOWN</element>
    <element>NULL</element>
    <element>NULL</element>
   </member>

  </externalization>
</utexture>
```

Figure 3. Network Information used in u-Texture

Topology-Aware Naming

As we have mentioned earlier, traditional naming schemes do not support real world location or relation (such as "A is next to B"). To solve this problem, we have developed a naming scheme u-Texture Structual Markup Language (USML), which is based on the "Chemical Markup Language"[10] (CML) for shape recognition based on vectors. USML is based on vectors, where each molecule is expressed as third dimension coordinates (which would be written as *x, y, z*) called lattice points.

Although expressing USML with third dimension vectors can work pretty well with u-Texture, it has its limits. For example, we cannot tell the difference between a u-Texture facing the ceiling than to the floor using this scheme. We have therefore added a 4th vector to the lattice points. This 4th vector can possess 1 of 3 values: a, b, c. It can also, if necessary, possess a positive or negative figure, depending if the u-Texture's foreside (In our current version, it would be the side with the display) were to be used reverse.

Figure 4 shows the USML scheme applied to the Smart Shelf. This shelf would be expressed: (0,0,0,C), (0,0,0,A), (1,0,0,C), (0,0,1,C), (0,0,1,A), (0,0,2,A). Each u-Texture can calculate all the coordinates in the Smart Furniture and determine the entire Smart Furniture's shape. The u-Texture then figures which furniture (also programmed as USML lattice points beforehand by programmers) it is capable of. The user can manually select or let the system automatically choose which furniture out of the nominations. All u-Textures react to the selection by loading applications, activating sensors, and reconfiguring the network.

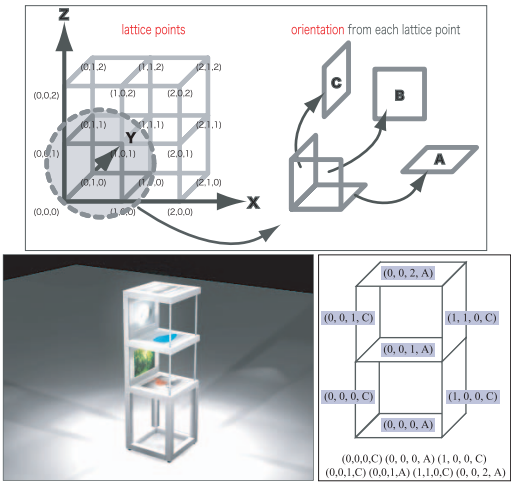

Figure 4. Naming Architecture in u-Texture

4.3 Dynamic Network Interface Switching

Since each u-Texture is equipped with one wireless network interface, one can expect a lot of radio interference within the area. This is one characteristic which completely differs from existing wireless networks(Ex. sensor networks). We use the network interfaces to transfer large amounts of continuous data. Such data includes sensitive applications such as video streaming, where packet delay would be more noticable to the user. To make things worse, these applications also make heavy use of multicast packets, which are also one thing wireless devices are terrible at processing.

In this case, it is better to switch over to a more reliable network, such as a wired interface. In u-Texture, we intend

to use this mechanism to switch network interfaces when somepart or whole Smart Furniture's architecture changes. For example, if a u-Texture were to be detached and used alone, we would want to switch our network interface to the wireless interface, and back to wired when we attach to another u-Texture.

There are ways to solve this problem, but the main discussion here is actually who is responsible to decide which interface to use. It may seem the programmer should be aware of which interface to use in their applications. However, we doubt any programmer can estimate what the hardware configuration is on the platform the application is to be run on. Having the system choose which interface best suits the case seems to work also, but how would we notify the system on when to switch interfaces?

In our current implementation, we initiate all network interfaces when we attach or detach a u-Texture. When we reconfigure the Smart Furniture to take shape of another furniture, the system loads the applications for the specific furniture, deactivates/reactivates any sensors which are needed for the application, and initializes the network interfaces to renew its coordinates.

During this session, the system disables any unused wired network interfaces and recreates the bridge and psuedo interface. The wireless interface instead is normarly left intact, but if the application wishes so, it can be disabled. Therefore, we can select wired or wireless connectivity, according to the applications needs.

5 Evaluation and Related Work

We used the actual implementation u-Texture Version 2 to evaluate our scheme. We have conducted tests with the embedded network interfaces of u-Texture using netperf[9]. As shown in Figure 5, we have found the wireless 802.11g interface becoming practically useless for application use after 3 u-Textures had their network interfaces enabled. Otherwise, turning all but 1 wireless network interface in the Smart Furniture recovered its throughput back to when only 2 u-Textures were communicating. In turn, the wired 100BASE-TX interfaces has shown up some promising results which show practical usage for high bandwidth consuming data; The throughput dropped by only some 1-2 Mbps as each u-Texture was added.

Out of all data transmissions, multicast packets could barely be transmitted over wireless. This is obvious and very well known to anyone who has worked with wireless networks, but we would like to intentionally restate this because it will become a serious problem if we were to deploy a multicast-dependent mechanism or protocol. (An example would be mDNS, which is proposed in Rendezvous)

Considering related work, we cannot at this moment find any work which have faced with similar problems confronted in this paper, in terms of providing reconfigurable networking material. Considering furniture with software and network support, iLAND[11], which is an environment using Roomware, claims on the use of wireless RF-based network to create collaboration between Roomwares. This network is very simple, but will fail to have multiple roomwares collaborate with one another due to the problems we have mentioned earlier.

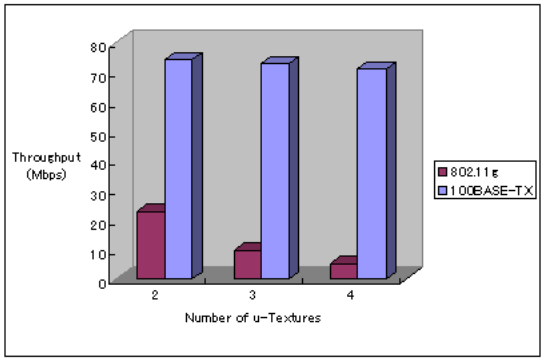

Figure 5. Evaluation of Throughput on u-Textures

As we have mentioned earlier on in the paper, this system is practically one huge bridge to take full advantage of broadcasting. Although this works fine with our current scale, we would need to introduce layer 3 routing to cut down on use of broadcasting for more high demand applications. We could borrow methods from ad hoc networks, because they seem to face similar problems such as distributed routing, naming, and leader election within wireless networking. In fact, the zeroconf technology we currently use has emerged from the ad hoc network community.

We have mentioned earlier that current dynamic routing protocols for the internet (RIP, OSPF) cannot fit our needs. In turn, the usage of DSR[6] or AODV[8], ad hoc network protocols used to locate pathes for routing in dynamic networks, could suit our requirements instead. To solve our problem with wired interface election, the methods described in Directed Diffusion[2] can possible be used to elect the cluster head and provide data aggregation. In u-Texture's case, this would be chosing the most closest wireless interface to the remote u-Texture. We can also benefit load balancing and efficient energy usage by using this method. Such funtions are to be found in future implementations of the middleware executed on all u-Textures.

Considering naming architecture, we could result to a more complex naming system which would be executed on each u-Texture. For example, Balakrishnan[1] has proposed a layered naming architecture for usage on the internet. Their approach is similar to ours in consisting three lay-

ers: service level identifiers, endpoint identifiers, and user level identifiers. Both systems share in the idea of having absolute and relative pointers to hosts or data. However, at this moment, we would like to focus on other more simple ways to "wrap" the lattice points to Network ID (In this case, IP addresses).

6 Conclusion and Future Work

In this paper, we have presented an architecture to create extemporaneous networking for smart material. Our smart material, u-Texture, dynamically creates networks when arranged to form Smart Furniture. Challenges within our architecture are dynamic routing, automated network addressing and topology-aware naming, and multiple interface selection. We have confronted these problems by provides simplified wired mesh routing, automatic topology-aware addressing and routing, and dynamic network interface switching. Finally, our evaluation shows our architecture's benefits to providing a more flexible naming architecture and steady throughtput towards the development of ubiquitous applications.

For future work, we are trying to suppress network broadcasting and network roaming.

To be more specific, we could suppress the usage of broadcast packets and use unicast based application level multicasting instead. This would not only save up on bandwidth, but also allow applications to efficiently transmit synchronous data across multiple u-Textures.

Providing true concurrent work flow is also a remaining critical issue. We must be able to forward sessions between multiple network interfaces without losing performance. We currently initiate all session currently running on each interface when attaching or detaching a u-Texture. There are many approaches to solve this problem, and we wish to look into them more closely.

7 Acknowledgement

This research was conducted as part of the Ubila Project supported by the Japanese Ministry of Affairs and Communications. We would like to thank the uService team for their support, and Uchida Yoko Corporated (An office appliance manufacturer) for their collaborative in constructing u-Texture. More detail on u-Texture can be found at http://www.ubi-lab.org/u-texture/.

References

[1] H. Balakrishnan, K. Lakshminarayanan, S. Ratnasamy, S. Shenker, I. Stoica, and M. Walfish. A Layered Naming Architecture for the Internet. In *ACM SIGCOMM 2004*, Sep 2004.

[2] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next Century Challenges: Scalable Coordination in Sensor Networks. In *ACM MobiCom 1999*, Aug 1999.

[3] Gary S. Malkin. RIP Version 2. RFC 2453.

[4] M. Ideuchi, D. Maruyama, T. Murakami, N. Kohtake, J. Nakazawa, K. Takashio, and H. Tokuda. u-Texture: A Self-Organizable Material for Building Smart Furniture. In *The 6th International Conference on Ubiquitous Computing (UBICOMP2004) Poster Session*, Sep 2004.

[5] John Moy. OSPF Version 2. RFC 2178.

[6] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. *Mobile Computing*, 1996.

[7] N. Kohtake, T. Yanagihara, T. Murakami, M. Ideuchi, D. Maruyama, K. Koizumi, T. Yonezawa, H. Sakakibara, Y. Matsukura, J. Nakazawa, K. Takashio, and H. Tokuda. u-Texture: A Board-shaped Smart Material that Provides Ubiquitous Services. In *The 2nd International Symposium on Ubiquitous Computing Systems (UCS2004) Demo Session*, Nov 2004.

[8] C. E. Perkins and E. M. Royer. Ad hoc On-Demand Distance Vector Routing. In *The 2nd IEEE Workshop on Mobile Computing Systems and Applications*, Feb 1999.

[9] Rick Jones. The Netperf Performance Resources. http://www.netperf.org/.

[10] H. Rzepa, P. Murray-Rust, and B. Whitaker. The Internet as a Chemical Information Tool. *Chem. Soc. Revs*, 1(10), 1997.

[11] N. Streitz, J. GeiBler, T. Prante, S. Konomi, C. Muller-Tomfelde, W. Reischl, P. Rexroth, P. Tandler, and R. Steinmetz. i-LAND: An Interactive Landscape for Creativity and Innovation. In *The ACM Conference on Human Factors in Computing Systems (CHI'99)*, May 1999.

[12] N. Streitz, T. Prante, C. Muller-Tomfelde, P. Tandler, and C. Magerkurth. Roomware - The Second Generation. In *The 4th International Conference on Ubiquitous Computing (UBICOMP2002) Video Session*, Sep 2002.

[13] H. Tokuda, K. Takashio, J. Nakazawa, K. Matsumiya, M. Ito, and M. Saito. SF2: Smart Furniture for Creating Ubiquitous Applications. In *The International Workshop on Cyberspace Technologies and Societies (IWCTS2004)*, Jan 2004.

[14] M. Weiser. Some Computer Science Issues In Ubiquitous Computing. *Communications of the ACM*, 36:75–84, Jul 1993.

[15] Zero Configuration Networking Working Group. Zero Configuration Networking. http://www.zeroconf.org/.