Networked Surfaces: A New Concept in Mobile Networking

James Scott* Frank Hoffmann* Mike Addlesee† Glenford Mapp† Andy Hopper*†

*Laboratory for Communications Engineering, Cambridge University Engineering Department, Cambridge CB2 1PZ, UK

†AT&T Laboratories Cambridge, 24a Trumpington St, Cambridge CB2 1QA, UK

October 24, 2001

Abstract

Networked Surfaces are surfaces which provide networking to specially augmented objects when these objects are physically placed on top of the surface. When an object (e.g. a notebook computer) connects, a handshaking protocol assigns functions such as data or power transmission to the various conducting paths that are established.

This paper describes the position occupied by this concept in the world of networking, presents an overview of the technology used in its realisation, describes the current prototype implementation, and outlines the implications in the fields of Ubiquitous and Sentient Computing.

Keywords: Mobile Networking, Ubiquitous Computing, Sentient Computing

1 Introduction

The Networked Surfaces Project at the Laboratory for Communications Engineering (LCE) was started in October 1998. The initial vision was of a floor-based network for wearable computer users, but the focus quickly shifted towards applications involving desks. As of June 2001, a prototype Surface[1] has been built, the controlling software has been written, and tests are underway to explore the limits of the technology.

[1] To save space, the word "Surface" (capitalised) is used as an abbreviation of "Networked Surface". Lowercase "surface" is used to describe the physical surface portion of a Surface system.

This paper compares Networked Surfaces to other networking technologies, outlines the technologies used in their realisation, describes the current prototype implementation, and discusses implications of their use.

This section describes the Networked Surface concept, its position in the networking world, the advantages it exhibits over other networks, and an example application, the "Networked Desk."

1.1 The Networked Surface concept

A surface (such as a desk or floor) is described as a *Networked Surface* if a suitably augmented object can acquire *connectivity* to data and/or power infrastructure, simply by being in *physical contact* with that surface.

Connectivity is achieved by providing a number of electrically independent paths, or *links*, between the surface and object, which are allocated to *functions* such as data transmission or power. Different objects may require different functions, and the functions which are available to objects may differ between Surfaces.

Physical contact in this case means that the object may occupy any position and orientation on the Surface. This flexibility is achieved using a special layout of conducting *pads* on the Surface and the object. When the object touches the Surface, pairs of surface and object pads provide communication channels.

While current research focuses on direct electrical conduction between the pads, other means of connection such as capacitive coupling and induction may be explored in future.

1.2 Supporting Mobile Networking

To make computer systems usable by users on the move, some form of mobile networking is required. Traditionally, the user would carry computing equipment with them and connect to wired or wireless networking when they require communications facilities. Alternatively, mobile users can be supported by "virtual networking".

Virtual networking can free users from carrying mobile devices, if computing hardware becomes so prevalent that a user will find adequate computing and communications facilities wherever they might go. Technologies such as VNC [18] allow users to access their working environments irrespective of location. However, such a massive prevalence of computing facilities is unlikely in the foreseeable future, so this possibility is

not explored further in this paper.

Wired networking requires a physical medium for network traffic, which must be attached to each node. Bandwidth may be plentiful, as the entire capacity of the medium can be dedicated to each device, using high-speed switches. Also, being wired for networking means that it is easy to provide power to a device, as it is already connected to cabling.

However, the need to be physically attached to the network leads to a lack of mobility. One solution to this problem is the use of disconnection-friendly file systems [13], which employ caching and other techniques to facilitate useful work whilst in a disconnected (and therefore mobile) state. However, this is not a complete solution, as such systems can only offer "weak" consistency.

Wireless networking stands at the other end of the spectrum. Bandwidth is inherently shared inside "cells," which are delimited by the range of the transmitted signals. Mobility is of course "free" within a cell, but there is the problem of handover if the mobile devices cross cell boundaries. The main drawback with wireless systems is fulfilling their power requirements; their batteries must be charged from time to time, causing a loss of mobility during these periods.

1.3 Characteristics of Networked Surfaces

Networked Surfaces attempt to combine the best qualities of both wired and wireless networking for indoor environments. On one hand mobility is effectively provided, because the use, transport, and connection of wiring is not required. On the other hand, wired-style connectivity and bandwidth are present, and power may be provided to charge batteries.

Networked Surfaces allow objects to be networked in a simple yet powerful manner. The network is flexible in that many different types of objects may be used on the same Surface, and convenient in that objects are easily attached to and detached from the network. They are upgradeable, as new types of data busses may be added without replacing the physical surface. The bandwidth available is also easily increased by adding more data busses to the Surface.

The use of such a generic interface as a surface lends itself to interoperability, but without the possibility of misconnection. This is in contrast to wired networks, where the user is either forced to carry many types

of cabling (if different cables are used for different functions, such as RS-232 data and Ethernet data), or, if a single generic cable type is used, the user may misconnect the system causing lack of function at best and damage to hardware at worst. On the Surface, the object "asks" for the function it requires electronically, thus creating a situation where no wiring is required, and where the user may remain ignorant of minutiae such as the networking protocols being used.

Providing power over the Surface requires the use of current protection hardware, as shorts are inevitable in a situation where a spilt drink can temporarily connect any "wire" on the network to any other. However, such provision frees the user from worrying about battery levels; if an object is habitually stored or used on a Networked Surface, there will rarely be insufficient charge for operation during periods of mobility.

1.4 Target Environments for Networked Surfaces

In order to explore the scope of usefulness of Networked Surfaces, devices are grouped into five categories, namely fixed incommunicado devices, fixed networked devices, mobile devices, wearable devices, and devices for outdoor use. The applicability of Networked Surfaces for each category is examined below.

Fixed incommunicado devices include clocks and other normally isolated appliances. The Networked Surface can provide power for them, allowing those which use batteries to never need replacements, and those which use mains power to be free of connecting wires. Some use can also be found for networking such devices with low bandwidths, for example a clock could also show up-to-date news or share prices. Alternatives to the Networked Surface in this arena include power-line networking [19], and radio-based networks designed for embedded use, such as AT&T Labs' "pen" (formerly "piconet") [3], and Bluetooth [4]. While power-line networking obviously provides power as well as networking, it would only work for mains-powered systems. The radio-based systems, on the other hand, do not provide power facilities.

Fixed networked devices such as computers, peripherals and telephones can use Networked Surfaces for networking and power, instead of using traditional wired networks such as Ethernet. The disadvantage for Networked Surfaces lies in the expense, which will always be more than a simple wire. However, the Surface provides more scope for mobility than a wire, which tethers an object to a certain space. As will be discussed later, the Surface can also provide useful facilities such as the auto-configuration of networks.

Surfaces are also useful for mobile devices which need to communicate (e.g. PDA's/notebooks) or to recharge their batteries (e.g. mobile phone). The disadvantage is that the Surface does not give full 3D mobility, unlike a radio-based system such as WaveLAN [12]. However, the power-providing capabilities of the Surface give it an added quality that radio networks cannot match.

For devices that are "wearable" [15], it may actually be an inconvenience to take the device off and place it on the Surface, for recharging or communications, so a solution involving radio or Personal Area Networks (PAN) [24] may be more suitable.

For devices used outdoors, the Networked Surface is impractical for networking due to the difficulty of networking a useful portion of the environment, and the problems inherent in exposing conductive pads to rain, etc. The Networked Surface is therefore considered an indoor technology.

1.5 The Networked Desk

One appealing use of Networked Surfaces is as a "Networked Desk". In today's computer-enabled workspaces, there is an abundance of wire on and around desks, causing clutter and making rearrangement of the working environment difficult. By providing networking and power through the desk itself, most of these wires could be eliminated.

Such a desk could simultaneously provide computer-peripheral connections (e.g. replacing USB wiring), computer-computer connections (e.g. replacing ethernet wiring), power connections, and even telephone connections. Note that it is only the wiring that is replaced; the protocols themselves can still be used when appropriate, so existing hardware need not be extensively modified to become Networked Surface capable.

For power, it is of course infeasible to supply mains voltages due to the likelihood of shorts. However, many devices do not require mains-level voltages; a separate transformer is often used to reduce this to, say, 12V DC. The output of such transformers may be safely carried over a Networked Surface, with current limiting hardware to safely detect and handle shorts.

Prior research into enhancing desks has focussed in their use for computer input/output as opposed to networking [16, 17]. While the Networked Surface is aimed primarily at communications, it may also be used to provide sensory facilities, such as the knowledge of the location and orientation of the objects on the

Surface, to enhance these applications. This is discussed in later sections.

The main disadvantage of networking a desk is cost; such a desk will never be as cheap as a plank of wood and some network and power cabling. While cost analysis is not the focus of this research, it is noted that cost may be reduced by only networking a small "smart patch" of a desk, or by confining the use of Networked Surfaces to those places where it is important to have mobile, convenient, uncluttered computing environments; one such example is a conference table.

A photo of the prototype Surface providing network connectivity to a PC is shown in Figure 1.

Figure 1: A notebook PC on the prototype Networked Surface (inset back of notebook)

2 Technology

This section explains the overall architecture of the Networked Surface design and discusses the technologies used, from the physical surface itself through to data transport concerns.

2.1 Overall Architecture and Nomenclature

A Networked Surface has many hundreds or thousands of electrical *pads*, only some of which will be in use at any one time. This poses a huge control problem — in order to detect new devices, each pad has to be monitored closely. Using a single entity to handle this would limit scalability; therefore a distributed approach is taken.

The Surface architecture is illustrated in Figure 2, and is structured as follows. Surface pads are grouped into *tiles*, each of which has a *tile controller*. These are responsible for handshaking new connections by communicating through the surface pads with *object controllers* on the object. Both types of controller are only active during the handshaking phase; after an object is patched through to its required *functions*, the controllers can no longer participate. This is a deliberate design decision; the controllers are not provided with hardware necessary to interpret the connected functions. This allows the same tile design to be used on Surfaces with different sets of available functions.

The functions, which may be data- or power-related, are provided as wired busses on the surface, and terminate at the *surface manager*, which is responsible for driving the power lines and bus-mastering the data lines. The surface manager communicates with all connected objects on the Surface, and bridging data between the Surface and the Internet at large. The surface manager also drives the *tile control bus*, which is used to facilitate the *handshaking protocol* discussed below.

On the object side, the *object manager* provides the object with access to the data busses and/or power, and exchanges control traffic with its surface counterpart. In reality, the object pad controller and object manager may be implemented as a single unit (since there is no scalability problem for an object).

2.2 Topology

In order for an object to establish a network connection, a number of independent electrical conducting paths, or *links*, are required. The layout of pads on the surface and object required to achieve this is known as the *topology*. Exactly how many links are required depends on the type of network, and whether the provision of power is desired. The simplest connection types require two links, namely ground and either half-duplex data or power. More complicated connections may have six or more links, for large data busses,

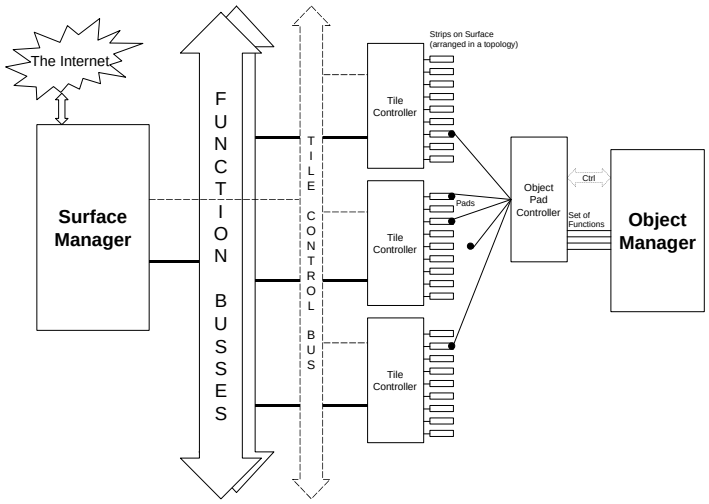

Figure 2: Networked Surface Architecture

or in situations where more than one data bus is required.

The topology must guarantee that enough links will be present for a connection to be made, for any position and orientation of the object on the surface. Furthermore, the topology should support both simple (two link) and more complicated (six or more link) objects, without having to change the surface pad layout.

2.2.1 Design

There are many possible choices for the topology, including hexagonal, grid-like, and brickwork-like arrangements of surface pads.

The chosen topology is one where rectangular strips cover the surface, and a small number of circular pads, themselves arranged in a circle, form the object's footprint. The gaps, or "margins," between the strips are chosen to be strictly bigger than the size of the object pads.

This topology, illustrated in Figure 3, has the following properties:

- The object pad size guarantees that object pads will never span two strips. Each strip can therefore perform a different function for the object, since there will be no short circuit created.

- By increasing the number of object pads around the circle, guarantees can be made that at least one

object pad will be in contact with each strip under the object, within a particular "column" of strips. This means that no strip will be wasted.

- Due to the previous two properties, the footprint required of an object is always bounded, since the object need only span as many strips as the number of functions it requires, and no more.
- This is true of many different numbers of functions required; for any given surface of this style and any given number of functions, it is possible to produce an object footprint that will guarantee connection, so long as the surface strips are wide enough (the topology relies on the fact that at most two "columns" of strips are reachable by an object at any one time).
- It is geometrically simple and therefore easy to manufacture.

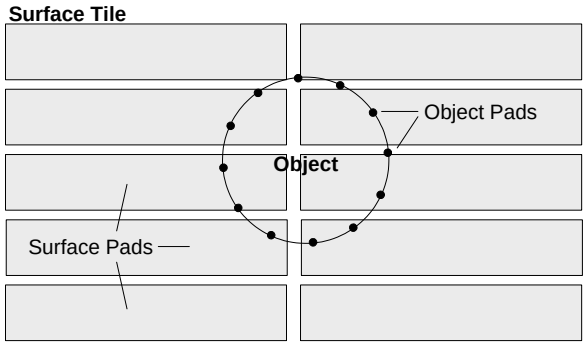

Figure 3: Example layout for the chosen topology

2.2.2 Results

In order to get actual workable dimensions for the chosen topology, a simulation was used to find the minimum size of object footprint and minimum number of object pads on that footprint, for various numbers of functions required. These tests were done for a surface with the characteristics outlined below:

- Width of strip (including margin): 2.08 cm. This is chosen for ease of manufacture (24 strips per 50cm tile).

- Length of strip (including margin): 12.5 cm This is again chosen for ease of manufacture, and is irrelevant to the results below, so long as an object never touches pads from more than two "columns."

- Size of margin: 0.3 cm. This limits the size of the object pads.

The specifications above do not cause loss of generality for the results below, as the topology can scale up or down to different surface pad sizes.

Links Required	Object Pads Required	Diameter of Object (cm)
2	5	2.64
3	9	4.62
4	12	6.78
5	16	8.82
6	19	10.80

Table 1: Minimal object specifications for the Surface described above

As Table 1 shows, this topology can support objects with between two and six links. The number of pads required is approximately three times the number of links, and the object diameters can be accommodated on the bottom surface of objects such as PDA's or notebook PC's. In summary, this topology is feasible to implement.

2.3 Object Connection and Disconnection

New objects are discovered and connected using a *handshaking protocol*, illustrated in Figure 4. When disconnection occurs, the pads involved must return to a state ready for the next handshake.

2.3.1 The Handshaking Protocol

The handshaking protocol operates upon the pads which are currently unconnected; such a pad is said to be in "handshaking" mode. When a connection is established, the pads it uses are said to be in "connected" mode, and do not participate in further handshaking.

One difficulty with handshaking is that the object is not commonly grounded with the surface, so simple transmission on one pad at a time will not work. To achieve a common ground, a strategy of *grounding by*

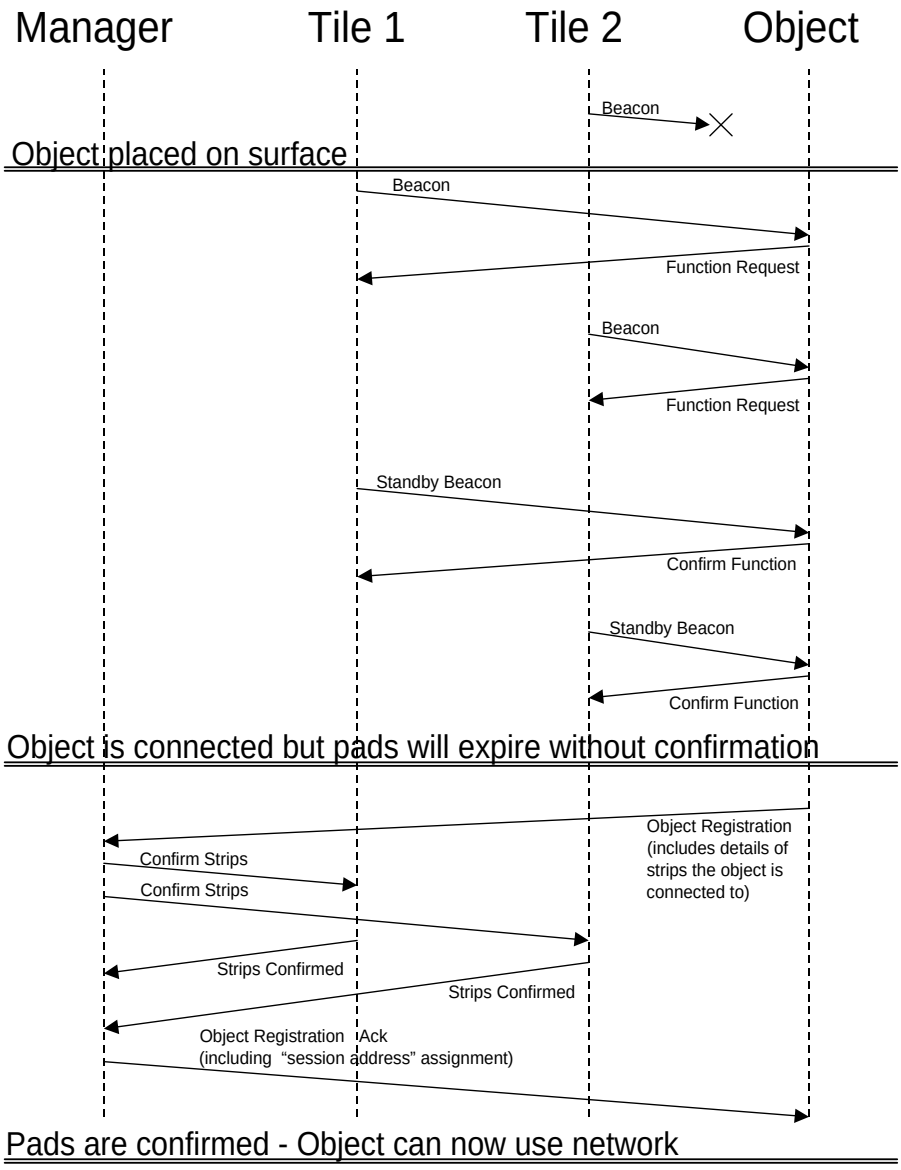

Figure 4: Handshaking Protocol for an object with two pads on different tiles

consensus is used, whereby the object takes the average voltage level of its pads, using a resistor network, to be the ground value. This is illustrated in Figure 5.

Using this grounding scheme, the links are discovered as follows. Tile controllers periodically send beacon messages along each surface pad, while holding other ("handshaking" mode) pads around that pad to ground. The object controller uses comparison against the consensus ground to detect this beacon on one of its pads, and replies with a "function request." After a further exchange of identification data, both controllers put their respective pads into a "standby" mode.

When that surface pad is next due for a handshaking beacon, the tile controller instead sends a "standby beacon," incorporating data so that the object can ascertain that the link still "belongs" to it. The object can use this mode to request further information from the tile controller, or to wait for enough links to become available for a connection, before finally transmitting a "connect" message to each pad.

One of the connected links will be the function "ground," thereby commonly grounding the surface and object so that future communication need not use *grounding by consensus*.

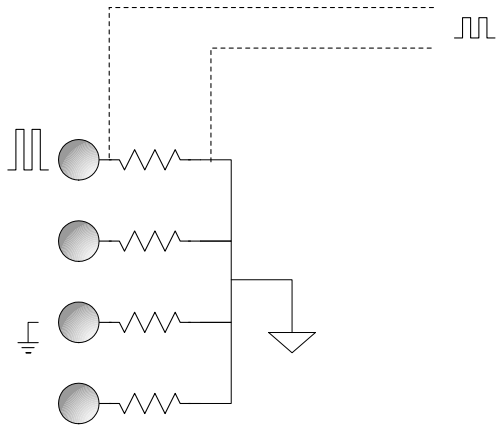

Figure 5: Resistor network used for "grounding by consensus" effect

2.3.2 Disconnection

Once the "connect" message is issued, the pad controllers have no means of communicating with one another; the connection is handed off to the surface and object managers, which must therefore perform the task of disconnection detection. When disconnection is detected, the managers cause the relevant pads to be re-

inserted into the handshaking protocol by sending control messages to their respective pad controllers.

However, in order to identify the correct surface pads, the surface manager must know the mapping between objects and the surface pads they are using. This is achieved by having the object collate this information during handshaking using the "standby" mode to make appropriate queries, and by mandating the object manager to send said information to the surface manager immediately on connection.

Furthermore, it is noted that an incomplete handshake, or a malicious object, could result in the pad data transmission never taking place, and therefore result in pads being "stuck" in a connected mode, but actually doing no useful work. This is solved by causing the tile controllers to set time-outs on connected pads, bringing them automatically back into the handshaking system, unless the surface manager explicitly instructs them not to, which the surface manager would do only on receiving the pad information from the object. Thus, the handshaking protocol is made robust.

There remains the problem of what constitutes a "disconnection," and how this is detected. Disconnection is defined as one or more of the pairs of previously connected pads losing connectivity, for example if the object were either picked up or moved.

Disconnection detection can take place using one of two methods. Firstly, a hardware-based method may be able to tell if, for example, an object has stopped drawing current along a power line. Secondly, objects may transmit periodic "heartbeat" packets on data busses, failing which the surface may conclude that the object has disconnected. Similarly, an object may disconnect its pads if it has not heard the surface's "heartbeat" in a particular period. Note that erroneous disconnection is not too costly, since a new connection will be immediately re-established.

2.4 Surface Busses

After handshaking is complete, a network connection is set up between the surface and object managers.

Up to this point, the Surface is generic — any sort of network may be handshaked in this fashion, as the Surface described so far simply acts as a wire, albeit with slightly different electrical properties.

The only restriction on the networks that may be used is that they must be multi-drop, since many devices may simultaneously connect, either on the same tile or on different tiles. If a non-multi-drop bus

such as RS-232 were used, the Surface would have to provide some guarantee that only one connection could be made to an interface at a time. Alternatively, the object controller could be enhanced to bridge between the RS-232 connection and a Surface multi-drop bus.

The use of a multi-drop architecture for data seems to sacrifice one of the features of wired-style networking, that available bandwidth can be provided in a dedicated fashion to many devices simultaneously, and not be shared amongst many objects. However, this property is instead achieved by employing multiple busses on the surface.

The provision of more than one data bus gives rise to many interesting properties making use of asymmetries between the busses. For instance, different busses may span different areas of the surface, perhaps with more busses on high-usage areas; this would result in partitioning of bandwidth by location. Another option is to have different bus flavours running simultaneously; this could be used to give more powerful devices higher bandwidths using a fast bus, while at the same time supporting simpler devices with a slower bus.

A further interesting possibility is the use of busses to enforce Quality of Service (QoS) guarantees on the Surface, by allocating devices to busses according to bandwidth requirement and priority. Devices could connect to a default bus on coming into contact with the Surface for the first time, and be instructed to re-connect to another bus with restricted access, if their QoS requirements warrant. Policing these busses is particularly easy on the Networked Surface - misbehaving objects can be physically disconnected from the bus, a more powerful technique than dropping packets, which wastes bus bandwidth.

Finally, it is possible for an object to connect to two or more busses simultaneously, given a large enough physical footprint. This may be useful if the object is multi-purpose and as such has two or more independent networking requirements. An object such as a notebook computer may also inherently have many bus interfaces, and may wish for more than one to be connected at a time, although this is again limited by footprint size. The object may be able to configure its object controller to ask for different busses at different times, for example, if an unattached keyboard is present next to a notebook computer, the notebook may ask to be connected to the bus that keyboard is on.

2.5 Data Transport

While simpler devices on the Surface may have their own bespoke transport-level protocols, such as the Palm Pilot's RS-232 "Hot-Sync," more complicated devices will usually make use of a TCP/IP stack in the object. The Surface therefore needs to provide support for TCP/IP.

However, the Networked Surface is peculiar in that disconnection and reconnection can occur often. As such, unmodified TCP/IP may well be a bad choice, due to its inefficiency in conditions of high packet loss, which it assumes to be due to congestion [14]. On a Surface, congestion is highly unlikely, as there is no routing in broadcast situations such as multi-drop busses, and there are therefore no "bottleneck routers" at which congestion can occur.

To make TCP/IP disconnection-aware, there are several strategies that could be employed, the most obvious being to re-write the TCP code to include a new "disconnected" state, in which the transfer is effectively frozen until re-connection moves the TCP session back into another state. However, this requires re-implementation of TCP on every object that the Surface would like to support, which is not ideal.

A different approach, and the one taken in this project, is to make the Surface link layer "smart." This is achieved by buffering one or more recent TCP packets for each TCP connection, and when re-connection occurs, immediately re-sending those packets on TCP's behalf, in order to "kick-start" the connection. In lieu of this impetus, TCP simply waits for a timeout, which due to its exponential backoff techniques may not occur until many seconds after reconnection.

2.6 Power

The provision of power across Networked Surfaces is one of the most compelling reasons for their use, in that it removes the ever-present need to plug in mobile devices from time to time. However, such provision also immediately brings up many human safety issues. While human protection is paramount, the networking circuitry in the surface and objects is actually much more sensitive, therefore a surface which protects this circuitry would intrinsically be safe for use by humans.

As previously discussed, power can be in many cases be provided at a low DC voltage level, since many portable devices only require these kinds of voltages. Current requirements for these devices can further be

reduced by noticing that time-to-recharge is much less of an issue, since networking the surfaces they are placed on would result in being "plugged in" a higher percentage of the time. However, even at low voltages, shorts can damage circuitry, unless current-limited.

To prevent this, configurable current-limiting and short-detecting hardware can be used on all pads, both surface and object. Such hardware immediately disconnects and disables a pad on detection of a current higher than reasonable for the appropriate link, and re-enables the pad when the short is removed, allowing the connection to be re-established.

3 Implementation

This section outlines the prototype implementation of a Networked Surface, and presents some preliminary measurements into the performance of the prototype.

3.1 Prototype Hardware

The prototype Networked Surface includes physical implementations of the tile and object controllers and the surface and object managers referred to in Figure 2. A photograph of the prototype is shown in Figure 6.

The prototype tile and object controllers use PIC microprocessors to execute the handshaking and tile control protocols, which interface with FPGAs controlling the switching of the functions assigned to pads on the surface. Both these technologies lend themselves to quick prototyping and ease of reprogramming, facilitating rapid development of the system.

The object manager function can be fulfilled by the object controller for simple objects, for example those requiring only RS-232 data connectivity, or objects just requiring power. For more complex objects such as notebook computers, the object itself can fulfill the role of object manager.

The surface manager role is performed by a software driver in a Linux PC with a custom PCI card, which includes FIFO buffers for incoming and outgoing data on all busses, implemented in an FPGA. Again, this solution is chosen to offer maximum flexibility.

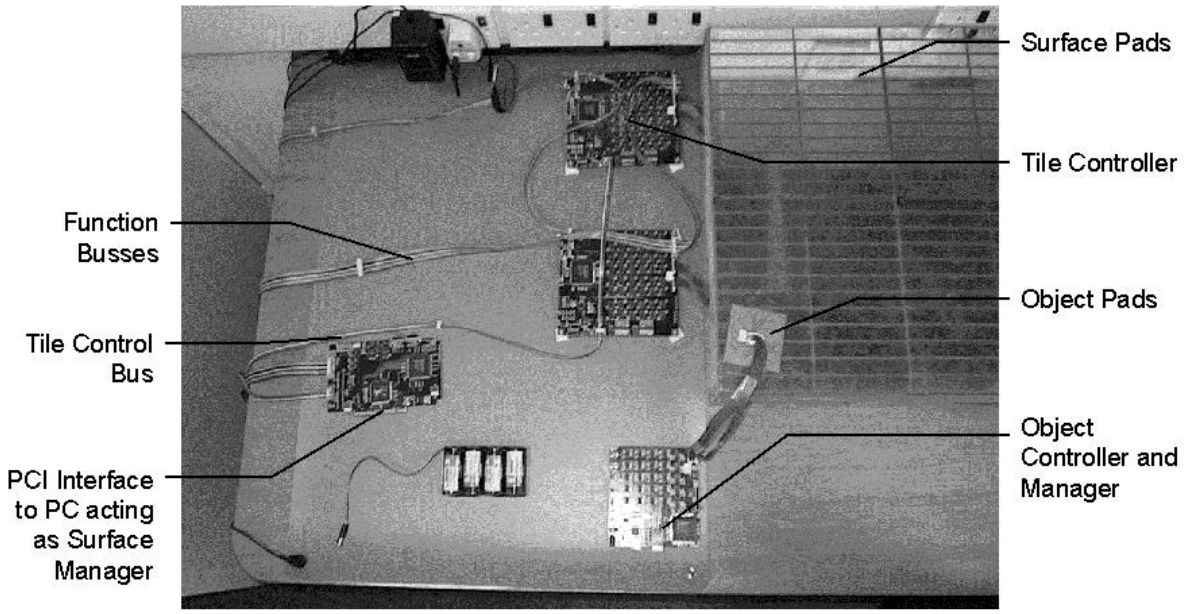

Figure 6: Photograph of prototype Networked Surface Components

3.2 Prototype Networks

The prototype Surface supports two bus types, namely I^2C [11], and a custom high-speed multi-drop bus based on the Bus LVDS (BLVDS) [5] transmission scheme.

The I^2C bus is inherently multi-drop, using wired-AND logic, with arbitration, addressing, and flow control. It was chosen due to its simplicity and the fact that many embedded systems have built-in I^2C capability. This bus is used to support simple devices with low bandwidth requirements; the first object created was a simple I^2C-enabled LCD. It can also be used to tunnel RS-232 or other low-rate connections, using the object controller as a bridge, so that RS-232-based devices can use the Surface, despite the fact that RS-232 is not a multi-drop bus.

The BLVDS busses use low voltage differential signalling, which results in lower sensitivity to noise and cross-talk, and is practicable despite the non-ideal electrical characteristics of the Surface. Since BLVDS only defines a modulation scheme, it allows a custom and adaptable link-layer to be put into place, which facilitates research into the Surface properties, for example the bus speed may easily be changed.

On top of BLVDS, a simple 8-to-9 coder is used to provide framing and clock synchronisation. This is

achieved by simply adding a ninth bit to each byte of data, which is the opposite of the eighth bit (guaranteeing transitions). Special symbols for "start packet" and "stop packet" are implemented as particular 9-bit sequences in which the eighth and ninth bits are identical. This coding was chosen because it is very simple to implement in an FPGA.

The next layer in the protocol is a "token star" link layer, to provide addressing and arbitration facilities; this is implemented in the Linux device driver. As the name implies, the surface manager grants each of the objects on that bus the right to send data in turn (the "token"), and that object either sends data (up to some maximum per grant), or sends the token back to the manager. This protocol inherently allows the manager to poll each object periodically, facilitating disconnection detection.

The BLVDS-based link layer is used on the prototype Surface for transmitting IP packets.

3.3 Preliminary Results

Tests on the characteristics and limits of the prototype Networked Surface are currently underway. This section discusses the metrics which are being measured, and gives preliminary results for the prototype Surface.

3.3.1 Bandwidth

The maximum bandwidth available to devices on the Surface is of prime importance, so that the Surface is comparable to modern high speed wired and wireless networks. The bandwidth is limited by two factors, firstly by the physical characteristics of the Surface medium, and secondly by limits of the prototype hardware. The electrical conducting path includes a wide strip of metal on the surface, the contact point of the surface and object pads, and two analog multiplexers.

Initial tests on the prototype system show that the LVDS channel can send error-free IP packets at speeds up to 2.8Mbit/s. However, at this point, limits which are inherent to the prototype system are reached. The prototype is currently undergoing further development to eliminate these bottlenecks.

Examination of packets at the physical layer shows that the rise and fall times of BLVDS signalling are approximately 75ns, which theoretically makes 13Mbit/s possible. This is the expected bandwidth limit of

the current prototype.

However, this still does not represent the final bandwidth limit of the Networked Surfaces medium. As previously mentioned, the channel characteristics are caused not only by the physical surface interface, but also the analog multiplexers used for switching. The multiplexers used impose a 300Ω series resistance; with such a high resistance (as compared to a wire), it is likely that the use of alternative switching circuitry will increase the bandwidth available. This remains work for the future.

3.3.2 Connection and Disconnection Timing

The time taken for an object to establish a connection determines many aspects of the usability of the Surface. With a short connection time, the user of an object on the Surface need not worry about accidentally moving the object slightly. With a long connection time, the penalty for slight movement may be irritating for the user. The prototype Surface can connect an object in under half a second, which is short in terms of human perception.

The disconnection detection period depends on more factors, such as the number of objects on a particular bus, but a worst case estimate based on the token-star link layer discussed above results in disconnection detection in under a tenth of a second (i.e. almost immediately in terms of user perception).

3.3.3 Number of Objects Supported

Another parameter is how closely the objects on the Surface can be placed. Whereas the Surface can accommodate a single object at any position and orientation, as soon as one object is using pads on the Surface, this creates one or more "dead zones" where a second object would not acquire enough pads to make a full connection. This leads to a maximum "density" of objects on the Surface.

For the prototype Surface, there can be nearly one hundred objects connected on each square metre of surface, if objects requiring four functions are used.

3.3.4 Power

The maximum power that can be provided, while protecting surface and object circuitry, will determine the usefulness of the Surface for powering objects. This applies to both seldom-moved objects, for which power cabling could be removed, and mobile objects, for which the need to plug them in to recharge their batteries could be eliminated.

However, the current prototype does not include any special power-provision circuitry, or short-protection circuitry, and hence is limited to providing power using the networking analogue multiplexers, which limits the current to 30mA at 5V. The prototype was designed for research into networking; experiments into the provision of power remain as work for the future.

4 Applications in Ubiquitous and Sentient Computing

The computer has for some time now been proliferating away from office desks, and finding its way into every imaginable device with potential for automation. This process began with embedded computers, such as the ones in office and home appliances, and has branched out into mobile and wearable devices which interact more directly with the user, such as MP3 players, mobile phones, and PDA's.

This movement has been studied in the fields of Ubiquitous Computing [1, 23], and Sentient Computing [10]. The former deals with the spread of computing into devices not previously computer-enhanced, with implications in functionality of these devices, and in the interaction between them. The latter is concerned with computers knowledge of their real-world environments, including such parameters as physical location.

The Networked Surface is intrinsically in line with the goals of Ubiquitous Computing; functionality is being added to commonplace surfaces such as desks. The Networked Surface provides three useful features in this arena, namely, transparent networking for a spectrum of device types, the provision of location information, and user interaction possibilities. These features are explored below.

4.1 Ubiquitous Networking using Touch

The Networked Surface uses an interface very familiar to humans, namely "touch". This is in line with the goals of Ubiquitous Computing; it facilitates a move away from interaction using mice and keyboards with monolithic computers, and towards the embedding of computers in everyday environments such as desks and floors, using simple intuitive interfaces.

As mentioned before, this "touch" interface frees users from having to know about cabling, bus types, power requirements, and other details. The object, being aware of its capabilities and requirements, can choose from a "menu" of functions that a particular Surface can provide, all completely transparently to the user.

4.2 Interacting with other Ubiquitous Systems

However, Ubiquitous Computing is not just about enabling individual devices, it is also concerned with the useful interactions between devices. This is particularly relevant to the Networked Surface, which is primarily a device for interaction.

The Networked Surface can provide other Ubiquitous Computing systems with convenient and user-transparent networking. Such applications include the MediaCup [8] by Gellersen *et al.*, and work on reactive videoconferencing environments by Cooperstock *et al.* [6]. The Surface can also add to these applications by providing location information.

4.3 Location Information

When objects connect to a Surface, information is gathered about the mapping between surface and object pads in order to make the connection. This information can also be used to infer the location and orientation of objects on the Surface.

4.3.1 Granularity of Location

The current prototype has not had location functionality added to it, but this is planned for the near future and involves the design and implementation of a software algorithm, with no additional hardware required.

However, by examining the topology used, one can estimate that the location will be to an accuracy of about 10cm, which is the width of a surface pad. The orientation granularity is estimated to be within about 45 degrees, which is an upper bound for the maximum angle an object may rotate whilst maintaining connection using the same pads.

Other indoor location technologies include the Active Badge [21] and Active Bat [22] systems developed at AT&T Labs, and the Locust Swarm system by Starner *et al.* [20]. The Networked Surface is set apart from these by the fact that it is primarily a network, with the ability to provide location information as a useful "side effect." More comparable research has been done by Bahl and Padmanabhan [2] on the use of signal strength data to locate WaveLAN-based objects; while this system shares networking capabilities with Networked Surfaces, it is only capable of room-grain location.

Another location system, also developed at the LCE and AT&T, is the TRIP system [7], which identifies and locates objects using cameras; the objects being tagged with circular barcodes. The properties of this system are that the tags are very low-cost, and can therefore be attached to anything and everything. However, the system relies on the foreknowledge of the location and orientation of the camera, as well as a network to the camera. For these reasons, it is conceivable that Networked Surfaces and TRIP can complement each other, the former providing networking and camera location services for the latter, and the latter locating devices which the Surface on its own cannot.

4.3.2 Auto-Configuration using Location

One application of the location information provided is internal to the Surface; location can be used to suggest a network configuration. When a "dumb" device such as a keyboard is placed on a Surface, the current prototype is designed to represent this device as being connected to a "virtual" port, however, the problem of which computer to attach this virtual device to has not been addressed.

With location information, such a device could be made to communicate with the device that is closest to it, both in terms of proximity and orientation. This leads to a myriad of applications. For example, if a keyboard were picked up and put down closer to a different monitor, then input from that keyboard could be re-directed to the computer associated with the second monitor.

This functionality also applies to smarter devices; for example a digital camera placed next to the components of a computer or notebook could cause a popup window on that terminal asking if downloading of pictures should commence. This strategy is also applicable to the case of a two-way synchronisation with PDA's.

Another application might be that devices with limited I/O capability could be made to export their interfaces, when they are placed next to devices with greater capabilities. This is applicable to notebook PC's when placed next to full-size keyboards and monitors, creating a "virtual docking station".

The use of real-world placement to indicate associations is intuitive, and from the user's point of view it would appear that devices have acquired extra useful functionality. This is consistent with Sentient Computing aims, and makes devices seem context-aware without actually requiring internal modification of those devices.

4.4 Ubiquitous Interfaces

The previous section has highlighted ways in which Surfaces can be used as human-computer interfaces, effectively using location information as a user input. This notion is expanded below.

The Networked Surface was conceived as a technology for computer-computer interaction. However, since such a surface already requires much augmentation to function, it is sensible to consider further augmentation which may, at little additional complexity, open doors towards new functionality, particularly in the realm of human-computer interaction.

As previously mentioned, desks have been the focus of much user interface research [16]. The Networked Surface can also support user interfaces, if augmented with input and output hardware.

The applications that could be made possible by augmenting the Surface in this way are abundant. However, as this remains speculation, and is not supported by the current prototype, only one application is presented here. This application is used to introduce some of the possible inputs and outputs that a Surface could support.

The application chosen is that of configuring data transfers between devices. The motivation behind this is that as Ubiquitous Computing goals are realised, more devices will become able to interact, posing the

problem of how a user indicates which interactions are desirable. The Surface can be used as a solution to this problem, as shown below.

4.4.1 User Input

Previously, object placement was presented as a possible user input. This concept can be expanded to include object movement, allowing some objects to become pointing devices. A user could employ such a pointing device to indicate relationships between objects on the Surface. In the example application, a user could "draw" lines between objects to indicate that data transfer should occur between those objects.

The actual pointing device could take a number of forms. A standard computer mouse could be augmented to allow pointing in the real world as well as the virtual world. Alternatively, a new type of device could be constructed, perhaps incorporating features particularly relevant to real-world pointing. Finally, pressure sensors may be incorporated into the Surface [9]. This would allow users to employ their own hands or fingers as pointing devices.

4.4.2 Output to the User

One can also imagine a Surface with an array of LED's in addition to the pads; such a surface could perform output tasks, making the Surface a true interface. This could be used in the example application in a number of ways. First, the objects being selected by the user could be circled by lit LED's, and a line "drawn" between them. This would aid the user in the choosing the correct objects.

Second, one might imagine that after a line is "drawn," a user confirmation would be required before the data transfer actually occurs. Instead of using a computer interface to make this confirmation, the Surface itself could be used to present a simple yes-or-no query, and accept the input. This might take the form of large icons such as ticks and crosses, which the user could then select from.

5 Conclusions

Networked Surfaces are a new network medium, sharing qualities of both wired and wireless technologies, and providing networking and power to objects placed on top of them.

The prototype implementation constructed has been shown to be capable of data rates of 2.8Mbit/s, and theoretically up to 13Mbit/s after further tuning.

The Surface has also provides location and orientation information about the objects on top of it. This facilitates Ubiquitous and Sentient Computing systems, and leads to the potential for human-computer interaction with the Surface.

6 Acknowledgements

The authors would like to thank everyone at the LCE and many people at AT&T Laboratories Cambridge for helpful discussions and comments, especially Leo Patanapongpibul of the LCE, and Ant Rowstron formerly of the LCE.

James Scott's PhD research is funded by the Schiff Foundation of Cambridge, and by AT&T Laboratories Cambridge. Frank Hoffmann's PhD research is funded by AT&T Laboratories Cambridge.

7 Biographies

7.1 James Scott

James Scott received his BA in Computer Science from the University of Cambridge in 1998. He has since been a member of the LCE, working towards a PhD as part of the Networked Surfaces project.

His research interests span the link, network, and transport layers, with particular interest in dynamic and adaptive networks. He is also interested in sentient and ubiquitous computing. He is a member of the ACM and the IEEE.

7.2 Frank Hoffmann

Frank Hoffmann received the Dipl.-Ing. (FH) degree in Technical Computer Science from the University for Applied Sciences in Hamburg, Germany in 1996. He spent 18 months working in the User Systems Ergonomic Research (U.S.E.R.) group at the IBM Almaden Research Lab in San Jose, CA USA, working

on capacitive sensors and pointing devices.

In 1998 he joined the Laboratory for Communications Engineering at Cambridge University as a PhD. student. His research interests include embedded systems, sentient computing and communications.

7.3 Glenford Mapp

Glenford Mapp has a BSc from the University of the West Indies, a MEng from Carleton University, Ottawa, Canada and a PhD from the University of Cambridge.

His research interests are home networking, Internet architectures, transport protocols and esoteric networks.

7.4 Mike Addlesee

Michael D. Addlesee (C.Eng. MIEE MIEEE) received the B.Sc. (Hons) degree in Physics from the University of Nottingham in 1981, the M.Sc. degree in Communications Engineering from Imperial College London in 1986 and the Ph.D. degree in Electrical Engineering from the University of Cambridge in 1991. In 1985 whilst working for Marconi Communication Systems, he was awarded a GEC Fellowship for a years study at the University of London. His Ph.D. research on Image Compression using the Subband Technique was, in part, funded by Thorn-EMI Central Research Laboratories, the IEE and the Fellowship of Engineering.

His research interests are in digital signal processing, image compression, genetic algorithms, portable multimedia computer networks and sentient computing.

7.5 Andy Hopper

Andy Hopper is the Professor of Communications at the University of Cambridge and a Fellow of Corpus Christi College. He has been Managing Director of AT&T Laboratories Cambridge Ltd from its inception as Olivetti Research Ltd in 1986, and is a Founding Director of seven other companies.

His research interests include networking, multimedia, and mobile systems. He received the BSc degree from the University of Wales in 1974 and the PhD degree from the University of Cambridge in 1978. He is a Fellow of the Royal Academy of Engineering.

References

[1] Gregory D. Abowd and Elizabeth D. Mynatt. Charting past, present, and future research in ubiquitous computing. *ACM Transactions on Computer-Human Interaction*, 7(1):29–58, March 2000.

[2] Paramvir Bahl and Venkata N. Padmanabhan. User location and tracking in an in-building radio network. Technical report, Microsoft Research, February 1999.

[3] Frazer Bennett, David Clarke, Joseph B. Evans, Andy Hopper, Alan Jones, and David Leask. Piconet - embedded mobile networking. *IEEE Personal Communications*, 4(5):8–15, October 1997.

[4] Bluetooth. A low-cost short-range radio networking solution. http://www.bluetooth.com/.

[5] BLVDS. A high speed multi-drop bus signalling architecture. http://www.national.com/appinfo/lvds/.

[6] Jeremy R. Cooperstock, Signey S. Fels, William Buxton, and Kenneth C. Smith. Reactive environments. *Communications of the ACM*, 40(9):65–73, September 1997.

[7] Diego Lopez de Ipina and Sai-Lai Lo. Sentient computing for everyone. In *Proceeding of DAIS 2001*. IFIP, November 2001.

[8] Hans-Werner Gellersen, Michael Beigl, and Holger Krull. The MediaCup: Awareness technology embedded in an everyday object. In *1st Int. Sym. Handheld and Ubiquitous Computing (HUC99)*, 1999.

[9] Robert Headon and Rupert Curwen. Recognizing movements from the ground reaction force. In *Workshop on Perceptive User Interfaces*, November 15-16 2001.

[10] Andy Hopper. The Clifford Paterson Lecture, 1999. Sentient Computing. *Philosophical Transactions of The Royal Society of London*, 358(1773):2349–2358, August 2000.

[11] I^2C. A networking solution for integrated circuits. http://www-us2.semiconductors.philips.com/i2c/.

[12] IEEE. *ANSI/IEEE Std 802.11*. IEEE, December 1999.

[13] James J. Kistler and M. Satyanarayanan. Disconnected operation in the Coda file system. *ACM Transactions on Computer Systems*, 10(1):3–25, 1992.

[14] Anurag Kumar. Comparative performance analysis of versions of TCP in a local network with a lossy link. *IEEE/ACM Transactions on Networking*, 6(4):485–498, 1998.

[15] Steve Mann. 'Smart Clothing': Wearable multimedia computing and 'personal imaging' to restore the technological balance between people and their environments. In *Proceedings of ACM Multimedia '96*. ACM, 1996.

[16] William Newman and Pierre Wellner. A desk supporting computer-based interaction with paper documents. In *Proceedings of CHI '92*. ACM, May 1992.

[17] Jun Rekimoto and Masanori Saitoh. Augmented surface: A spatially continuous work space for hybrid computing environments. In *Proceedings of CHI '99*. ACM, May 1999.

[18] Tristan Richardson, Quentin Stafford-Fraser, Kenneth R. Wood, and Andy Hopper. Virtual Network Computing. *IEEE Internet Computing*, 2(1):33–38, February 1998.

[19] SmartHome. Networked home products using X10 technology. http://www.smarthome.com/.

[20] Thad Starner, Dana Kirsh, and Solomon Assefa. The Locust Swarm: An environmentally-powered, networkless location and messaging system. In *Proceedings of the International Symposium on Wearable Computing '97*. IEEE Computer Society, 1997.

[21] Roy Want, Andy Hopper, Veronica Falcao, and Jonathon Gibbons. The Active Badge location system. *ACM Transactions on Information Systems*, 10(1):91–102, 1992.

[22] Andy Ward, Alan Jones, and Andy Hopper. A new location technique for the Active Office. *IEEE Personal Communications*, 4(5):42–47, October 1997.

[23] Mark Weiser. Some computer science problems in ubiquitous computing. *Communications of the ACM*, 36(7):75–84, July 1993.

[24] T. G. Zimmerman. Personal Area Networks: Near-field intrabody communication. *IBM Systems Journal*, 35(3,4):609–617, 1996.