The Implications of Pervasive Computing on Network Design

Bob Briscoe
<bob.briscoe@bt.com>
BT Research,
B54/130, Adastral Park, Martlesham Heath, Ipswich, IP5 3RE, UK

20 Nov 2004

Abstract

This paper concerns concerns the impact that pervasive computing will have on how we design networking. We identify the pressure points where pervasive computing will push current approaches to their limits, covering both technical and business implications. We use a broad definition of communications technology, to include not only infrastructure equipment and services, but also communications facilities within computing devices themselves.

We outline progress in redesigning the Internet for pervasive computing. We cover components of communications such as transport, routing and security. But we also consider how the industry will be arranged; explaining why new modes of communications (e.g. publish-subscribe) will become prevalent, where functions will be placed and how their deployment will happen. We give the rationale behind the most respected approaches being adopted. We give reasoned, if sometimes controversial, views of what *should* happen, built on our own research. We dispel some myths and outline the research agenda that still stands between us and realisation of the vision of pervasive computing.

1 Scope

Mark Weiser's late 1980s vision of an age of calm technology with pervasive computing disappearing into the fabric of the world [1] has been tempered by an industry-driven vision with more of a feel of conspicuous consumption. In the modified version, everyone carries around consumer electronics to provide natural, seamless interactions both with other people and with the information world, particularly for e-commerce, but still through a pervasive computing fabric.

Based on cost trends, trillions of devices globally have been predicted. Using a high-level economic argument we predict that most will be globally connected, at least at a rudimentary level. To a casual outsider, a global network of either form of pervasive computing would seem to require a major re-design of the world's communications systems. However, to a reasonable extent, assumptions made during the development of the Internet's architecture took into account the more obvious issues that pervasive computing would raise. For instance the packet abstraction allows for brief one-way datagrams which are ideally suited for reporting changes in the physical environment, and IPv6 has sufficient address space for perhaps 10^{18} addresses per square metre of the earth's surface[1]. Also, the Internet protocol is deliberately designed to abstract away from the creation of new link layer technologies (IP over everything), such as those appropriate for wireless links to battery-powered sensors or devices in hostile environments.

However, when one delves a little deeper, problems surface. The Internet architecture had substantially crystallised out just before Weiser's vision was articulated. Perhaps as a result, the Internet was largely built on assumptions of relatively stable, fixed but slightly unreliable connectivity, where availability of plentiful mains electricity was never questioned. Pervasive computing breaks these assumptions, requiring a redesign.

We focus on the more fundamental issues, such as addressing and routing, progressing to likely changes to traffic profiles which have implications on control of congestion and how to assure reliable delivery. We address security issues throughout. Before drawing the paper to a close with conclusions, we consider how the *technical* changes we have outlined will affect the *business* of networking and *vice versa*. But before we start on any of the above issues we give a reasoned argument for why the prevalent *mode* of communications for pervasive computing will be publish-subscribe rather than primarily request-reply.

[1] The IPv6 address space accommodates 3×10^{38} different IPv6 addresses. But depending on allocation efficiency [2], this would lead to anything from 1.5×10^{3} to 4×10^{18} addresses per square metre of the earth's surface.

2 Architecture

2.1 Conceptual model

No prediction of the impact of pervasive computing on network design can be undertaken without some characterisation of likely applications. Other articles survey expected applications [3] including case studies of its use for care in the community [4], in the home [5] and in the automotive sector [6]. The MIT Internet Zero project [7, 8] has built lights in flexible office spaces containing Internet-connected switches through which they are connected to mains power, while touch sensors that look like light switches can be programmed to instruct the actual switches over the Internet in order to alter their state. Probably the application closest to widespread deployment is the use of identity tags in the supply chain to replace bar codes, while sensors on work tools and materials offer the promise of automating the recording of daily activity for workers, particularly in the service industries.

We believe we can abstract all these pervasive computing applications into a single conceptual model, which we will use throughout the rest of this paper. Pervasive computing devices allow the creation and use of models in information space that represent various aspects of our changing physical space. We can create information models that track changes in the physical world and conversely arrange to change the physical world when we modify our virtual models of it.

As a concrete example, the switch within an Internet Zero light socket contains a simple (two state) information model of the light, and touch sensors hold the addresses of the lights they control within their connectivity model. When a touch sensor is pressed, it uses its connectivity model to determine which light it requests to change its state. In reality, the entire world's Internet Zero lights and touch sensors are physically connected to the Internet. But each connectivity model is a logical mask programmed into each touch sensor that limits its logical connectivity to only a few lights. The connectivity model in any touch sensor can itself be reprogrammed to allow it to control different lights (or an Internet hi-fi, for that matter). For instance, when a 'screwdriver' (in reality an authorised ID tag and reader in a screwdriver-like casing) is touched against a light then a touch sensor, it might reprogramme the touch sensor's connectivity model so that in future it switches that particular light.

Thus, pervasive computing devices, in their role as the interface between physical and information worlds, have two complementary roles:

- to translate the state of the physical world into information (via sensors) and vice versa (actuators), collectively termed digital transducers,
- in concert with many other transducers, to share (i.e. communicate) information about the physical world with other computers in order to collectively build more comprehensive models of the real world to be used for various purposes.

In this paper we focus nearly exclusively on communications driven by sensing the world. A full treatment of our subject would include actuators, but we choose a scope that brought out at least most of the major issues, as sensors tend to be more diffuse and the load from them far greater. Wide area tight feedback loops between the physical and information worlds would have been even more challenging but will have to be set aside for future work.

Every sensor creates an information model of the part of the physical phenomenon it covers, but usually such localised models serve little useful purpose alone. Many of these localised models need to be composed into a more comprehensive model, requiring communication from the sensors, then processing to correlate the parts into a greater whole. This process is illustrated on the left of Fig 1, where the visual model is created from multiple photographic scenes.

Also, multiple information models of different aspects of the same physical thing may be required. For example, referring again to the left of Fig 1, its visual appearance, its thermal properties, its extent in 3-D space (an edge or vector model), etc. The creation of certain information models might be within the capabilities of the computing power of the network of sensors themselves. The way the visual model's creation is depicted in Fig 1 works this way, composing a wide, high resolution 3-D visual scene across a 'sensor network' of simple devices, each connected to a simple camera. Other models might require more computing power than is available within the sensor network. In this case sensors (or networks of sensors) would have to report their findings to 'co-sensors' (or co-sensor-networks), which are computational processes running on more powerful computers (or across many computers). The figure shows the co-sensor case for thermal and edge models of the same original physical phenomenon.

Parts of each newly created model, may in themselves recursively form part of a more comprehensive information model of reality. The figure shows the visual model and the edge models being combined into a model of the 3-D extent and appearance of the original phenomenon. Further, these

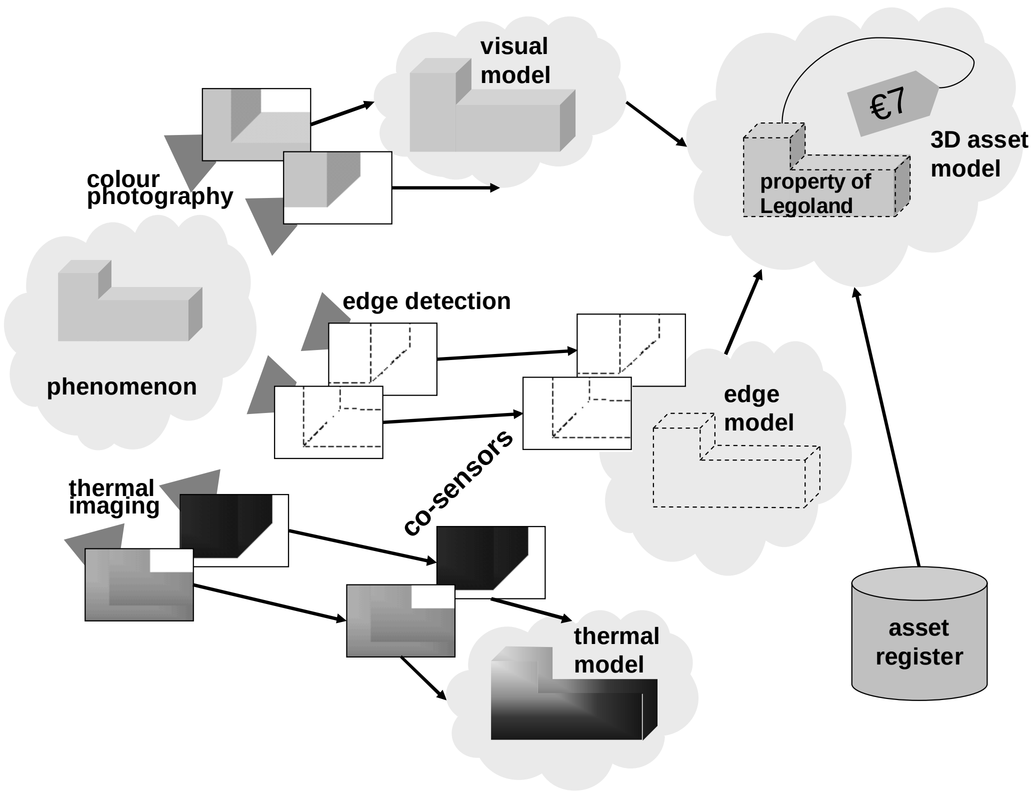

Figure 1: Information models of the environment

models may be combined with non-physical information, such as ownership, value etc, extracted from on-line databases, as shown on the right of the figure.

Note that there is no implication of some grand plan leading to the ultimate model of everything. Each of the millions of mini-models will be required in their own right and each by some different interested party, who may or may not be willing to pass on their model to help build other models. But we can expect some owners of these models to exploit them both for their own purposes and to sell to others for theirs — retailing and wholesaling respectively.

Continuing the example of the Internet Zero light switches as an illustration, behind each touch sensor might be multiple connectivity models; a default and one for each person who had ever chosen to reprogramme the lighting system for their office space (assuming the touch sensor was arranged to also detect the identity behind each touch). A further model correlating all these separate models together could be built, also encompassing a model of the presence of individuals built from other sensor inputs. So the office lighting could adapt, trying to satisfy every occupant, given potentially conflicting requirements where some people turned lights on that others turned off.

2.2 Layered connectivity

In the above scenarios, connectivity is implicit. Physical and logical connectivity has been deliberately arranged so that each stage in the system can connect to and be understood by the next stage in the model. It is true that humans will often deploy sensor networks for specific purposes and ensure that their wireless transmissions can all reach each other, that they are all transmitting on the same frequencies, with the same codings and with the same application level languages and formats. However, many applications will include the creation of connectivity as part of the application. And it will be desirable to be able to use existing devices and models to create new applications. So we have to allow devices to find each other and form into networks in the first place.

A classic, well-worn scenario is where an individual walks up to a shop window with her personal digital assistant (PDA) which connects to a monitor in the window to give the individual access to more screen space. Behind this simple idea lies a host of problems. There are dozens of radio link standards in fashion at any one time (Bluetooth, the WiFi family, etc), using different frequencies in each region. And hundreds of legacy standards still in use across thousands of different devices. To find each other by radio contact, the PDA would have to scan many frequency ranges and try all common coding, link and message formats (e.g. Jini, SLP, Twine, JESA, etc). The chances of such an approach leading to reliable seamless connectivity, any time, anywhere, are slim unless only one or two link standards predominate for all time.

An alternative approach would be to arrange at least the first stages of connectivity over a network common to any randomly chosen pair of devices, which was the purpose of introducing the Internet protocol — an intermediary between different link technologies ('IP over everything'). So both the PDA and the monitor use their own link technologies to connect to their respective points of presence on the Internet. The monitor advertises its location on a geographical extension of the Web (e.g. geographic mark-up language or SensorML), and the PDA, knowing its own location and coverage, uses a geographical search engine to find suitable monitors within range.

Clearly, the indirect approach to connectivity may also fail to connect our two devices due to lack of mutually understood standards at the application level (e.g. the XML format of the appropriate geographical search responses). However, the direct link approach has the same chances of mismatch at the application layer *on top of* the chances of mismatch at the physical, link and transport layers.

A logical model of the physical world has been created here in order to bootstrap creation of connectivity back in the physical world. Once the two devices have found each other in the information world, they can look up whether they have compatible interfaces in order to initiate a direct link. Failing that, they may have a good enough path to drive the graphics display through their Internet connection. Indeed, the monitor in our scenario may not have had any wireless connectivity at all, only having a fixed link to the Internet. Another advantage of the indirect connectivity approach is that devices can find each other in logical space even if the radio range of either of them is insufficient or not in line of sight.

A more far-reaching advantage is that logical objects can be placed in a model of the physical world even if they aren't located there in the real world. That is, a form of augmented reality, but within a model of reality, rather than the real thing (e.g. to test contingencies). If changes to the model were also able to drive events in the real world, it would be possible to test the response of the real world to simulated events.

On top of Internet connectivity, a further prerequisite for bootstrapping physical and logical connectivity in pervasive computing applications like that above will be comprehensive information

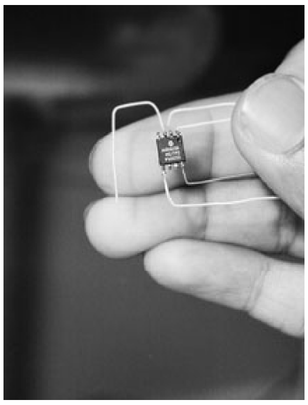

Figure 2: The iPic Web server

models of the location of things in the physical world [9, 10]. Again, we see the pattern where we start with potential connectivity to everything, then logically reduce connectivity to that possible within a confined geographical scope, then further reduce connectivity to that required for an application. Effectively we are creating overlay networks over overlay networks.

2.3 Modes of communications

Asynchronous communications Impressive work has been done compressing common communications protocols into tiny devices. For instance Shrikumar has implemented a Web server and all the protocols beneath it using 256B of code [11] (Fig 2). However, this example highlights a need to carefully consider what *mode* of communication is appropriate when interfacing with the physical environment. The Web works in request-reply mode. A client asks and the server responds. But this isn't how we *sense* the physical world. We don't ask the grass what colour it is. Photons arrive from the grass unsolicited. If we want to know whether it is dark yet, we don't keep asking a light sensor whether it's dark yet. It is far more efficient to configure the sensor to tell us when it's dark and, importantly, we will find out as soon as it gets dark, rather than the first time we happen to ask after it has got dark — *timely* notification of events. Thus, request-reply is appropriate for actuators, but less so for sensors.[2]

Thus a more appropriate mode of communication for sensors will be based on *asynchronous* events, a point made by Shrikumar himself about the applicability of his miniature Web server with respect to his earlier work on event notification embedded

[2]Request-reply has its place when a model is rarely used and can be stale between times.

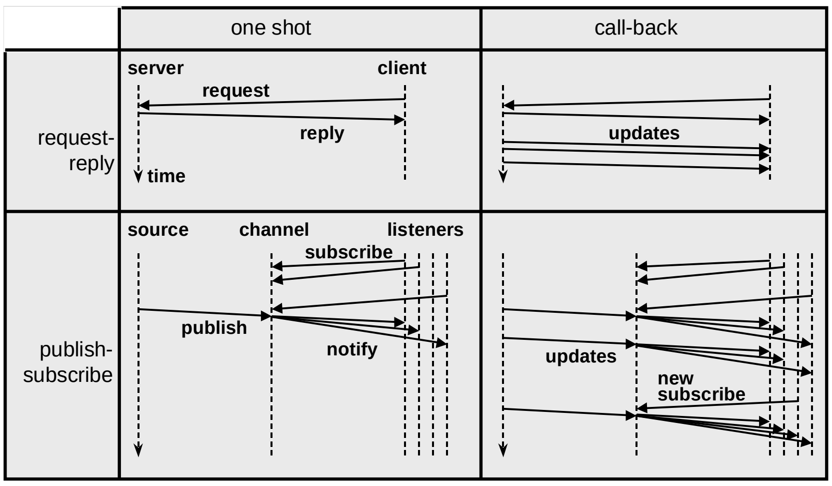

Figure 3: Communication modes

in devices [12]. An event should trigger in a sensor as its environment[3] changes, the magnitude of the necessary change determining the sensing *granularity*. An event notifies a change in the state of an object, allowing other copies or models of that object to update themselves, which in turn may require them to generate further events. Event notification requires the publish-subscribe (pub-sub) mode of communications (Fig 3) where an object publishes its *potential* to generate events to a logical channel, then listeners may subscribe to be notified when they occur. This model was popularised by Usenet and first commercialised by Teknekron Systems [13] (which became the information bus company or TIBCO).

In terms of the conceptual model built in the previous section, some information models will need to be physical-event-driven and *continuous*, requiring pub-sub feeds into them, while others will be created to fulfil one specific request, requiring request-reply. So each arrow representing communications between adjacent models in Fig 1 might involve either push or pull. However, an event-driven model can only receive genuinely timely events if *all* the models it depends on are also event-driven[4], right back to the physical phenomenon itself. Whereas a request-reply model will still be valid if it makes a request into a real-time model lower down the chain. Thus, all pervasive computing systems should support pub-sub.

In fact when one decomposes a request-reply service into its more rudimentary parts, one finds a pub-sub service is necessary within it. The request-reply service is merely a special event listener that holds

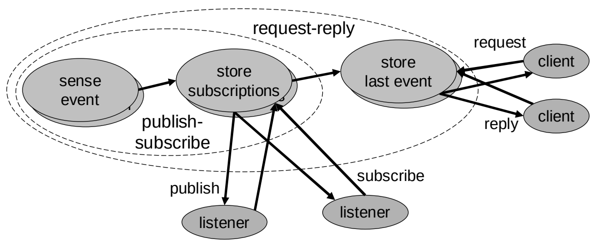

Figure 4: Decomposition of request-reply mode.

state, which it updates dependent on each event. It can then respond to requests for its latest state (Fig 4).

A common composition of both request-reply and pub-sub modes is where a pub-sub session is dynamically created (and eventually torn down) on request, with regular responses fed back to the requestor. This is an example of the call-back mode of communications (Fig 3). Later we will discuss an approach called directed diffusion [14] that works this way, with a request diffusing through a sensor network to the most appropriate sensors and regular responses following the same path in reverse to the requestor. We implemented another useful composition termed lookup-and-watch using our generic announcement protocol (GAP [15]), where the initial request gives an immediate reply, but also sets up a call-back session to notify future changes, the address of which is also published so others can join.

Having introduced the main modes of communication relevant to pervasive computing, we single out pub-sub for special attention below, examining two further features beyond timely notification that are particularly relevant to pervasive computing: group communications and no listening at the source.

[3]Time being part of a sensor's environment, which may be used to trigger scheduled reports on the rest of the sensor's environment.

[4]At the same or at a finer granularity.

Before moving on, we should clarify that we do not envisage the pub-sub mode being prevalent for machine-to-human communications, as few people want or need continual interruption (evident from the lack of take-up of Web push in the mid-1990s). Humans prefer the hybrid messaging mode where communications are queued until the recipient's attention is ready to turn to them. We only expect pub-sub mode to predominate for machine-to-machine communications.

Group communications Whereas request-reply is inherently point to point, pub-sub's inherent point to multi-point nature allows multiple parties to maintain the models they need. The underlying feeds from the real world may then contribute to a plethora of different views of the world.

But it would be unscalable to expect a sensor to remember a list of everyone interested in its output, because it would have no control over how much interest there might be. Also, every device listening for sensor events would have to remember to tell the sensor of any changes to its notification address while it was waiting.

Two approaches solve this problem. One provides a bare logical communication *channel* where those interested *join* the channel (e.g. IP multicast [16]). The other simply moves the problem to a proxy service for the sensor (e.g. the distributed event mediator service in the Cambridge Event Architecture [17]). In either approach, a sensor only has to remember one channel address and send just one message to it for each event, rather than overloading already challenged sensors with multiple requests.

Under the covers, both approaches are essentially similar. They are both generally implemented as numerous software objects, which are addressed collectively as a logical channel. Listeners declare their interest to their neighbouring instance of the channel, which remembers their interest and in turn subscribes onwards to other instances. When an event occurs, the source sends it to its neighbouring instance of the channel, which distributes it onwards towards wherever interest has been declared (Fig 8).

However, the fundamental difference is that the mediator provides a bundle of event-related services while the bare communications channel deliberately only provides dumb forwarding. So a bare channel could as easily be implemented either as a radio channel, or as a collection of software subscription lists as described above. The bare channel approach follows end to end design principles [18], deliberately keeping the common communications service rudimentary and *complementing* it with other services such as event archiving and recovery, but *only* if required. If recall of previous events were required, perhaps to help late joiners or those temporarily disconnected, the event mediator would provide this service itself, whereas without knowing why, the bare channel would simply forward the information to an archiving node because it had joined the channel in order to provide this service. Other approaches sit at intermediate points on this spectrum. For instance, Microsoft's SCRIBE [19] is a bare channel service, but also offers reliable delivery (discussed in §3.5).

Of course, we cannot expect everyone to share their data freely. Confidentiality may be required to protect trade, individual privacy [20], or other differences of interest. Again the above two approaches differ in how they offer confidentiality. An event mediator service includes an access control function that acts fully on behalf of the sensor. This requires each listener to give appropriate credentials before being allowed to join, while the dumb channel again leaves confidentiality to another service. The sensed data would have to be encrypted before sending to the channel, while another service would control access to the decryption key. So anyone could join the channel, but they would not understand it without the key. The merits of each approach are discussed in the sections on function placement (§2.4) and on security (§3.6).

Announce or listen A further advantage of the pub-sub mode is that the sensor has no need to listen for requests in 'server mode'[5]. Wireless sensors would very quickly run down their batteries if they had to leave their radio receiver running to listen for arbitrary incoming requests. It is far less power-consuming to leave only the sensing mechanism running, 'listening' for changes to the physical environment, which can then trigger power up of the radio circuitry only to send notification of the event.

But, when discussing our conceptual model earlier, we emphasised that pervasive computing systems need to be able to respond to arbitrary requests as well as asynchronous notification of events. Does this not mean that their radio receivers will have to remain powered up anyway? The answer lies in separating event memory from event communication. In order to answer requests for the *current* temperature, or the *current* state of a touch sensor, the sensor itself doesn't need to remember its own state. As long as something else (connected

[5]The term server relates to any process able to respond to arbitrary incoming requests, irrespective of whether it runs on a large machine often called a 'server' or on a tiny sensor.

to mains power) receives events from the sensor, it can respond to requests for the sensor's current state between times.

Proving many of the points so far, Shrikumar's tiny Web server was soon taken off the Internet, as its serial link was continually overloaded with requests from curious browsers. Its content is now served from a larger mirror server.

2.4 Function placement

Evolvability A system's evolvability is highly sensitive to the placement of communications functions, which is the chief concern of the Internet's end-to-end design principle [18]. The need for evolvability of today's communications infrastructure is obvious, given large investment costs require future-proofing. But one may wonder why evolvability of pervasive computing devices is a concern, given the growth of pervasive computing is based on projections of plummeting device costs [1]. Can we not assume each device is disposable, only ever being used for the purpose for which it was first deployed? The issue here is that the equipment is only a part of the cost of a new application. Deployment is a major cost consideration, so it is highly desirable to put devices deployed for one purpose to use for others. For instance, to add a one penny wireless sensor beneath the drum of your washing machine costs a lot more than remotely reusing the one that is already there.

Open by design, closed by choice In §2.2 on layered connectivity we recursively created confined connectivity within wider potential connectivity. We argue that the value of global connectivity greatly outweighs the cost. Metcalfe showed that the value of the connectivity of each device rises with the number of other devices it could potentially connect to, making global connectivity highly valuable. Whereas the cost of fully standards compliant Internet connectivity is about a mere 200B of code [11][6] plus the running cost of the processing and bandwidth required for the extra header layer (compressible to a byte or so). Devices might not carry an Internet stack, but instead connect to a hub that does. However, the added complexity of arranging and managing a hub hardly seems worthwhile given the low cost of direct Internet connectivity. Therefore economies of volume manufacture are likely to lead to every device design including an Internet stack, whether or not it is needed for every application.

[6] 2.5% of the memory of a Berkeley mote, which is likely to grow in capacity and shrink in size in line with Moore's Law

Global connectivity is also highly desirable for evolvability, otherwise new uses will be constrained by locality. For example, initially it may not seem necessary to give light switches global connectivity, but one day you might appreciate the facility to turn your lights off from your mobile phone while on holiday. However, more controversially, others may be able to remotely control or monitor your lights. One approach to this privacy problem is to place inherent accessibility limits in the design (e.g. to design a car door lock so it can only be opened within a limited range). But this only limits the first hop, which cannot be prevented from networking onward globally. A safer approach is to assume global accessibility might exist, and to close off undesirable uses logically rather than physically. To fully stress network design, we take the 'assume open then close' approach. However, we accept that designers of car security systems and the like will care more about present security than future evolvability. This dilemma is further explored in another article specifically on privacy [20].

Dumb relays Whether or not the devices themselves should be evolvable, it is certain that the rest of the systems they work with should be. All too often, principles of evolvable system design are too readily set aside in pervasive computing demonstrators. Rather than designing for messages from sensors to be transmitted to and from any other computer in the world, functions are often embedded in the base-stations that interface between the sensor and the rest of the world, perhaps doing some address or protocol conversion (e.g. from sensor-local to global). We discuss some cases below where intelligent base-stations can be justified on principle. But if intelligent relays are necessary to a design, the end-to-end design principle teaches us that it is worth taking a long hard look at alternative designs that confine intermediate relays to dumb forwarding.

Ideally, the sensors themselves should be first class Internet citizens. But failing that, the remote end-points they communicate with should supplement their limited capabilities, *not* neighbouring relays (so the fate of a communication depends only on state and trust shared between the end-points of the communication). Tying higher layer functions to specific instances of equipment operated by specific parties will limit flexibility, especially for new business models.

Power conservation can be a valid reason to relax the end-to-end principle, with base-stations taking on additional functions by virtue of their role as the first mains-powered node in the path. But this need *not* be an excuse to open the flood-gates to more functions than necessary. For instance, strong encryption is possible on challenged devices (see §3.6), so sensors should encrypt their data for confidentiality and the base-station should merely relay the encrypted data to another destination. Functions such as re-transmission can be achieved without access to encryption keys, the goal being to limit access to as few parties as possible. For instance it would be inadvisable to embed an event mediator service in the base-station, which decrypted events and controlled further access to them itself. This would limit future uses of the system to ones where the base-station operator was trusted by *everyone* who might want these future uses.

However, it may be more efficient for the sensors themselves to behave as intelligent infrastructure (relays) for the nodes around them — multi-hop sensor networks. Taking the sensor network as a whole, it has been proven that combining data aggregation with the relay function within each sensor saves considerable battery power for typical diffuse sensing applications [21]. When battery power is paramount relative to evolvability, it seems we have to compromise and make infrastructure more application specific — counter to end-to-end design principles. We return to this subject in §3.3 on routing.

3 Component services

So, faced with all the choices above and more to come below, which course should a wireless sensor manufacturer or network equipment vendor take? Is there a generic design for communications with miniature devices? Which communications services should service providers be considering? Of course, the detailed answer might depend on the specifics of the application or the devices required for it. But, the following sections discuss progress towards defining generic component services so that as many specific requirements as possible can be satisfied by building applications over these components.

To kick off the discussion we propose a straw man design for the communications systems of a miniature device. We believe that a large range of applications could be satisfied by this apparently strange proposal. We briefly outline the reasoning behind the proposal, but we do *not* intend to imply that any other approach must be inferior in all circumstances. The architectural discussion above was necessarily conducted at a fairly abstract level. We put up a straw man at this point to deliberately force a change of pace in the paper. As we proceed through the following sections, the choices widen further for each aspect: routing, addressing, reliable delivery, congestion control, traffic profiles, security and other ancillary communications services. We hope the straw man will provoke thought while reading those subsequent sections, and remind the reader that the economies of scale necessary for the pervasive computing vision require device manufacturers to commit to a design that will survive a large production run.

3.1 A straw man proposal

We would recommend a configuration-free device (Fig 5) that powered up its radio transmitter whenever it sensed a threshold change to its environment. It could then send a 'fire-and-forget' datagram to a hard-coded multicast address[7] and immediately power down its transmitter. It would also send heartbeat messages, confirming the previously sent event at regular intervals[8]. The device would have to be one hop from its base-station. It should be relatively tamper-resistant, at least outputting a warning if tampered with, so that it could hold an internally stored key, which it would use to seed a regularly changing pseudo-random sequence of keys used to encrypt its messages.

Our reasoning for choosing this design is that, although a sensor may only be able to perform one simple function, to bring down its cost of production, it should be manufactured in volume. So its single function will need to be applied in many different ways. So the device must be secure to some degree even though it may not always need to be. And we have deliberately chosen to allow it to be re-purposed pre- and post-deployment but *without* re-configuration, so that the device need have no server listening for configuration changes, saving battery power and improving inherent security. For instance, proofing it against the sleep deprivation torture attack [22], which leaves any node that responds to arbitrary requests vulnerable to rapid exhaustion of its battery.

We have recommended multicast addressing, as otherwise a device hard-coded to send to a unicast address is permanently tied to a specific corresponding node. Multicast addresses have no topological significance, so the receiver of each message can be changed without involving the sender, instead only requiring potential receiver(s) to be configured to listen. Multicast addresses need only be loosely unique, as a multicast transmission can be

[7]Preferably IP multicast, but otherwise a multicast medium access control (MAC) address, with a different one in each device.

[8]Slow changes to the interval could be arranged with repeated advance warnings.

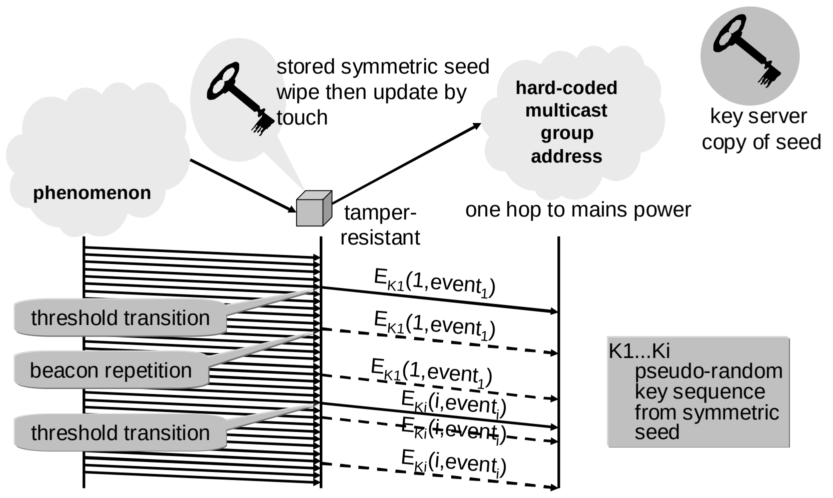

Figure 5: Straw man device design.

locally scoped, to prevent clashes with transmissions to the same address. Any receiver could re-multicast with a wider scope if necessary.

The device's multicast event notifications could be limited to just one receiver, by not revealing the seed key more widely. Or the seed key could be published if there were no need for confidentiality. Between these two extremes, time-bounded parts of the pseudo-random sequence could be revealed to selected groups by a separate key management server (as in our own MARKS key management scheme [23]).

At least one receiver (perhaps the only authorised receiver), could maintain the sensor's state, acting as its proxy for arbitrary incoming requests, and archiving events for late joiners. The sensor would not be able to receive acknowledgements for anything it sent, so, if a receiver missed a heartbeat, it would need to rely on other receivers for a re-transmission (as in SRM [24]) or, if there were no other successful deliveries, wait until the next heartbeat.

This straw man proposal is intended to cover a wide set of circumstances, but only where each sensor alone has sufficient radio range to reach a mains-powered communications device. Multi-hop networks of sensors would not be possible with send-only sensors. Multi-hop sensor networks come up repeatedly in the rest of this paper, being a major focus of the research agenda. But that doesn't make this straw man proposal irrelevant, given single hop wireless sensors may turn out to be more prevalent in practice.

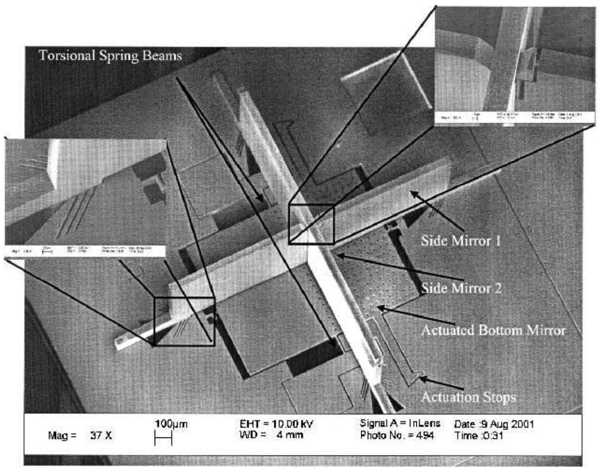

Figure 6: Scanning electron microscope picture of a corner cube reflector.

3.2 Unusual link technologies

Given power constraints, communications links to pervasive computing leaf nodes often use highly constraining technology. Connectivity may be the exception rather than the rule [25, 26], due to line-of-sight issues as much as power conservation. Link availability may have to be pre-scheduled or opportunistic. Error rates may be high even when connected.

Further, the range of one end-point may be much greater than that of the other (typically where one end is connected to mains power), implying that often connectivity will be unidirectional. Nonetheless, novel technologies have been built to scavenge energy from communications in one direction, modulating it in order to respond in the other. Radio frequency identity tags work this way, capturing the excitation energy in the reader's radio transmission until sufficient energy has been built up to respond by modulating the incoming signal. Microelectromechanical (MEM) fabrication has been used at UC Berkeley to build corner cube reflectors on 'smart dust' devices (Fig 6). It is projected that smart dust will be small enough to float in air or water. The reflectors work on the same principle as roadway cat's eyes, but the angle of one face can be varied slightly to modulate the reflection of an incoming optical beam back to its source [27]. The team have manufactured a sub-millimetre cube and achieved 400b/s over 180m with good signal-to-noise ratio [28], while their analysis claims many kb/s are possible over hundreds of metres in bright sunlight.

Such power scavenging technologies seem more suited to request-reply mode, where a reader base-station requests a reading and the sensor responds. However, smart dust is equipped with hybrid communications technology, so that it can hail the base-station with a short burst of radio broadcast. The base-station then shines its optical beam in order to pick up the message the sensor has for it [27], giving the sensor on-demand access to the link for minimal energy cost.

Even with more conventional wireless link technologies, there is considerable scope for improving power efficiency by designing medium access [29, 30] and transport protocols [31] to minimise energy use, with only minor compromises in traditional performance metrics necessary.

All this novel link technology helps, but doesn't remove the problem of poor connectivity, which changes the rules for designing all the layers built over it, right up to the application. In our own index-based event-messaging system [32], eventual message delivery over disconnections is provided by the managed messaging layer on announcers and listeners. Messages are clustered into groups based on interest and indexes of the current version of messages in each cluster are beaconed at regular intervals. As nodes regain connectivity, their managed messaging layer transparently finds which messages they have missed and accesses them from an event archive.

3.3 Routing

The task of routing messages through an ad hoc network of mobile battery-powered nodes is hard enough when the nodes are allowed large, rechargeable batteries. A 2003 review of practical progress [33] revealed most research was still stronger in theory than practice. Just one implementation (as opposed to simulation) of a routing protocol (AODV [34]) was able to reliably route packets over more than one hop. These were results from the Internet Engineering Task Force's (IETF's) Mobile Ad Hoc Networks (MANET) working group, which has been chartered since 1995/6 to focus on standardising protocols that are near to market.

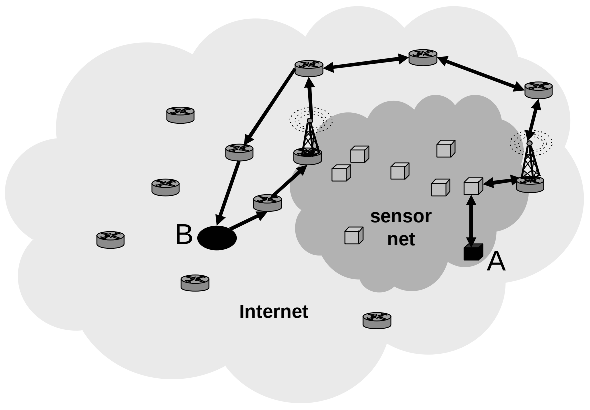

Figure 7: Hybrid mains/battery powered routing with a deliberately tortuous mains-powered path, to avoid concerning Internet routing with power optimisation.

To realise the vision of pervasive computing, far more challenging scenarios are envisaged. Routing protocols have been implemented for large numbers of tiny, dust-sized devices with very short range, where each second of battery use is one second closer to death. In such scenarios, few nodes are within range of a gateway giving access to wider networks, but collectively a network of sensors should be able to relay most nodes' messages to a gateway.

Routing hierarchy Routing research for sensor networks has rightly separated routing within the sensor network from the routing beyond it. Although there may be a choice of routes *out* of the sensor network, it is rarely necessary to consider optimising routes across both networks at once, as the sensor network leg is far more resource-critical (Fig 7). However, ensuring that messages *destined for* the sensor network enter it from the most efficient gateway requires a model of the sensor network to be held by routers outside the domain. Baker [35]) has argued that, if routing nodes are relatively powerful (e.g. MANETs being considered by the IETF), popular link state routing protocols can be sufficient. They can be tuned to allow the combination of fixed and mobile nodes to work together effectively, without too much power and memory consumption within the MANET.

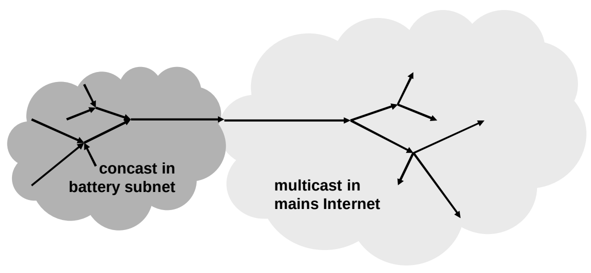

Figure 9: Energy shortage makes concast, not multicast the most efficient communications mode.

System-wide energy conservation Routing protocols are generally designed to be agnostic about which route cost-metrics to use. So it is relatively straightforward to replace traditional hop metrics with power-aware metrics. Singh *et al* [36] showed mean time to battery exhaustion for a whole mobile ad hoc network could be considerably improved by using such metrics.

Chang *et al* [37] proved analytically that battery lives across a whole network of fixed but wireless devices could be maximised with no need for global co-ordination, only requiring an algorithm local to each node that ensured local energy consumption was proportional to local energy reserves. However, often the lifetimes of critically placed nodes become the determining factor for the lifetime of the whole network (e.g. those close to a base-station).

Application-specific concast In §2.3 we explained the need for pub-sub mode of communications and introduced the requirement for routing events to multiple parties. However, sensor network routing has avoided multicasting, as it is more power efficient to relay events across sensor nodes to a single gateway rather than duplicating the message and the power required to transmit it. From there, the multicast model can be offered, with multiple parties subscribing to a logical event channel.

In 2001, researchers at the Internet Sciences Institute (ISI) showed that when on-net capacity (within the sensor network) is critical, it is more efficient to hold back sensor data until it can be aggregated with other readings *while* routing them (Fig 9), rather than sending each reading immediately and separately to an off-net reader to be aggregated there [21]. This proved the correctness of their seminal earlier work called directed diffusion [14].

Directed diffusion was a radical departure. Traditionally routing is based on node addresses, ensuring everywhere in the network maintains routes to any other address so that applications can send messages to other nodes by their addresses. Instead, directed diffusion addressed messages by the *content* of the question that required answering. And nodes advertised the answers they could give by the content of the questions they could answer. Thus, for a sensor reading such as "temperature = 17" the relevant content name is "temperature". This was the first instance of what came to be termed content-addressable networks (CANs).[9] However unlike subsequent CANs, directed diffusion worked without the benefit of any underlying routing. Sensors capable of answering "temperature" questions would advertise this fact to their neighbours, who would flood this fact outwards. Queries about "temperature" could then be routed to the correct sensors, leaving behind a trail of 'bread-crumbs' so return messages could be reverse routed to the original place the query had entered the network. While return messages were relayed across the sensor network, the temperature values could be aggregated. For instance, the weighted mean might be calculated before forwarding on the result.

The aggregation functions to use and the routing logic of each directed diffusion session were highly application specific. As explained earlier, evolvability has to be compromised if other factors (e.g. power conservation) are paramount. But then, in 2002, researchers from the database community at UC Berkeley generalised the ideas in directed diffusion, using common aggregation primitives found in commercial databases (COUNT, MIN, MAX, SUM, and AVERAGE) [38]. Thus, a more generic routing capability has been implemented in the TinyOS operating system built for the Berkeley motes (Fig 12). It is called TinyDB, given it involves the rather unexpected inclusion of database aggregation primitives within a routing protocol.

In fact, generic aggregation functions at merge points in a network were recognised as important outside the sensor networking field in the late 1990s, appearing in Cisco's generic router assist (GRA) technology [39] — a generalisation of all concast modes of communications. In concast, multiple messages converge, with some combination function at each merge point (Fig 8). In the mid-1990s, concast was recognised as a valid communications mode, used primarily when data traversed a multicast tree in reverse, for functions like aggregating multicast feedback or merging bandwidth reservations.

We should add that both directed diffusion and TinyDB allow for a time series of responses to an initial query. In this respect, they use the call-back

[9]CANs became prevalent the next year for routing in peer-to-peer overlay networks for file-sharing and related applications.

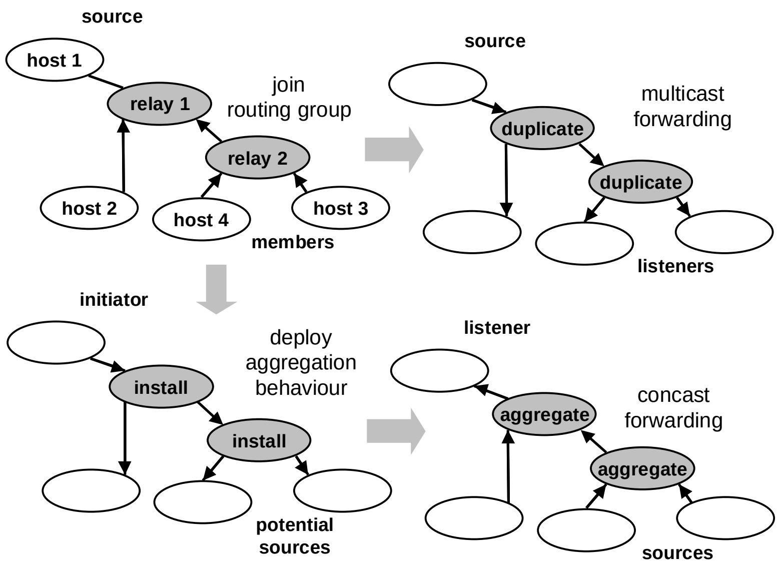

Figure 8: Multicast and concast group formation and forwarding.

mode of communications introduced in §2.3, creating a logical pub-sub channel (e.g. on the "temperature" topic) in response to a session initiation request.

Unusual routing for unusual links Beyond designing for challenging power requirements, routing protocols are also having to be re-designed to cope with inconsistent link availability (see §3.2 earlier). If sensor nodes are acting as relays in multi-hop networks, but they can only power up intermittently, they all have to synchronise to get any relaying done.

Also, routing is notoriously difficult with unidirectional links (as routing inherently concerns passing advertisements of downstream connectivity to upstream nodes). Routing over unidirectional links has been solved for the classic example of satellite down-links, by creating a tunnelled virtual link along a back-channel, through which routing messages can be passed [40]. However, routing over links that become unidirectional unpredictably is still a research issue.

Routing security If all the above research issues were not enough, routing in ad hoc networks presents numerous further unsolved security challenges. Already authentication of route advertisements is a difficult area if there is a high churn of network operators. It is unrealistic to expect trillions of devices to appear in the next decades unless *billions* of networks also appear to connect them together, implying extremely rapid appearance of new network operators (unless *all* networking is out-sourced to the few existing operators, which seems hopelessly optimistic).

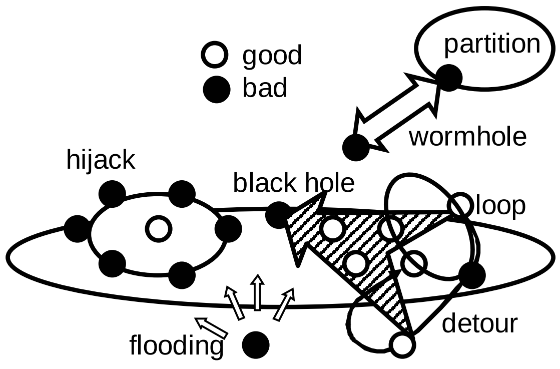

Figure 10: Types of attack on routing (from [43]).

In order to validate routing message authentication, any one network must know whom it would *expect* to receive adverts from. So authentication of routes through newly appearing networks will require trusted third parties to introduce the new players. Also, just because a route advertisement is authenticated, doesn't mean it is honest, as pointed out by Perlman in 1988 [41]. Commercial networks have an incentive to understate the cost metrics of their routes to attract extra custom. So unless the cost of an advertised route can be objectively tested (as, for example, in feedback-based routing [42]), authentication means little.

Although problems of authentication are hard, the range of possible attacks on the *availability* of ad hoc networks are even harder to solve. We have already mentioned sleep deprivation torture attacks on power constrained nodes. Law *et al* list a range of further attacks in their excellent introduction to this area [43]: black-holing, flooding, detours, loops, worm-holes, blackmailing and hijacking (Fig 10). Each node's finite battery power makes it vulnerable to any attack that requires it to use its energy before it can determine whether it is being attacked.

Clearly, if any section of society takes a dislike to certain wireless devices in their environment, they will only need to identify one of these vulnerabilities to attack them. The send-only sensor in the straw man proposal above, is a pragmatic way to avoid many of the vulnerabilities, although it is still susceptible to physical layer attacks such as channel jamming and the like.

3.4 Naming and addressing

To a casual observer, the size of the required address space for a computing network should relate to the number of hardware devices in the world. However, if pub-sub communications become predominant, the size of the problem will be determined by the number of software objects that have an audience of other software objects watching for updates to their state — potentially far more numerous than the trillions of devices on which they sit.

Encapsulation Hierarchy has invariably been used to ensure addressing schemes can scale. For instance, variable names used within a programme resolve to memory addresses, but the same variable name or memory address used in two programmes (or even two instances of the same programme) are distinguished by the identity of the process in which they run (strictly the identity of the interface to the process). Continuing up the hierarchy, processes are identified relative to the interfaces of the machines on which they run. Machines are identified relative to their subnetwork and so on.

Address aggregation However, large distributed processes like environment monitoring sit across machines rather than within one machine, inverting the 'normal' hierarchy[10]. Sharing the variables of the process across machines causes the scaling problem introduced above. Imagine a large set of sensors also acting as relays for each other. Imagine that a subset comprises temperature sensors, which, along with those interested in their readings, all hold a common interest in the group-name "temperature". We explained earlier (§2.3) why the only practical approach here is for each sensor to send to a logical channel — pub-sub mode — rather than each hold a list of interested listeners. The logical channel is created by the interest of the listeners and the combined routing of all the relays. So every relay has to remember the list of those logical neighbours to which to forward messages concerning the temperature question. This list consumes a small amount of memory on each relay.

[10]Distributed processes will be the norm in the future.

Another subset of the same sensors may be gathering road traffic statistics. So each relay also has to set aside memory to route messages for the "road-traffic" group. If there are a large number of network-wide processes being supported across the sensor network, there could be a routing group for each variable in each process — potentially one routing group for each software object on the network. Alternatively, many groups can be combined into one, so that some end-points have to filter out messages they aren't interested in (a problem familiar to people who use e-mail lists). However many groups there are, each relay will have to store a different neighbour list for each group.

Unfortunately, there is no generic way for each relay to aggregate the per-group neighbour lists it holds, so the memory required on each relay grows linearly with the number of routing groups. Each pair (group-name, neighbour-list) bears little relation to any other pair. Even if numbers are uniquely assigned to each group, in a numeric order that helps one node to group its neighbour-lists efficiently, this order is unlikely to help any others. In 2001, this problem was articulated formally as the 'channelisation problem' [44], also applying to radio channel allocation. Given we believe pub-sub will be the predominant mode of communications in pervasive computing, the channelisation problem puts inherent limits on the economic viability of pervasive computing. The designers of directed diffusion and TinyDB have not had to worry about this problem, because current sensor networks are application-specific, requiring only a few routing groups. However, if our conceptual model of pervasive computing (§2.1) becomes prevalent on a global scale, the channelisation problem will arise in the (mains-powered) networks connecting together the physical and information worlds. We predict that events from the physical world will spread out across a web of listeners around the globe, with little correlation by location.

Until recently, only limited group aggregation potential was considered feasible on relays [45]. A naïve solution is to use one coarser logical channel

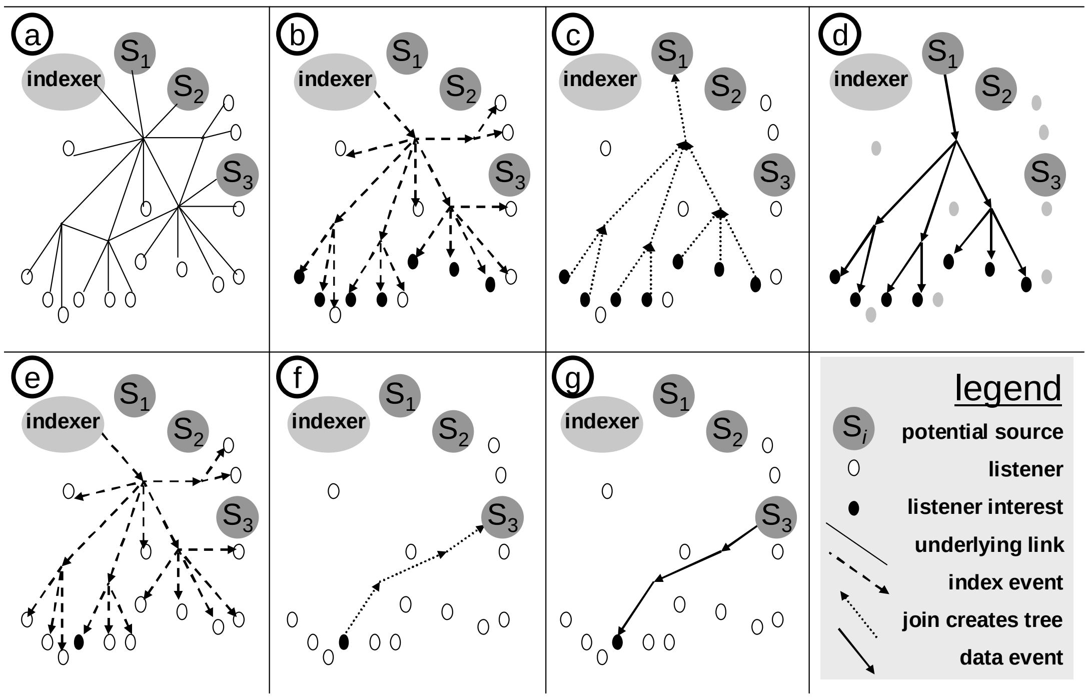

Figure 11: Index-based event messaging.

to replace many more granular ones and just expect receivers to filter out what doesn't concern them. Some of the filtering burden can be carried by the network, even using commercially available Internet routing products. Hop-by-hop filtering can be added to these products [46] using their support for generic concast functions [39] (see §3.3).

Our own solution to this problem [32] takes an end-to-end[11] rather than hop-by-hop approach. We use an addressing scheme where any object may take a globally unique[12] and meaningless address for the sequence of events it will generate as its state changes. Then it gives each event within the sequence an incrementing version number. Our aim was to solve the address scalability problem, but without revealing any more than necessary to intermediate parties, given messages will often contain commercial or private information.

We create routing state on relays for very granular routing groups (one for each event sequence) but only for the brief time it is needed to pass a message, which at first sight seems to make sending anything impossible. But we introduce indexers that regularly beacon a list of related event identifiers along with their latest versions. Listeners monitor the index or indexes of their choice in order to hear when a message arises for them. So the persistent routing groups for the indexes will only be held open on relays for those indexes with an audience.[13] Indexes beacon regularly for reliability as listeners come and go (see §3.5. The indexing function may itself be distributed. Changes to indexes are notified to end-points interested in them using the same messaging system as for the events themselves, and indexes may themselves be further indexed recursively. When an event source has a message to send, it first asks the indexer to send out an update (Fig 11b or e). Those listening then send join requests to their closest relays, creating the group's routing structure (Fig 11c or f) only for the brief time required to send the event (Fig 11d or g). The solution makes memory usage scalable as more groups are required, but can trade this off against the greater need for messages to co-ordinate the indexes.

An approach using similar intermediate structures to the above indexes was proposed by Kulik at MIT [47]. However, in Kulik's approach every relay stores the relevant parts of the directed acyclic graphs relating indexes together, whereas in our scheme listeners store the relevant branches of the graph. Kulik's approach clearly speeds up event notification at the expense of more storage on relays.

[11]Throughout the paper, the term end-to-end is used in its technical sense to mean "involving only the end-points to provide the function in question".

[12]Messages contain only a locally unique sub-range of the address to reduce overhead.

[13]On a longer timescale, listeners reveal all the sequences they are interested in so that indexers can optimise which sequences are announced in which index, around how listener interest is clustered.

Topographical addressing We have already briefly discussed requirements for mapping between physical space and the logical addresses of processes on computing devices (§2.2). In order for two software objects to establish whether they are close in a modelled co-ordinate space, the most efficient approach is to create further objects to represent each volume of space. The model of each space holds the identifiers of objects placed within it and announces whenever an object enters or leaves it. Then, any object has to monitor the object representing the space it occupies in order to find other nearby objects, rather than having to continually interrogate the position of all objects in the world in case any one of them passes close by.[14]

Thus we can immediately see a scenario where every space (of varying volume) in the physical world is represented by an object in the information world, which many other objects that have placed themselves in that space are continuously monitoring. Just this single proximity application would be sufficient to cause the group address aggregation scaling problems we have outlined — when trillions of objects are watching for changes in trillions of other objects. Further, many service providers will maintain space models requiring co-ordination between multiple models of the same space.

Internetworking beyond the Internet As networks of computing devices are formed, they may all inherit their networking characteristics from the evolving Internet architecture. However, new independent forms of network may develop without any overarching architecture common to all the networks. Overlays that work across multiple network architectures are perfectly feasible, the Internet itself being the classic example. But federations of different network architectures are fraught with problems.

The Plutarch proposal [48] considers what is necessary to federate multiple addressing architectures, also making some inroads into wider federation issues. But address resolution differences can be confined to end-points and clearly identifiable gateways between architectures (cf. IPv4 to IPv6 gateways). The hard problems start when federating data handling *along* the whole network path: detecting routing loops or congestion, or controlling priority, capacity reservation or routing. The *gain* from not using an Internet-wide architecture would have to override the *pain* suffered dealing with such a wide range of issues.

[14]Recall that any object might jump to anywhere in cyberspace.

3.5 Data transport issues

Traffic profiles If computing does become pervasive one might assume that a majority of traffic flows would consist of just a few packets, with single datagram flows being common. But we cannot predict whether these bursts would constitute a large proportion of total traffic *volume*. In the last half of the 1990s, due to the rise in popularity of the Web, flows of a few packets (< 20) constituted the large majority of the flows (95%) and a sizeable proportion of the volume of traffic on the Internet (see, for example, Brownlee & Ma [49]). The rise in the popularity of peer-to-peer file sharing between 2000 and 2004 has seen longer lasting flows take over in prominence. Thus, not only would it be foolish to try to predict the future traffic mix, it would be reckless to design networks and protocols without contingency for a range of scenarios.

Even our intuition about the traffic implications of *current* uses of the Internet can be misguided. Most people imagine multicast is used to stream long-running sessions. Although this use does indeed constitute a large *volume* of multicast traffic, a study of nearly four million native multicast *flows*[15] [50] found over 75% consisted of just one packet. Closer investigation found a large proportion of these consisted of co-ordination messages to announce the existence of other sessions through various multicast indexing tools. The same bimodal distribution caused by the division in human behaviour between managing and doing (whether work, social interactions or whatever) is seen in unicast traffic [51]. But, past behaviour (mostly from human-attended sessions) is not a reliable guide to the future traffic mix if *un*attended sessions come to predominate.

For instance, we might see huge avalanches of messages firing across the Internet following major changes in the real world (road-traffic accidents, battles, security advisories, etc). Network management event avalanches already create problems that are hard to quantify until they happen. Once the information world is much more tightly coupled to the real world, the complex chains of dependency between messages (that may also get lost during such events) may become near impossible to design for.

Congestion control If the large majority of Internet traffic volume does turn out to consist of little bursts of packets between uncorrelated end-points, current congestion control techniques will be largely useless. In order to pace its rate, a

[15]Monitored throughout July 2001 on MCI/WorldCom's backbone

sender should mimic the behaviour of the transmission control protocol (TCP-friendly), which relies on congestion feedback from routers on the path. When a sender starts a new flow, it carefully introduces packets to the network, sending two for every one that gets acknowledged by the receiver, until it senses congestion. However, brief flows finish long before they discover the rate they could have used, because they have to proceed so cautiously.

If the traffic mix is mainly short flows, there may be insufficient long-running flows to use up the remaining network capacity. So there will come a time when further increases in network capacity will not bring any benefit. TCP-friendly slow-start algorithms will become the limiting factor for network capacity. Of course, the contrary approach would not work either; where everyone agrees that TCP should be allowed to ramp up more aggressively. We could then return to the episodes of persistent congestion collapse that led to the introduction of TCP in the 1980s.

Further, in the pervasive computing vision, distributed computations will always be fighting against unavoidable propagation delays due to the speed of light. Already many GRID applications use parallelism extremely judiciously, because the messaging delays between computers are so great relative to local delays. TCP's 'slow-start' puts an inherent limit of $O(\log(n))$ on the latency of a short burst of n packets. Whereas often a whole burst could have been served in one round trip, but that can only be discovered in retrospect.[16]

The data rate of reality As we have mentioned, sensor networks can tend to generate event avalanches. The Anderson project at Columbia Uni has proposed a novel congestion control mechanisms, tailored to the particular problems of sensor networks [52]. The problem is that traditional congestion control only exerts back pressure on the sender's transmission rate. Herein lie two problems:

- the ultimate sender is the real world — it is obviously not possible to slow the data rate of reality,
- often, there is no backlog of messages at the sender which may have only contributed one message — congestion only arises at certain unfortunate relays in the sensor network.

By definition, congestion implies a high probability of not being able to service a request due to high utilisation of resources. So, again by definition, if a relay is congested, it will have no spare buffer space up its sleeve. So, once a relay is congested, it has very little room for manoeuvre, given all the data has probably already left the sources that caused the congestion. All that can be done is to ask upstream relays to hold back data, and sources to buffer any new data arriving 'from reality'. Clearly though, as events continue to unfold in the real world, if congestion persists there will come a point where data has to be dropped.

We should emphasise that some messages, no matter how tiny, might carry very important information. But the importance will rarely be understood by the sensor or relay — often it will only be possible to establish how important a message is once its meaning has been extracted by the receiver.[17] As with the quality of service question in public networks, the dilemma lies between adding capacity (which will rarely be sufficient in all circumstances) and introducing complex prioritisation or multi-path routing schemes. Both increase the size and cost of the devices, just to provide cover against unlikely eventualities, which may never happen[18]. But catastrophes are exactly the time when reliable data is most required.

As messages cross the Internet at large, these problems can be ignored if the 'data rate from reality' is dwarfed by the data rate between entities in the information world. The latter is more elastic, so it can hold back to allow for fluctuations and avalanches of data from reality.

Delivery reliability Having introduced the possibility of dropping messages (whether as a result of congestion or wireless channel fade), this leads us directly into a further challenge to conventional thinking: how does a receiver know a message has been lost when it wasn't expecting it anyway?

Usually a receiver can detect a missing message when a sequence of packets stops arriving, or the next in the sequence arrives revealing a gap. Asynchronous events are not expected by definition. So the receiver can only detect a loss when the next event arrives, which may be anything from microseconds to years later. If using positive acknowledgement (ack) this is not a problem, because the sender can detect a lack of ack, and re-send. But if a large number of listeners want to be informed of each event, the source can be overwhelmed by what

[16]Our current research solves these 'slow-start' problems by reversing the incentive structures of the Internet's congestion control mechanisms. A publication is in preparation.

[17]In the physical world, this problem is solved by transmitting information in analogue form. For instance, an arbitrarily large volume of information can pass through the lens of the eye and the retina, in order to be prioritised during processing by the brain.

[18]Aggregation of data within a sensor network (§3.3) might alleviate most potential congestion problems.

is termed an 'ack implosion'. Therefore, for scalability, the pub-sub model is deliberately arranged to hide the comings and goings of listeners from the source. So how can the source know how many listeners are interested, and therefore how many acks it should have received?

Negative acknowledgement (nack), where receivers only complain if they miss a message, is preferred in multicast streaming scenarios. So nacks can be aggregated on their way to the source if more than one receiver misses a message (Cisco provides the pragmatic general multicast (PGM) capability [53] on their routers to aggregate nacks, which is a specific type of concasting within their generic router assist framework discussed in §3.3). This still leaves the above problems where no-one can nack a missed message that they weren't expecting.

Our own index-based event messaging system (introduced in §§3.2 & §3.4 in the context of intermittent links and address aggregation) was also designed to solve this problem. Because its indexes beacon regularly, it is clear when one is missed. And the indexes list the version number of the latest message on each message channel. So receivers can nack a missed message, even being able to catch up if they have been disconnected for some time.

Hop-by-hop ack or nack handling *can* be used for reliable delivery. PGM aggregates nacks [53], while SCRIBE [19] uses the acks of a long-running TCP session between each intermediate system. However, an early insight during the design of the Internet was that hop-by-hop acknowledgement didn't remove the need for end-to-end acknowledgement (e.g. during re-routing or router failures) [18]. The situation is no different for asynchronous events. We have to check delivery end-to-end, so hop by hop is redundant, hence justifying our choice of beaconing indexes.

The above concerns reliable multicast delivery across global networks, originating from a sensor just one hop from mains power as in our straw man proposal (§3.1). When messages arise in a *multi*-hop sensor network, we have already (§3.3) recommended avoiding duplication of power consumption, aggregating with concast, rather than duplicating with multicast until a node with mains power is reached. How reliability mechanisms should most efficiently span the two parts of the message transport in these cases is a matter for further research.

3.6 Security

All distributed applications require some degree of trust between the devices comprising the system. Elsewhere, Seleznyov *et al* discuss an approach to allow pervasive devices to determine whether they hold sufficient trust in their correspondents for the risk associated with the action in hand [54]. Such systems require rudimentary primitives for secure communications.

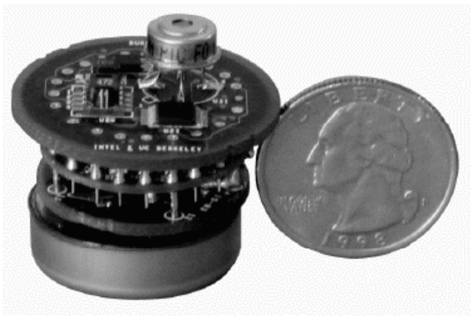
Figure 12: A Berkeley mote, generic sensor nodes manufactured by Crossbow.

Pervasive placement of computers in the environment breaks many traditional communications security assumptions. Most importantly, storage of keys on a device assumes the device is physically secured against those who don't know the key, which is not the case for many sensor networking applications spread liberally throughout the environment.

Verifiable location However, many other applications of pervasive computing involve devices within business or residential premises. Recent research at UC Berkeley has introduced a technique to establish whether a device is located where it claims to be [55], although a trusted node close to the device is a prerequisite. Techniques that don't rely on a nearby trusted node are far more complex, as described in Gabber's excellent survey of the field [56]. If it can be established that a device is within, rather than outside, a physically secured building, immediately more trust can be placed in messages from that device. Also, it may be possible to authenticate that a device is what it says it is, because it is located where it is meant to be.

Tamper-resistance and challenged hardware If security-sensitive devices must be subject to general access, it may be sufficient to use mildly tamper-resistant casings (as in our straw man proposal in §3.1), as long as the limitations are understood [57]. The sheer numbers of devices can be used to offer sufficient protection by designing the whole system to be resistant to compromise of a minority of devices. SPINS [58] is an example of this approach, which has been implemented on Berkeley motes (Fig 12). It consists of two message security primitives: SNEP for confidentiality and μTESLA for message authenticity and integrity. Both are

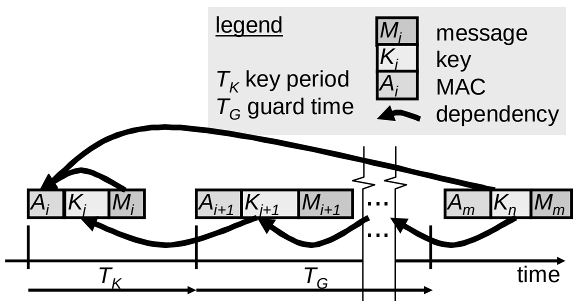

Figure 13: TESLA source authentication for group communications. A key and a message authentication code (MAC) accompany each message. But they both depend on future keys not known to receivers. In advance the source generates a sequence of keys by repeating a simple one-way function. Each key period it reveals a new key back along the sequence. But, for the MAC, it uses a key it will reveal later, thus committing to a key it guarantees no-one else will know for at least the guard-time (typically 500ms). As the source reveals each key commitment, every receiver can be sure earlier messages were authentic, giving slightly delayed but strong authentication for challenged hardware.

designed to provide strong protection despite extremely limited hardware, ruling out the exponentiation required for asymmetric cryptography. Further, each data source needs to authenticate its communications to a large group of receivers, ruling out the use of shared keys too. So μTESLA[19] derives a new form of asymmetry for its keys from the one-way passage of time (Fig 13). In SPINS, these primitives are built up into a secure sensor network system bootstrapped from an association with a base-station.

Key management Public key (asymmetric) cryptography requires considerable computing resources, so it is considered infeasible on miniature devices for the foreseeable future. Even on high-speed computers, asymmetric keys tend to only be used to bootstrap lighter weight symmetric cryptography by exchange of a symmetric key. Without the ability to use public keys, it is hard to bootstrap a secure system of miniature devices by authenticating that the source of the initial keys is trusted. This is why SPINS (above) cannot bootstrap its authentication with any trusted identity, being limited to having to trust the base-station owner, which it authenticates by locality.

[19]μTESLA is based on TESLA, our own joint work with Carnegie Mellon Uni, UC Berkeley and IBM for lightweight, per-packet authentication of broadcast multimedia streams. It is currently in the process of standardisation through the IETF [59].

Stajano and Anderson [22] propose that a useful key establishment primitive for miniature devices would be physical touch. That is, a master key can be established in a device when it is placed in contact with a mother device. This key cannot then be overridden by the touch of another device. Only the original mother can release the device from its mother's binding, after which it can be bound to the first new mother that touches it. The reliance on touch can be relaxed to include proximity [60]. Chan *et al* [61] establishes keys by giving each of a set of sensors a different subset of a large set of random keys prior to deployment. It uses the increased probability of any two nodes sharing common keys to bootstrap key establishment.

Once initial keys are established, key management procedures must be used to refresh the keys actually employed for message cryptography [62], reducing the time attackers have to guess the keys in use by brute-force methods (as in our straw man proposal in §3.1). However, when broadcasting secrets to a constrained group, traditional key management techniques developed for one-to-one communications cannot be used, because the authorised membership of the group may change over time.

One approach is to assume the presence of a stable set of trusted event mediator nodes (see §2.3). Then messages are sent to these mediators and group members form a one-to-one security association with any one of these mediators. However, trust in stable intermediaries is neither generic nor necessary in pervasive computing scenarios (or across the Internet at large). A preferred approach is for the message to be encrypted with a group key (using that classic oxymoron, the secret shared over a large group) at source, and remain encrypted throughout its end-to-end journey, at least avoiding sharing the key unnecessarily with intermediate nodes. As end-points join and leave the group, the group key is changed. Moyer *et al* provides a useful overview of the issues in group key management and a survey of solutions at the time [63].

3.7 Ancillary communications services

Throughout the paper so far, we have kept our focus on the *rudimentary* functions necessary for communications to realise a pervasive computing vision. Beyond these, a richer set of ancillary communications services may be necessary for each specific application: search and discovery, group forming, session co-ordination, certification, transactional messaging, causal message ordering, and so on.

There are two schools of thought on how to achieve all but the most rudimentary function of forwarding and routing:

bundled — functions embedded within routers and relays throughout the messaging network (e.g. the event mediation service[20]),

unbundled — communications services separately provided by third parties (e.g. event archives) or by the joint efforts of peer end-points (e.g. authentication and confidentiality).

Both approaches are distributed, they merely differ in whether the act of relaying is *unavoidably* bundled with other functions or not. In the unbundled approach, relays act as common carriers, irrespective of the messages being passed.

Our instincts are to always try to design unbundled functions (which can then *optionally* be collocated with the relaying function). But each case must be taken on its merits, and reasoned justifications given, rather than following the end-to-end design principle as a religion. We have given these reasoned arguments throughout.

But once we move from rudimentary to higher level functions as listed above, there is no question that they are best designed so that they can be separated from the basic networking service. In this respect, they have less impact on network design, justifying ruling their detailed discussion out of this paper's scope.

4 Business implications

Trends Over the last couple of decades asynchronous programming has become prevalent, at least for non-networked code local to the machine's own bus (e.g. an event listener is set for a mouse click event and, when it triggers, the code checks where the mouse pointer is and calls the relevant function). This trend will extend to distributed programming, though initially through the use of intermediary message servers rather than direct peer-to-peer remote procedure calls. As computing becomes more interfaced to the wider physical world, we can expect distributed asynchronous programming to become commonplace. Crowcroft envisages 10^{10} event messages per second world-wide [46]. We have already explained that, if we expect trillions of devices, there will be orders of magnitude more event sources and listeners (§3.4).

[20]Which may itself be an overlay network built on more rudimentary primitives, but it consists of message relays itself.

This warns us that the potential growth of pervasive computing will be dependent on the successful deployment of the pub-sub mode of group communications, whether directly in the network infrastructure (e.g. as IP multicast [16]) or using overlays [19, 64].

The value of group forming Metcalfe's Law predicts that the value of a network is $O(n^2)$, because each of n users has the potential to communicate with $n-1$ other users. Reed's Law goes beyond this, pointing out that n users[21] have the potential to create 2^n different groups between themselves [65], making the value of a network that supports formation of groups rise exponentially $O(2^n)$. Although neither law is intended to be a guide to prediction of actual market size, they give a feel for what shape the market growth curve should take in each case. They certainly show that a business that allows unfettered creation of groups ('Yahoo Groups for machines') will tend to be popular. If growth in computing device numbers remains exponential (the predicted doubling time is about 0.6 years), then growth in group numbers will be doubly exponential, that is $O(2^{2^t})$. Or put another way, if we are to create thousands of trillions of groups by 2020, world-wide group creation facilities will have to have been capable of creating millions of groups every second.

Whose business? Which customer? If devices are spread throughout the fabric of our lives, in shared office blocks, in industrial units, in public places and in private dwellings, up to a point, networks of devices will be able to act as their own infrastructure. But part of the capacity of global networking infrastructure and all the capacity of ancillary services dedicated to asynchronous messaging will also be required. Why would anyone invest in this infrastructure? For the public good? For private gain? What socio-economic mechanisms will cause these prerequisites of the pervasive computing vision to happen?[22]

Because the impact of each device is so small, the question of who is responsible for its share of infrastructure becomes blurred. Many assume that infrastructure operators will cover their costs by cross-subsidising infrastructure from higher level services. But it is unrealistic to expect any company to be successful enough at these very different businesses for one to sustain the other, given competition from specialists in services for pervasive

[21]Or processes?

[22]The wider question of who will have the incentive to invest in deployment of the devices themselves is just as valid, but outside the scope of this paper.

computing. So, if a washing machine detects new fabrics in its wash-load and looks up their washing instructions over the Internet, who should arrange for the required infrastructure and its maintenance? The manufacturer? the retailer? the householder? Who should pay for the service engineer to come out to a faulty washing machine only to discover the fault is due to the householder not paying their Internet bill? If sensor A is measuring light levels for one application, but also relaying for sensor B, they may both need global connectivity, but should relay A charge sensor B for its services?

Clearly, there will be many informal acts of give and take over what will often be trivial issues. But many tiny acts of take without any give mount up. Solutions to these free-rider problems must be found if the vision of pervasive computing is to be realised.[23]

Pricing Assuming *someone* (probably not the end-user) will take responsibility for paying the bills for the communications infrastructure necessary to support pervasive computing, how will it be priced? The casual observer would expect event messaging services to be flat charged (irrespective of usage). Usage charging[24] would incentivise responsible use of resources, but we have seen that miniature devices already have good reason to be parsimonious, particularly to conserve battery power.

Our intuition is that flat charging will prevail wherever pervasive computing is 'scavenging' resources provided for other purposes (e.g. the wider Internet). But where messaging services are provided specifically, the potential sheer volume of messaging will tend to force the introduction of usage charging.

We have also explained that many applications will be part of vital business processes, their communications requiring priority over more optional messages during surges in demand. Clearly, priority service will attract premium charging. Given priority will rarely have to be exercised, premiums are unlikely to be levied only when priority is needed (usage). Rather they will be paid regularly in advance for the right to priority as a contingency (flat).

Pub-sub business model Traditionally, multicast has been associated with efficient distribution of streamed media, but it is also suitable for the timely delivery of asynchronous events. The sessions are just considerably shorter. One of the reasons pub-sub has not been widely deployed for media distribution is the desire to re-create content distribution business models over it that mimic those used today, by both the content and the network industry. Specifically, they expect the distribution network to charge the sender proportionately to the number of receivers.

Therefore, the full potential of pervasive computing will not be realised unless we can establish a viable separation between the business models for media distribution and for event notification. It is clearly not feasible to charge for event notification messaging per session (i.e. per event), but, if pub-sub were charged flat rate, it would be hard to prevent the same mechanisms being used to bypass media distribution pricing.

To bundle or to unbundle? When we discussed function placement (§2.4), we recommended that functions should not be assigned to specific equipment. Rather, we advised that the system should be designed open. Then, it is still perfectly possible to bundle functions together into a piece of equipment (e.g. if a vendor believes the business model is advantageous). But bundling is *chosen*, not inherent to the technical design.

Throughout, we have shown that there are rarely technical reasons to lock in even rudimentary communications services with the business of operating a dumb pub-sub network.[25] But, despite having no inherent lock-in, we can predict that messaging services will make good business for network operators in the next few years. However, as the pervasive computing scenario develops, we predict that all, or at least most of these services will be self-provided more competitively by distributed software across the pervasive computing base. Therefore the technology should be designed for this transition from closed to open from the outset ("Design for Tussle" [66]), justifying our intuition of an open design closed by policy choice.

5 Conclusions

We have given controversial but reasoned explanations for why most pervasive computing will largely be connected globally and for why it will rely heavily on models of the physical world within the information world. It is infeasible to imagine that any arbitrary pair of computing devices will be able to talk directly to each other, just because they are in physical proximity. Instead physical devices will be bound to information models of themselves,

[23] See `http://www.mmapps.org/` for our collaborative research on motivations in peer-to-peer communities.

[24] In bulk — it goes without saying that itemisation is out of the question.

[25] Message aggregation and duplication — concast and multicast — being exceptions.

which will meet in cyberspace, communications between devices taking place across cyberspace, as often as by direct wireless connectivity. Continuous co-ordination between the physical and information worlds will lead to a web of tens of billions of messages per second across the world's networks.

We have explained why we predict that asynchronous event messaging using publish-subscribe will be the predominant mode of communications for pervasive computing. Pub-sub will mostly be based on the multicast mode (whether native or overlay), once messages from the physical world have hit the first gateway to the rest of the Internet (the first connection to mains power). But, if there is a multi-hop wireless network between the physical world and that first gateway, in order to conserve power, concast will be the most likely communications mode up to that gateway.

We have questioned the assumption that tiny cheap computing devices will be considered disposable and application-specific. Economies of scale in volume device manufacturing will always favour reuse of the investment in generic device *designs*, whether each unit is disposable or not. Also, the investment in *deploying* devices will be higher than their cost, tending to encourage new ways to combine deployed devices to novel ends. Therefore working out the most generic communications facilities for miniature devices is a central part of the research agenda.

Having set out the grand challenge, we have surveyed its implications on each rudimentary element of communications: routing, addressing, congestion control, reliable delivery, security, etc. We have described the problems precisely and the various main approaches being adopted by the research community to solve them, giving rationale. We have exploded some myths particularly explaining why it is often more effective and less complex to create an x-like system from un-x-like parts (e.g. a reliable system from unreliable parts, or a secure system from insecure parts). We have also corrected some misunderstandings that the research community seems to have carried over from their previous research areas. For instance, it is fruitless to try to control congestion by varying the data rate of reality and it is fruitless to expect receivers to know when they have missed an asynchronous message, which by definition they weren't expecting anyway.

Finally, we have introduced the implications of pervasive computing on the *business* of communications, explaining the trouble the commercial community has with the pub-sub model and emphasising the importance of group creation services. Many open questions remain, particularly concerning how to promote investment in pervasive computing infrastructure, given the benefits are so thinly spread, making collection of return on investment costly, even with flat pricing models. We have explained why 'open by design but closed by policy choice' will still be the appropriate commercial approach, given communications *designs* will still need to be generic, even though the *devices* on which they are implemented will be disposable.

Acknowledgements

Jon Crowcroft (Cambridge Uni), Jane Tateson, Trevor Burbridge, Andrea Soppera, Alan Steventon (BT Research) and the anonymous reviewers.

References

[1] Weiser M: 'The computer for the 21st Century', Scientific American, 265, No 3, pp 94–104 (September 1991)

[2] Durand A, Huitema C: 'The host-density ratio for address assignment efficiency: An update on the H ratio', Request for comments 3194, Internet Engineering Task Force, URL: rfc3194.txt (November 2001)

[3] Bulman J, Crabtree B, Gower A, Oldroyd A, Lawson M, Stutton J: 'Mixed reality applications in urban environments', BT Technology Journal, 22, No 3 (July 2004)

[4] Brown S, Garner P, Sixsmith A, Hine N: 'Care in the community', BT Technology Journal, 22, No 3 (July 2004)

[5] Bull P, Payne R: 'Pervasive home environments', BT Technology Journal, 22, No 3 (July 2004)

[6] Bilchev G: 'Traffimatics — A pervasive view of co-operative vehicle highway systems', BT Technology Journal, 22, No 3 (July 2004)

[7] Gershenfeld N, Krikorian R: 'Building intelligence'. Web page URL: http://www.media.mit.edu/physics/projects/IP/bldg/bi/ (June 2002) Briefing note.

[8] Krikorian R: 'Internet 0; Bringing IP to the leaf node'. URL: http://www.bitwaste.com/wasted-bits/blog/conferences/et2003/I0-oreilly.pdf (April 2003) (Presentation).

[9] Yokoji S, Takahashi K, Miura N: 'Kokono search: A location based search engine', In: Proc. Tenth International World Wide Web Conference (WWW10), (May 2001) URL: http://www10.org/cdrom/posters/contents.html#a5 (Poster).

[10] Buyukkokten O, Cho J, Garcia-Molina H, Gravano L, Shivakumar N: 'Exploiting geographical location information of web pages', In: (Informal) Proc. Workshop on Web Databases (WebDB'99), ACM (June 1999) URL: http://citeseer.ist.psu.edu/buyukkokten99exploiting.html

[11] Shrikumar H: 'IPic - A match head sized Web-server'. Web page URL: http://www-ccs.cs.umass.edu/shri/iPic.html (October 2002)

[12] Shrikumar H: 'Connecting absolutely everything', White Paper WP010308, Ipsil, URL: http://www.ipsil.com/resources/whitepapers/connecting.pdf (2001)

[13] Oki B, Pfluegl M, Siegel A, Skeen D: 'The Information Bus; an architecture for extensible distributed systems', In: Proc. 14th ACM Symposium on Operating Systems Principles, (1993) URL: http://www.cs.utah.edu/~retrac/cs7460/infobus.pdf

[14] Intanagonwiwat C, Govindan R, Estrin D: 'Directed diffusion: A scalable and robust communication paradigm for sensor networks', In: Proc. ACM International Conference on Mobile Computing and Networks (MobiCOM'00), ACM (August 2000) URL: http://www.isi.edu/scadds/projects/diffusion.html#publications

[15] Soppera A, Burbridge T, Briscoe B, Rizzo M: 'GAP: The generic announcement protocol for event messaging', In: Proc. London Communication Symposium (LCS'03), UCL (September 2003) URL: http://www.ee.ucl.ac.uk/lcs/papers2003/103.pdf

[16] Deering S E: 'Multicast routing in internetworks and extended LANs', In: Proc. ACM SIGCOMM'88. (August 1988)

[17] Bacon J, Moody K, Bates J, Hayton R, Ma C, McNeil A, Seidel O, Spiteri M: 'Generic support for distributed applications', IEEE Computer, pp 68–76 (March 2000) URL: http://www.cl.cam.ac.uk/Research/SRG/opera/publications/pre-2002.html#2000

[18] Saltzer J H, Reed D P, Clark D D: 'End-to-end arguments in system design', ACM Transactions on Computer Systems, 2, No 4, pp 277–288 (November 1984) URL: http://web.mit.edu/Saltzer/www/publications/endtoend/endtoend.pdf An earlier version appeared in the Second International Conference on Distributed Computing Systems (April, 1981) pages 509–512.

[19] Rowstron A, Kermarrec A M, Castro M, Druschel P: 'SCRIBE: The design of a large-scale event notification infrastructure', In Crowcroft J, Hofmann M, eds.: Proc. 3rd International COST264 Workshop on Networked Group Communication (NGC'01). Volume 2233 of LNCS., Springer LNCS pp 30–43 (November 2001) URL: http://link.springer.de/link/service/series/0558/bibs/2233/22330030.htm

[20] Soppera A, Burbridge T: 'Maintaining privacy in pervasive computing', BT Technology Journal, 22, No 3 (July 2004)

[21] Heidemann J, Silva F, Intanagonwiwat C, Govindan R, Estrin D, Ganesan D: 'Building efficient wireless sensor networks with low-level naming', In: Proc. Symposium on Operating Systems Principles, ACM pp 146–159 (October 2001) URL: http://www.isi.edu/~johnh/PAPERS/Heidemann01c.html

[22] Stajano F, Anderson R: 'The resurrecting duckling: Security issues for ad-hoc wireless networks', In Christianson B, Crispo B, Roe M, eds.: LNCS 1796, Proc. 7th International Workshop on Security Protocols, Springer Verlag pp 172–194 (April 1999) URL: http://citeseer.ist.psu.edu/227971.html

[23] Briscoe B: ‘MARKS: Zero side effect multicast key management using arbitrarily revealed key sequences’, In: LNCS 1736, Proc. 1st International COST264 Workshop on Networked Group Communication (NGC’99), Springer (November 1999) URL: http://www.btexact.com/publications/papers?doc=70250

[24] Floyd S, Jacobson V, Liu C G, McCanne S, Zhang L: ‘A reliable multicast framework for light-weight sessions and application level framing’, Proc. ACM SIGCOMM’95, Computer Communication Review, 25, No 4, pp 342–356 (October 1995) URL: http://www.acm.org/sigcomm/ccr/archive/1995/conf/floyd.html

[25] Fall K: ‘Delay tolerant networking research group’, Working group charter, Internet Research Task Force, URL: http://www.dtnrg.org/ (2002) (Continuously updated).

[26] Fall K: ‘A delay-tolerant network architecture for challenged internets’, Proc. ACM SIGCOMM’03, Computer Communication Review, 33, No 4, pp 27–34 (October 2003) URL: http://portal.acm.org/citation.cfm?id=863960&dl=ACM&coll=portal

[27] Kahn J M, Katz R H, Pister K S: ‘Emerging challenges: Mobile networking for ‘smart dust”, Journal of Communications and Networks, 2, No 3, pp 188–196 (September 2000) URL: http://citeseer.ist.psu.edu/kahn00emerging.html

[28] Zhou L, Kahn J M, Pister K S: ‘Corner-cube retroreflectors based on structure-assisted assembly for free-space optical communication’, IEEE Journal of Microelectromechanical Systems, 12, No 3, pp 233–242 (June 2003) URL: http://www-ee.stanford.edu/~jmk/pubs/ccr.jmems.pdf

[29] Chlamtac I, Petrioli C, Redi J: ‘Energy conservation in access protocols for mobile computing and communication’, Microprocessors and Microsystems Journal, 1, pp 20–32 (March 1998)

[30] Chockalingam A, Zorzi M: ‘Energy efficiency of media access protocols for mobile data networks’, IEEE Transactions on Communications, 46, No 11, pp 1418–1421 (November 1998) URL: http://citeseer.nj.nec.com/259740.html

[31] Kravets R, Krishnan P: ‘Application-driven power management for mobile communication’, Wireless Networks, 6, pp 263–277 (2000) URL: http://citeseer.nj.nec.com/kravets98applicationdriven.html

[32] Soppera A, Burbridge T, Nekovee M: ‘Index-based event messaging’, In: Proc. International Workshop on Large-Scale Group Communication, (October 2003) URL: http://srds2003.cnuce.cnr.it/papers/soppera.pdf

[33] Tschudin C, Lundgren H, Nordström E: ‘Embedding MANETs in the real world’, In: Proc. Conference on Personal Wireless Communications (TWC’03), IFIP (September 2003) URL: http://www.it.uu.se/research/group/core/publications.php?pub_id=41

[34] Perkins C, Belding-Royer E, Das S: ‘Ad hoc on-demand distance vector (AODV) routing’, Request for comments 3561, Internet Engineering Task Force, URL: http://www.tcs.hut.fi/~anttit/manet/aodv/ (July 2003) (Experimental).

[35] Baker F: ‘An outsider’s view of MANET’, Internet Draft draft-baker-manet-review-01, Internet Engineering Task Force, URL: http://bgp.potaroo.net/ietf/old-ids/draft-baker-manet-review-01.txt (March 2002) (Expired).

[36] Singh S, Woo M, Raghavendra C S: ‘Power-aware routing in mobile ad hoc networks’, Proc. ACM International Conference on Mobile Computing and Networks (MobiCOM’98), pp 181–190 (1998) URL: http://citeseer.nj.nec.com/singh98poweraware.html

[37] Chang J H, Tassiulas L: ‘Energy conserving routing in wireless ad-hoc networks’, In: Proc. IEEE Conference on Computer Communications, pp 22–31 (2000) URL: http://citeseer.nj.nec.com/chang00energy.html

[38] Madden S R, Szewczyk R, Franklin M J, Culler D: ‘Supporting aggregate queries over ad-hoc wireless sensor networks’, In: Proc. Workshop on Mobile Computing and Systems Applications, (2002) URL: http://www.cs.berkeley.edu/~madden/madden_aggregation.pdf

[39] Cain B, Speakman T, Towsley D: ‘Generic router assist (GRA) building block; motivation and architecture’, Internet draft, Internet Engineering Task Force, URL: http://bgp.potaroo.net/ietf/old-ids/draft-ietf-rmt-gra-arch-01.txt (March 2000) (Expired Sep 2000) (also invited talk at NGC’99).

[40] Duros E, Dabbous W, Izumiyama H, Fujii N, Zhang Y: ‘A link-layer tunneling mechanism for unidirectional links’, Request for comments 3077, Internet Engineering Task Force, URL: rfc3077.txt (March 2001)

[41] Perlman R: ‘Network layer protocols with byzantine robustness’, Technical Report 429, MIT, URL: http://www.lcs.mit.edu/publications/specpub.php?id=997 (October 1988) (Based on PhD thesis).

[42] Zhu D, Gritter M, Cheriton D R: ‘Feedback based routing’, Computer Communication Review, 33, No 1, pp 71–76 (January 2003) URL: http://portal.acm.org/citation.cfm?id=774774&dl=ACM&coll=portal

[43] Law Y, Dulman S, Etalle S, Havinga P: ‘Assessing security-critical energy-efficient sensor networks’, In Gritzalis D, Vimercati S D C D, Samarati P, Katsikas S, eds.: Proc. 18th TC11 Int. Conf. On Information Security, IFIP, Kluwer pp 459–463 (May 2003) URL: http://www.ub.utwente.nl/webdocs/ctit/1/00000087.pdf

[44] Adler M, Ge Z, Kurose J, Towsley D, Zabele S: 'Channelization problem in large scale data dissemination', In: Proc. IEEE International Conference on Network Protocols (ICNP'01), (2001) URL: http://www.cs.umass.edu/~micah/pubs/channelization.ps

[45] Thaler D, Handley M: 'On the aggregatability of multicast forwarding state', In: Proc. IEEE Conference on Computer Communications (Infocom 2000), (March 2000) URL: http://www.ieee-infocom.org/2000/papers/632.ps

[46] Crowcroft J, Bacon J, Pietzuch P, Coulouris G, Naguib H: 'Channel islands in a reflective ocean: Large scale event distribution in heterogeneous networks', IEEE Communications Magazine, 40, No 9, pp 112–115 (September 2002) URL: http://www.cl.cam.ac.uk/Research/SRG/opera/publications/2002-pubs.html

[47] Kulik J: 'Fast and flexible forwarding for Internet subscription systems', In: Proc 2nd Intl Workshop on Distributed Event-Based Systems, ACM press pp 1–8 (June 2003) URL: http://doi.acm.org/10.1145/966618.966635

[48] Crowcroft J, Hand S, Mortier R, Roscoe T, Warfield A: 'Plutarch: An argument for network pluralism', In: SIGCOMM FDNA'03, ACM (2003) URL: http://www.cl.cam.ac.uk/~jac22/out/plutarch.pdf

[49] Brownlee N, Ma A: 'NeTraMet flow lifetimes and implications for routing context'. Web page URL: http://www.caida.org/analysis/workload/netramet/lifetimes/ (June 2002) (Data in and out of UC San Diego Supercomputer Center collected Jun 2000).

[50] Beverly R, claffy k: 'Wide-area IP multicast traffic characterization', IEEE Network, (January/February 2003) URL: http://www.caida.org/outreach/papers/2003/mcastworkchar/

[51] Crovella M, Bestavros A: 'Self-similarity in World Wide Web traffic: Evidence and possible causes', IEEE/ACM Transactions on Networking, 5, No 6, pp 835–846 (1997) URL: http://citeseer.ist.psu.edu/22080.html

[52] Wan C Y, Eisenman S B, Campbell A T: 'CODA: Congestion detection and avoidance in sensor networks', In: Proc. First International Conference on Embedded Networked Sensor Systems, ACM pp 266–279 (2003) URL: http://comet.ctr.columbia.edu/armstrong/

[53] Speakman T, Farinacci D, Lin S, Tweedly A, Bhaskar N, Edmonstone R, Johnson K M, Sumanasekera R, Vicisano L, Crowcroft J, Gemmell J, Leshchiner D, Luby M, Montgomery T, Rizzo L: 'PGM reliable transport protocol specification', Request for comments 3208, Internet Engineering Task Force, URL: rfc3208.txt (December 2001)

[54] Seleznyov A, Ahmed M O, Hailes S: 'Cooperation in the digital age: Engendering trust in electronic environments', BT Technology Journal, 22, No 3 (July 2004)

[55] Sastry N, Shankar U, Wagner D: 'Secure verification of location claims', In: Proc. Workshop on Wireless Security (WiSe 2003), ACM (September 2003) URL: http://www.cs.berkeley.edu/~nks/locprove/

[56] Gabber E, Wool A: 'On location-restricted services', IEEE Network, 13, No 6, pp 44–52 (1999)

[57] Anderson R, Kuhn M G: 'Tamper resistance — A cautionary note', In: Proc. Second USENIX Electronic Commerce Workshop, pp 1–21 (November 1996) URL: http://www.cl.cam.ac.uk/users/rja14/#Reliability

[58] Perrig A, Szewczyk R, Wen V, Culler D E, Tygar J D: 'SPINS: Security protocols for sensor networks', In: Proc. ACM International Conference on Mobile Computing and Networks (Mobicom'01), pp 189–199 (July 2001) URL: http://citeseer.ist.psu.edu/568886.html

[59] Perrig A, Canetti R, Song D, Tygar D, Briscoe B: 'TESLA: Multicast source authentication transform introduction', Internet draft, Internet Engineering Task Force, URL: draft-ietf-msec-tesla-intro-03.txt (August 2004) (Work in progress).

[60] Juels A: '"Yoking-proofs" for RFID tags', In Sandhu R, Thomas R, eds.: Proc. First International Workshop on Pervasive Computing and Communication Security, IEEE Press (2004) URL: http://www.rsasecurity.com/rsalabs/staff/bios/ajuels/publications/rfidyoke/

[61] Chan H, Perrig A, Song D: 'Random key predistribution schemes for sensor networks', In: Proc. Symposium on Security and Privacy 2003, IEEE (2003) URL: http://www.ece.cmu.edu/~adrian/publications.html

[62] Basagni S, Herrin K, Bruschi D, Rosti E: 'Secure pebblenet', In: Proc. International Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc'01), ACM pp 156–163 (October 2001) URL: http://www.ece.neu.edu/faculty/basagni/publications.html

[63] Moyer M J, Rao J R, Rohatgi P: 'A survey of multicast security issues in multicast communications', IEEE Network, 13, No 6, pp 12–23 (1999) http://www.comsoc.org/

[64] Ratnasamy S, Handley M, Karp R, Shenker S: 'Application-level multicast using content-addressable networks', In: Proc. 3rd International COST264 Workshop on Networked Group Communication (NGC'01). Volume 2233., Springer LNCS pp 14– (November 2001) URL: http://link.springer.de/link/service/series/0558/bibs/2233/22330014.htm

[65] Reed D P: 'That sneaky exponential — Beyond Metcalfe's Law to the power of community building'. Web Page URL: http://www.reed.com/Papers/GFN/reedslaw.html (1999) Online Supplement to Article in Context magazine.

[66] Clark D, Sollins K, Wroclawski J, Braden R: 'Tussle in cyberspace: Defining tomorrow's Internet', Proc. ACM SIGCOMM'02, Computer Communication Review, 32, No 4 (August 2002) URL: http://www.acm.org/sigcomm/sigcomm2002/papers/tussle.pdf

Document history

Version	Date	Author	Details of change
A	04 Dec 2003	Bob Briscoe	Outline Draft
B	12 Dec 2003	Bob Briscoe	Updated outline draft based on reviewers comments
C	03 May 2004	Bob Briscoe	Completed ready for review.
1	06 Jun 2004	Bob Briscoe	Modified following reviewer comments.
2	18 Jun 2004	Bob Briscoe	First issue sent to publisher.
3	18 Jul 2004	Bob Briscoe	Minor corrections for BTTJ publication.
4	20 Nov 2004	Bob Briscoe	First issue for iSpaces book chapter.
5	20 Nov 2004	Bob Briscoe	Minor corrections following ASteventon review