Internet 0 — inter-device internetworking

R Krikorian and N Gershenfeld

The assumptions behind Internet architectures do not scale to small devices — they have a baseline cost that is still too high for small, few-dollar, embedded objects. This barrier either leaves many devices network disenfranchised or encourages the creation of segmented networks. Internet 0 attempts to enable pervasive computing and networking on the embedded level by providing the Internet protocol as a communications substrate, and, through the use of an end-to-end modulation scheme, to speak to devices. I0 is a framework to bridge together heterogeneous devices via IP in a manner that is still compatible with designing globally large computer networks.

1. Introduction

While the Internet has introduced new networking technologies and communications methods, its reach has not extended far below fairly computationally intensive devices [1]. Most think the smallest computational unit that is Internet-capable are those running full-blown operating systems (Windows, Mac OS X, Free DOS, Unix-compatible, etc) and are equipped with high-speed networking hardware (Ethernet, IEE802.11, etc). Unfortunately, this leaves a disfranchised group of devices — those which are too small, too embedded, or thought to be too ‘simple’. This paper introduces Internet 0, a framework to bring networking to that forgotten class.

John Romkey is famous for creating the first Internet ‘device’, a toaster that could be turned on and off over the Internet, in 1990. To make that happen, Romkey spliced a power relay to his toaster and connected that to the parallel port of his network connected laptop. Unfortunately, this example illustrates exactly why nothing has changed in the last decade. An engineer asked to put an incandescent light on to the Internet in the present may do the very same thing. He or she may simply splice a relay on to the AC power line running to the light bulb, connect the relay’s switching line to a pin on the parallel port of the computer, then author a simple CGI script that takes input from an HTML button and then outputs state to that pin.

Taking this particular example a bit further — consider creating a light switch somewhere else in the building (or even in another) that controls this light. Using the set-up described above as the light bulb, a light switch simply needs to be the embodiment of a Web client. An engineer can attach a light switch to another pin on the parallel port (this time an input one), author software that monitors that pin, and when the pin state changes issue a Web request to the computer with the light bulb.

I0 networks are based around seven basic tenets

While this light bulb and switch is a functioning solution for network-enabled control, it underscores how the Internet has failed at ‘interdevice internetworking’. The above example requires two fairly expensive network-enabled desktop-class computers (at a cost on the order of $1000 each) and each has to be individually configured to know how to make or receive requests from the other. The demonstration also assumes that both locations have either Ethernet or IEE802.11 and an IP address issuing authority (either computational as a DHCP server, or personal via a network administrator) — both of which take a qualified IT department to set up and maintain. And finally, there is more than just the power consumption issues of the light bulb and switch themselves, but now also the watts of power required to power the computer’s CPU and hard drives.

I0 attempts to solve those issues by defining a framework that not only enables small devices to inter-communicate with each other, but also inter-operate with any other machine, no matter how large or small. I0 networks are based around seven basic tenets:

- bringing the Internet protocol all the way to the device level to make devices full network citizens,
- compiling standards and delayering network protocol stacks to make them computationally efficient enough to fit into embedded microprocessors,
- allowing devices to talk to each other directly to remove the necessity of centralised servers and protocol converters,
- advertising a device not only to the virtual network, but also to the physical one to allow direct interactions between objects and objects, and also objects and people,
- slowing down networks to decrease network complexity and therefore simplify network access,
- using the same modulation scheme across many different media so that device designers can be free to choose their preferred hardware medium while not being isolated from devices that use another,
- pushing the engineering politics of open standards to inspire competition not in differing architectures, but in differing network services.

There are many other standards which aim to do the same thing — everything from CAN bus for the transportation industry, LonWorks for building automation, and sensor networks, I2C for inter-IC communication, etc. Each one of these embody portions of the seven listed above, but none, except I0, has all characteristics. The Internet has scaled through orders of magnitude in both size and speed while the device network standards have yet to attempt global naming and routing.

2. IP to the leaf node

Historically, engineers have stayed away from using native Internet protocols, in all portions of an embedded device network or system, out of fear that the implementation of the stack was too difficult or that the use of IP was inefficient. This mentality has unfortunately left device networks in a myriad of different protocols — almost resembling the disparate computing networks in the early days of the Internet. To rectify computer networking, the Internet protocol [2] was introduced as the lowest common denominator — any computer that wishes to join an IP network simply needs to prepend all data it transmits with a 20 byte IP header, and any network technology simply needs to carry packets as small as 576 bytes. This type of abstraction allows for any computer to talk to any other computer on the Internet without knowledge of the networking technology that interconnects them.

I0 uses micro-IP stacks that are compliant enough with conventional Internet protocol stacks, thereby enabling any I0 device to talk to any other device that speaks IP — all in a couple of kilobytes of code. Unlike more ‘conventional’ IP stacks that are run as portions of fully fledged operating systems, occupy 100s of KBs of code, and require processors that draw watts of power, these stacks are small and simple enough to be trivially embedded into a $1 8-pin microcontroller that consumes only milliwatts of power. Moreover, that custom stack can be translated into a couple of square millimetres of silicon that can be added to an ASIC.

The fears of inefficiency in IP is derived from the requirement of the IP header — any transmitted data must be prepended by 20 bytes of the IP header, plus an additional 12 or 20 bytes for a UDP or TCP header respectively. If proportionally, data is not larger than that header, then transmitting IP may be wasteful, detractors argue. However, on the byte level, IP compression is rather standard [3, 4] achieving header sizes as small as 5 bytes in long-lived TCP/IP flows (tighter compression may be possible by compressing the entire packet and not simply the header). As for power considerations, power does not equal transmitted bits, rather, power is related to receiving bits. A sender and receiver can agree on a strategy for the receiver to turn on its amplifier to listen for a low signal at a particular time in order to be more power efficient.

IP on its own, however, is not that useful as most network applications tend to use IP to wrap the user data protocol (UDP) [5] or the transmission control protocol (TCP) [6]. While there is much debate on which protocol is the correct one to use in particular situations (consider a light switch based on UDP versus one based on TCP — the UDP-based switch requires much less bandwidth, however on a congested network where the light switch’s packet is dropped, the person toggling the switch may need to press the button again, unlike a TCP-based switch which would handle the negotiation itself; Bauer and Patrick propose a human-factors extension to the ISO/OSI network model [7] that formalises scenarios just as that), either can be implemented on a small scale. TCP does require more state than UDP; however, implementing TCP does allow for trivial interoperability with standard network applications such as Web browsers.

We assume that all data is transmitted to I0 nodes in a serial fashion — specifically using the serial line protocol (SLIP) [8] (the predecessor, and simpler, sibling to the point-to-point protocol (PPP) [9] which is currently used on most dial-up and some wired connections to ISPs). SLIP specifies a start byte that is used to frame a packet that is sent over the line making it very simple for an I0 processor to pick out the start of the packet. At no point does the I0 processor need to buffer the entire packet or even the packet header in memory (an implementation is free to do so, if it wishes, but it is not necessary), it can simply read and process the packet as it streams in. While it streams in, the software only needs to keep track of a few bytes of state — a 16-bit sum of the packet header (used for checksum verification), the source IP address, and the destination IP address. Once the checksum can be confirmed to be correct, the processor can determine whether the packet is destined for it. If not, it can simply ignore all incoming bytes until the next SLIP start byte. All IP header processing can be performed with only 10 bytes state and through use of accumulated operations.

Following the above, transmission is straightforward. When the device wishes to transmit data, it only needs first to transmit 20 bytes of header containing the length of data it wishes to send, an identification number, a checksum, and the source and destination of the data (the rest of the fields in the IP header are kept constant). The identifier of the data could be chosen at random (but to co-operate with the rest of the Internet, it can instead be monotonically increasing) and checksum computations can be kept to a minimum by precomputing the checksum with known constants. That pre-computed value can then be updated with the length of the packet, its identification number, and the destination IP address.

abstraction layers also fall into redundancy traps

UDP transmission comes next. The UDP header specifies a destination and source port number for the packet, as well as a checksum over the entire UDP header. As with the IP header, the UDP checksum can be pre-computed so the source and destination ports can be transmitted, the checksum can be updated with that information and transmitted, and then the actual data can follow. Performing TCP/IP at this scale is possible, but we do not cover it within the scope of this paper.

3. Compiled standards

The above Internet protocol stack reduction is an example of compiling standards. The IP stack is traditionally described using the 7-layer ISO/OSI network model which dictates network handling through physical, data link, network, transport, session, presentation, and application levels. This model serves well for the design and maintenance of the networking stack where a different entity takes charge of each individual layer. Unfortunately, this model also leads to networking stacks that are more complicated than the application that is using them; layered networking stacks are very appropriate for general-purpose operating systems that may not know what program is running on them a priori, and so they must handle all possible error cases that may occur. Fortunately, a light bulb does not need to run many simultaneous processes.

A light bulb can well define all the capabilities of the IP stack that it will require beforehand, and therefore it is possible to ‘compile out’ of the stack whatever portion is no longer necessary. This is very similar to what a software compiler does, as an optimising compiler is able (within certain bounds) to identify segments of code that will never be executed, and then simply leave them out of the compiled executable. A given light bulb may also know, at compile time, which port it is going to open. Armed with that knowledge, a software engineer can simply remove the portion of the stack that signals an error when two applications attempt to open the same port for listening. There are many more of these types of optimisations that are possible.

Abstraction layers also fall into redundancy traps. Each layer may need to reverify a data buffer, or re-do computation that another layer may have already performed simply because it does not have access to cross-layer information. Crosstalk can be useful in removing redundancy, or performing a joint-optimisation between layers for careful control [10]. The I0 micro-stack is an example of where information at the IP layer (traditionally encapsulated at the network layer of the ISO/OSI stack) is used to optimise packet handling at the TCP/UDP/HTTP layers (the transport and application layers), thereby creating much tighter network code.

Lastly, this type of optimisation co-exists quite nicely with Moore’s law. Even as processors get smaller and smaller, careful software optimisation means that an even smaller, cheaper processor that has a lower power draw, and is simpler to package, can be used to implement the Internet.

4. Peers do not need servers

I0 devices function without the need of servers as any two nodes, via the Internet protocol, have the direct ability to talk to each other without having to go through an intermediary; two nodes can exchange information directly rather that going through a central broker to get the same information. This independence allows each node to have ownership over its state and threads of execution. This model also addresses scalability for data storage and computation through redundancy and locality in ways that a centralised system cannot [11, 12].

Centralised systems, such as the Web server/client relationship, are prone to failure at the one information source. If a Web server fails, then the Web client has no redundancy plan and is required to simply wait until the problem has been rectified. Distributed systems, such as the GNUtella file sharing network, do not have this problem. GNUtella nodes locally cache information and are redundant data sources throughout a network — when a particular piece of information is requested, the network can provide many sources, with the client even being free to choose a source which it believes will be ‘easier’ to access. When a single node on the network is removed, the rest of the network still operates without failure.

The above holds true when discussing an Internet 0 device network. If all devices are required to proxy their information through a centralised node, then the entire network is prone to failure when that single node becomes overwhelmed, fails, or is under attack. Allowing a more open network where devices directly intercommunicate with specific other nodes means that failures are localised to those relationships; if a single node goes down, all that is affected are the other nodes that deal directly with it. No other state nor execution is directly affected.

All this is not to say there is no room for centralisation of hierarchy in this network [13]. Google is a prime example for centralisation — the Google spider walks the entire World Wide Web indexing information and providing it so that a user can access at a single point, the Google Web site. Without Google, the WWW would continue to exist and function normally; however, Google has added a higher level service to the network that makes it more valuable. Hierarchy is too important as it solves problems where certain nodes may have very valuable information that all other nodes wish to obtain, but which, for engineering issues, would be too much of a burden to require a single node to disseminate it to the entire network [14, 15].

5. Physical identity

The crux of any network is the ability for nodes to be able to identify each other. Names, however, mean very different things to different people. Computers on the Internet have Internet addresses as their names, but those names are only used to specify where in the network the computer is located. Network adapters, however, do have hardware addresses that allow for unique identification between machines, but the management of such a scheme can be burdensome.

Internet protocol addresses are not suitable for identification purposes because they are not doled out on the basis of physical location; rather, they are assigned based on where on the Internet hierarchy that machine currently resides. Additionally, many organisations assign internal and unroutable IP addresses to their organisation's computers in such a way that there does exist another machine in the world with the same IP address (most network address translation (NAT) equipment obtain a single IP address on its globally routed interface, and then assign IP address from the 192.168/16 subnet to the internal machinery — therefore there does exist an approximately one in 100 000 chance that a computer inside a NAT has an address that is used by another computer in a different NAT). IP addresses are simply not globally unique, nor do they have any notion of permanence.

Hardware addresses, such as those used as the media access control (MAC) address on Ethernet are, however, globally unique. To maintain such a system requires a centralised serialisation authority — the IEEE. The IEEE makes sure never to assign the same block of hardware addresses to two different parties at the cost of those parties purchasing either an 'organisationally unique identifier' or an 'individual address block' at the rate of US$1650 and US$550 respectively. This, unfortunately, locks out many experimenters and developers from creating their own network interfaces.

I0 devices rely on zero configuration schemes [16] to obtain IP addresses along with a random 128-bit string as its hardware address. The use of a 128-bit string as a MAC-like address comes from the observation that the chance for collision of two IID strings of that length is approximately 1 in 10^{38}, making it 'mostly' unique (MAC addresses need not actually be unique, simply unique enough that two interfaces with the same MAC address do not appear in any network smaller than two subnets bridged together, but also that it may be possible to use that string as an IPv6-like address [17] in future work.

With the ability to physically identify devices, a new programming paradigm can be introduced which involves physically accessing the network nodes. There are certain operations that one may not want to expose over the network — however, forcing an operator to physically access a node to verify that he or she has permissions to perform the network access is very promising.

6. Big bits

Most development in networking technology has been allocated to going as fast as possible as that means saturating all available bandwidth on a channel — for the Internet that means hardware research is devoted to faster-than-terabit Internet 2 links, while software research delves into saturating those links [18]. Unfortunately, two crucial points are easily forgotten in this race — a light bulb does not need to watch video on demand, and there are many hidden costs to pushing bits quickly.

Every bit in a network has a size. Given the speed of light, transmitting one bit a second means that the bit grows to a size of 3.0×10^8 metres long. Likewise, in a gigabit network, each bit has a size of approximately 30 cm. This bit size is effectively the window of opportunity for the two devices to agree on what is being transmitted on the network. When the network is operating a very fast data rate, there are considerations such as the impulse response of the medium and impedance matching between interfaces that must be accounted for (the impulse response dictates what the onset of a bit looks like, while the impedance matching allows for efficient power transfer between media without causing an 'echo' of the transmitted energy to be reflected back to the source), which in turn causes the network technology to become complicated and expensive as agile radios, active splits, and efficient cabling is needed.

every bit in a network has a size

If the network is slowed down such that a bit is larger than the structure of the network, each node is effectively operating in the near field. All the vagaries of the network do settle down on that time-scale, and the entire network reflects the value that the transmitter is sending. Less consideration needs to be made to the nonlinearities that occur at high data rates, and therefore transmitters and receivers can be constructed very simply and cheaply.

7. End-to-end modulation

The end-to-end principle in systems design puts all the interaction intelligence at the edges of the network, and not in the central core. The central core is kept as agnostic to the actual transmission as possible to prevent redundancy between central nodes, and between central nodes and those involved with the communication at the end points. The Internet exhibits end-to-end design in its use of the Internet protocol — no matter what hardware is being used, or what application is being run, all of them use the same network transport allowing for flexibility because neither the hardware nor the application need know the details of the other. A non-end-to-end system would require that intermediary nodes process and interpret all the data that is streaming in, possibly reformat it, and then retransmit the data.

Internet 0 relies on an end-to-end modulation scheme to transmit Internet protocol packets so therefore it is not only agnostic to which network the information it is transmitting is destined for, but also the transmitter need not worry as to the actual media that the data is moving through. This is achieved by using a modulation scheme based upon impulse radio [19] where data is transmitted through time positioning of high-frequency 'clicks' (a 1 µs click yields saturation from DC to 1 MHz). These clicks can be passed through almost all media — through IR via the flashing of a LED, through the air via ultrasonic speakers, through AC power lines by capacitively coupling, etc. Each one of these media has very specific frequency pass-bands and other transmission characteristics; however, they can all pass a portion of the transmitted energy. As long as enough energy is received by the other end in a manner that allows for careful positioning of the onset of that energy, then this encoding scheme is appropriate. These media can also be coupled together without the need for demodulating at the terminal of one, and then remodulating at the terminal of the other — this is very similar to a Morse code operator synchronising their dots and dashes on an electrical telegraph with the flashes seen from a light coming from a ship; there is no need to actually translate those pulses into English and then back into code.

A single bit is divided into two time intervals in a Manchester-like encoding scheme — if an impulse occurs precisely in the centre of the first interval, then a 0 is being encoded. Likewise, an impulse precisely in the centre of the second interval encodes a 1. Any other impulse can be rejected as noise. These bits are then strung together in an 8N1 serial fashion with a start bit, eight data bits, and then one stop bit. Both the start and stop bits are identified as they have transitions precisely in the centre of both the first and second bit time intervals (Fig 1).

This modulation scheme has the property of being able to reject spurious transitions as they will be incommensurate to the rest of the byte; if a click appears in a place where one is not expected, then that bit can be easily thrown out and rejected. Similarly to most UWB systems, a spreading code can be used for additional noise rejection [20, 21] through the careful positioning of the start of each byte click sequence (there do exist simple implementations of the spreading encoders and decoders that can be used in an I0 device [22]). The onset of each byte is dictated by the spreading code, and not the positions of each bit, to allow a transmitter to use a spreading code if desired, but not dictate that the receiver use one; a receiver can receive an I0 click sequence which has had the onset of each byte positioned through spreading, and simply ignore that additional piece of information and decode as before.

Additionally, this scheme has no specification of how quickly or how slowly the transitions can be sent — the only item gating their speed is the impulse response of the system. This self-clocking specification is appropriate to being run at terahertz speed for on-chip communication, or millihertz speed for encoding into the waves of an ocean. Finally, it is also promising in being able to allow multiple transmitters to share the same channel as a receiver with enough computational power can separate out multiple transmitters based solely on the click interval they are each using. A transmitter can simply pick a random click interval, and then blindly transmit on the channel — if the receiver knows exactly what click interval it is looking for then it can reject all other impulses as noise, or it can simultaneously decode all the data, and sort through all the incoming data based on click length.

8. Open standards

The Internet has grown as fast as it has because of open standards. No licensing fees are necessary for anybody to create and then deploy hardware and software on the Internet, as long as one stays compatible with the Internet protocol. This one protocol is open for anybody to implement, and it is in the implementor's best interest to stay compatible so that he or she can intercommunicate with other machines and people.

the Internet has grown as fast as it has because of open standards

A classic example of non-open standards and closed systems is the global cellular phone system. Almost the entire world relies on the GSM system for cellular service — this allows manufacturers to make the telephones they wish to make, it allows service providers to simply provide the end user with service, and it allows third parties to deploy applications on the network that people can then use and pay for. North America, on the other hand, used to operate on closed systems. Carriers used to compete on infrastructure, and therefore they negotiate with manufacturers to create the telephones they wish to make, and applications (if they are ever developed by an outside party) need to be carefully managed and deployed on individual systems. On every link, the quality of the service may have been better than GSM, but the overall quality that the consumer received was less because the system did not provide for rich access.

This same battle is being played out in the building infrastructure space. There are many different competing standards; however, they all have intricate licensing and co-operation fees. Echelon Technologies both sells devices which

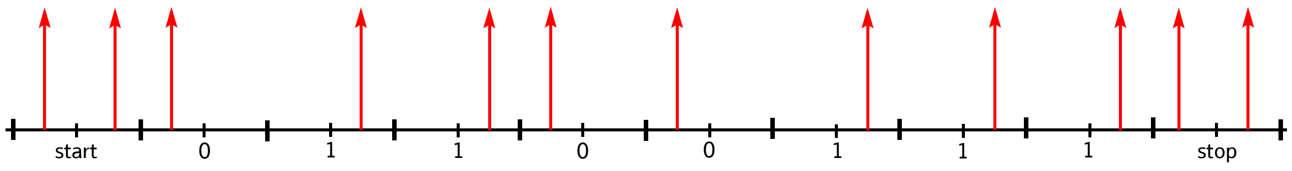

Fig 1 One byte being transmitted as clicks.

operate on the LonWorks network and maintains the Open Systems Alliance which is a group of corporations orbiting around this standard. Development of integration technologies related to LonWorks requires membership in the Open Systems Alliance; however, membership only qualifies those to develop systems that use LonWorks, it does not allow one to create new devices and software to interact with the network. That requires licensing of a patent held by Echelon.

Another example is the UPnP (universal plug-n-play) standard. UPnP is an interoperability protocol aimed to allow devices to automatically configure themselves when in the presence of others. Obtaining the standards for UPnP is simple as they are widely published; however, to deploy a commercial device and advertise it as UPnP compliant requires registration into the UPnP Implementors Corporation. Therefore, while it is possible to create a UPnP compliant device, it is not allowed to advertise itself to the world as compliant until the creator joins the corporation.

It is systems and administrations such as these that leave device networks in the same state as computer networks before the introduction of the Internet. Internet 0 is aimed to be as open as the Internet standards are, therefore allowing anybody to implement it and therefore be interoperable with any other device.

9. Implementations

There are two implementations of Internet 0 currently under development. The first is a very lightweight one based around the 8-pin ATMEL ATTiny15 series of microprocessors (Fig 2). It has a single data port over which it receives both DC power and it can 'click' a transmission on top of that DC offset — it is then possible to use that same data port to transmit over any media simply by AC coupling in (to remove the offset), and then transmitting those impulses. This particular board is aimed towards tagging and other very simple communications applications. The scenario under development is a tag that can be powered without contact (either inductively or capacitively) and then transmit a packet out to a server somewhere on the Internet for data collection.

The second embodiment is a slightly more powerful board that can both transmit and receive I0 packets. It is currently based around the ATMEL ATTiny26 series and has three data ports (using the same DC offset technique as mentioned above) to allow for branching and routing in the network. Also, those three data ports are then amplified and fed through a comparator to pick out the impulses that may be occurring on the line. This board is not capable of high speed communications, but it does provide ample room for experimentation and development.

10. Conclusions

Internet 0 describes how a subnet operates — everybody within a cloud transmits to and receives from everybody else in the cloud. At no point has Internet 0 specified routing or hierarchy, and in order to speak globally it has to rely on the infrastructure already laid in place by the Internet. But after having removed servers from Internet 0, it seems unfortunate that it still needs to rely on servers to work globally. A possible solution to this problem lies in mathematical programs and graphical models [23]. These models specify global optimisation solutions through local problem solving, and this is precisely what you want to do with routing and naming on a global scale — the ideal solution would be to have nodes co-operate on a local level to allow for the network on the global scale to function.

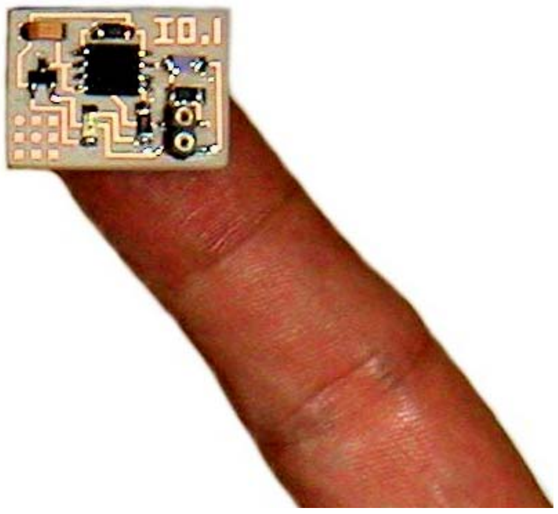

Fig 2 8-pin microprocessor I0 board.

As it stands, Internet 0 is an enabling step to bring networking to the device level. It is not simply getting a few nodes to talk to each other, instead it is transforming these small and embedded devices into first class citizens on a global network. This is a large step towards extending the reach of the Internet.

Acknowledgements

The authors would like to thank the National Science Foundation for their support of this work. They would also like to thank the members of the Things That Think consortium for their generous support. Finally, they would like to thank BT for their support of the Fellows Program.

References

1 Gershenfeld N, Krikorian R and Cohen D: 'The Internet of Things', Scientific American (October 2004).

2 Postel J: 'Internet Protocol', IETF RFC 791 (September 1981).

3 Degermark M, Nordgren B and Pink S: 'IP header compression', IETF RFC 2507(February 1999).

4 Cellatoglu A, Fabri S, Worrall S, Sadka A and Kondoz A: 'Robust Header Compression for Real-Time Services in Cellular Networks', Proceedings of 2nd Int Conference on 3G Mobile Communication Technologies, London, UK (March 2001).

5 Postel J: 'User Datagram Protocol', IETF RFC 768 (August 1980).

6 Postel J: 'Transmission Control Protocol', IETF RFC 793 (September 1981).

7 Bauer B and Patrick A: 'A Human Factors Extension to the Seven-Layer OSI Reference Model', Institute for Information Technology, National Research Council, Canada (2002).

8 Romkey J: ‘A Nonstandard For Transmission of IP Datagams over Serial Lines: SLIP’, IETF RFC 1055 (June 1998).

9 Simpson W: ‘The Point-to-Point Protocol (PPP)’, IETF RFC 1661 (July 1994).

10 Chiang M: ‘To Layer or not to Layer: Balancing Transport and Physical Layers in Wireless Multihop Networks’, Proceedings of IEEE INFOCOM, Hong Kong, China (2004).

11 Stoica I, Morris R, Karger D, Kaashoek M and Balakrishnan H: ‘Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications’, Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, ACM Press (2001).

12 Waldspurger C, Hogg T, Huberman B, Kephart J and Stornetta W: ‘Spawn: A Distributed Computational Economy’, Software Engineering, 18, No 2, pp 103—117 (1992).

13 Yang B and Garcia-Molina H: ‘Comparing Hybrid Peer-to-Peer Systems’,The VLDB Journal (September 2001).

14 Fuller V, Li T, Yu J and Varadhan K: ‘Classless Inter-Domain Routing: an Address Assignment and Aggregation Strategy’, IETF RFC 1519 (September 1993).

15 Mills D: ‘Internet Time Synchronisation: The Network Time Protocol’, Global States and Time in Distributed Systems, IEEE Computer Society Press (1994).

16 Stirling D and Al-Ali F: ‘Zero Configuration Networking’, ACM Crossroads, 9, No 4 (2003).

17 Deering S and Hinden R: ‘Internet Protocol version 6’, IETF RFC 2460 (December 1998).

18 Jin C, Wei D and Low S: ‘FAST TCP: Motivation, Architecture, Algorithms, Performance’, IEEE Infocom (March 2004).

19 Win M and Scholtz R: ‘Impulse Radio: How it Works’, IEEE Communications Letters, 2, No 2 (February 1998).

20 Win M and Scholtz R: ‘Ultra-wide Bandwidth Time-Hopping Spread-Spectrum Impulse Radio for Wireless Multiple-Access Communications’, IEEE Transactions on Communications (2000).

21 Laney D, Maggio G, Lehmann F and Larson L: ‘A Pseudo-Random Time Hopping Scheme for UWB Impulse Radio Exploiting Bit-Interleaved Coded Modulation’, Proceedings of the 2003 International Workshop on Ultra Wideband Systems, Oulu, Finland (2003).

22 Vigoda B: ‘Analog Logic: Continuous-Time Analog Circuits for Statistical Signal Processing’, PhD Thesis, Massachusetts Institute of Technology (2003).

23 Kschischang F R, Frey B J, and Loeliger H-A: ‘Factor Graphs and the Sum-Product Algorithm’, IEEE Transactions on Information Theory (2001).

Raffi Krikorian is interested in distributed, ‘organic’ systems of extreme scales.

He is currently building very large networks of very small Internet protocol-enabled devices.

These devices can self-organise and co-operate for use in distributed sensor networks and other embedded infra-structures without centralised servers.

Previously, he worked on autonomous mobile code systems and distributed computational systems.

A second-year master’s student, he has both bachelor’s and master’s degrees in engineering and computer science from MIT.

Neil Gershenfeld directs the Center for Bits and Atoms, a part of the MIT Media Laboratory, and heads its Physics and Media research group. He investigates the relationship between the content of information and its physical repre-sentations. His projects have led to new paradigms for computation, including the development of tangible interfaces using everyday objects, affective computers that recognise and respond to human emotion, wearable computers sewn with electronic embroidery and powered by human motion, a ground-breaking demonstration of quantum computation, and even interspecies information technology for animals. A prolific author, he received a BA in physics from Swarthmore College, a PhD from Cornell University, and was named a Junior Fellow of the Harvard University Society of Fellows. Prior to coming to MIT, he was a member of the research staff at Bell Labs.