Editor: Siobhán Clarke • siobhan.clarke@cs.tcd.ie

When TCP Breaks

Delay- and Disruption-Tolerant Networking

Stephen Farrell, Vinny Cahill, Dermot Geraghty, Ivor Humphreys, and Paul McDonald • *Trinity College Dublin*

The authors give an overview of current work on delay- and disruption-tolerant networking and review the overall architecture proposed by the Internet Research Task Force's Delay Tolerant Networking Research Group. Their approach to networking makes no assumption that nodes will have end-to-end connectivity, which could be missing with extremely high-latency connections, if the nodes are only in contact with one another infrequently or if contacts are being continually disrupted. They also describe the main protocols the group is developing and give examples of some pilot networks that use these protocols.

Sending data over a computer network can be difficult, especially if that network suffers significant delays or disruption. There are many reasons for this, along with various ways to handle such problems. You might take the view that delays or disruption don't require any new networking technology — thinking, for example, that standard Internet protocols are sufficiently capable. However, in certain networking scenarios, important Internet protocols just aren't usable, and it's for these cases that a number of research groups are developing the delay- and disruption-tolerant networking approach.

The critical fact is that, in such situations, the Transmission Control Protocol (TCP)[1] often doesn't work, although it underlies most of the applications we use every day — email, the Web, and enterprise single-sign-on, to mention a few. So, if we have a set of interesting application scenarios for which TCP simply can't work, then we really do have a compelling need for new protocols. We argue that new delay- and disruption-tolerant protocols are required and outline the ongoing work in researching, developing, and piloting protocols that address these scenarios.

The Problem with TCP

To explain why TCP presents such problems for certain application scenarios, it might help to provide an example. Much of the work described here has its roots in a NASA research project to develop an interplanetary Internet, or *interplanetary network* (IPN). The basic idea is to try to make data communications between Earth and (very) remote spacecraft seem almost as easy as that between two people on different sides of the world. As it happens, before a network node can send any application data using TCP, a three-way handshake is required that consumes 1.5 round-trip times (RTTs). There's also a generic, two-minute timeout implemented in most TCP stacks: if no data is sent or received for two minutes, the connection breaks. Putting these facts together, we can see that once a spacecraft is more than a minute away (in terms of light-trip time), every attempt to establish a TCP connection will fail, and no application data will ever be transmitted. In the case of Mars, for example, at its closest approach to Earth, the RTT is roughly eight minutes, with a worst-case RTT of approximately 40 minutes. Thus, normal TCP can't work at all for Earth–Mars communications.

Of course, communicating with spacecraft will never be as easy as using the terrestrial Internet because many other difficulties must be overcome — for example, your radio antenna is frequently on the wrong side of the planet. At

 Published by the IEEE Computer Society 1089-7801/06/$20.00 © 2006 IEEE

least in terms of networking, however, we can make good progress compared to how spacecraft data communications currently occur, which is to essentially be manually scheduled on a mission-by-mission basis.

Interplanetary becomes Delay-Tolerant

At its core, the "vision" behind this work is to try to extend Internet architecture to cater to application contexts for which delays or disruption are significant factors. This vision was initially fostered and promoted, in large part, by a relatively small group of people, including Vint Cerf and NASA engineers at the Jet Propulsion Laboratory (JPL) and elsewhere. Their work on IPN began in 1998, and, since then, a couple of fairly substantial protocol development groups and many interesting projects have received funding to further develop their work.

The IPN group eventually morphed into an Internet Society (www.isoc.org) IPN special-interest group (IPNSIG; www.ipnsig.org), which had (and still has, to an extent) a public Web site and mailing list for discussions on relevant topics. IPNSIG developed an architecture for a large-scale network and made some progress toward developing protocols for this architecture. In fact, much of that work survives in the delay-tolerant networking (DTN) protocols' latest versions, and work on an IPN is ongoing within NASA. Nevertheless, the IPNSIG had a problem — it's very hard to experiment with an interplanetary network, given that one doesn't exist and would be very, very expensive to create!

At the same time, other researchers were investigating how IPN concepts might apply to terrestrial applications — in particular, sensor networks, which turn out to have a lot in common with a putative IPN. Given that experimenting with a sensor network is a lot easier to do (and to get funding for), it became clear that the IPNSIG was no longer the best venue for doing work on this topic. Consequently, the Internet Research Task Force (IRTF) created a new research group to examine the more general area of DTN — that group is called the DTNRG, and it's currently the main open venue for work on the DTN architecture and protocols.

The DTNRG is developing two main protocols, the Bundle Protocol and the Licklider Transmission Protocol (LTP), which we discuss in more detail later. The DTNRG is documenting these protocols as so-called experimental RFCs — which

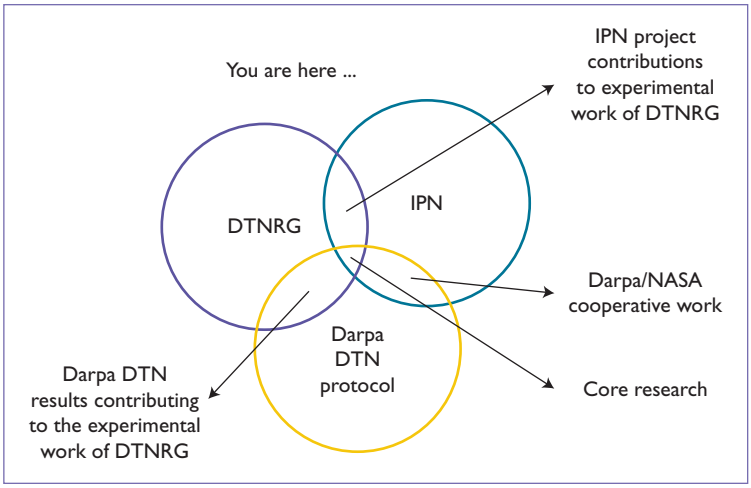


Figure 1. Delay-tolerant networking (DTN) diagram. Several organizations, including the Delay-Tolerant Networking Research Group (DTNRG), the interplanetary networking (IPN) group, and Darpa are trying to solve DTN and disruption-tolerant networking issues. (Figure courtesy of Vint Cerf and DTNRG members.)

are for protocols whose benefits are still uncertain, should they undergo wide Internet deployment. If, over time, the Internet community does believe that the Bundle Protocol or LTP is sufficiently useful, then the IETF would likely form a working group to produce new standards-track protocol specifications and create updated protocol versions.

To confuse matters somewhat, the US Department of Defense, under Darpa, issued a call for proposals in early 2004 for what it called "disruption-tolerant networking" (also called DTN), which is yet another generalization of the same concept. The difference is that until the Darpa call, the main focus of DTN work was on high-delay cases such as with the IPN or sparse sensor networking (in which sensor readings aren't needed in real time). However, other types of disruption can occur — such as radio shadowing or frequent passage in and out of base station range, for example — a fact that the phrase "delay tolerance" doesn't properly reflect. Whether the D in DTN will come to mean "disruption" or continue to mean "delay" isn't yet clear, but, in any case, the same architecture and protocols can hopefully serve in both contexts.

So, several different but overlapping groups are working toward the common goal of developing DTN protocols. Figure 1 shows a graphic that we used at a recent DTNRG meeting to explain this to the audience. The remainder of this article primarily describes the DTNRG position because it's the only fully open group depicted in the diagram.

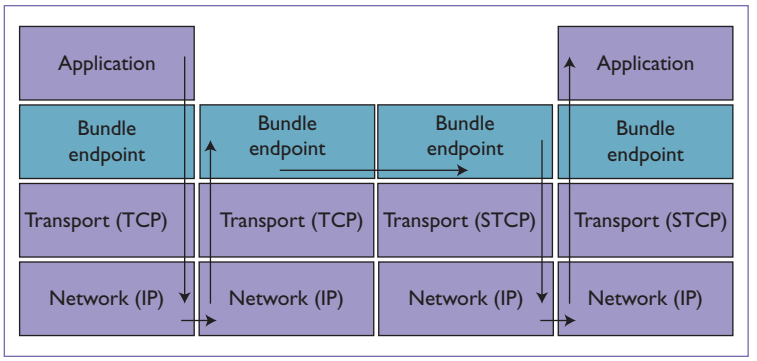


Figure 2. The overlay network approach. The Bundle Protocol, in teal, can run over various transport and lower-layer protocols.

DTN Protocols

Certain terrestrial applications must deal with various forms of disruption, as well as delays, although not on the same scale as light-trip times across the solar system. Delays and disruption in these cases will much more likely be due to the operating system turning off battery-powered devices to conserve scarce power or mobile devices leaving each others' radio ranges. In both cases, protocols that implement a store-and-forward approach reminiscent of how email works can offer significant advantages — and because TCP doesn't work that way, it doesn't work well for these applications.

It turns out that many applications can benefit from DTN. In addition to spacecraft, researchers have proposed or used DTN techniques to support rural schools in developing countries, zebra tracking in Africa, water quality in Irish lakes, social networking in northern Scandinavia, and even monitoring of the cane toad's ongoing invasion of Australia. Details of these use cases and more are available elsewhere (www.dtnrg.org).[2]

Of course DTN doesn't just involve a study of application requirements; we also need to think about some reasonable ways in which we can meet those requirements. So, the DTNRG are developing some protocols that could develop over the next few years into the kind of standard protocol TCP represents.

The Bundle Protocol

The Bundle Protocol is an example of what is generally called an *overlay network* (see Figure 2), and can run on top of the current Internet protocol suite as well as over the more esoteric protocols for spacecraft, complex sensor networks, and other challenging environments. The protocol packages a unit of application data along with any required control information into a "bundle" that's similar to an email message. Nodes then forward this bundle along a route consisting of several intermediate machines that can each store it for significant periods. Thus, the Bundle Protocol is an overlay network store-and-forward protocol.

If the source machine is a lander on Mars, for example, it might create a bundle but be unable to forward it to a Mars orbiter for a few hours until the next time the orbiter is overhead. When the orbiter receives the bundle, it, in turn, might have to store the bundle until its next scheduled contact with an Earth station. When the Earth station receives the bundle, it can quickly forward it to its destination — perhaps the desktop machine of a scientist studying some Martian rock formation. The overall delay could be hours, or even longer if sufficiently intense rain disrupts the orbiter-to-Earth-station contact, in which case it could take days for the data to arrive at the scientist's PC. In this example, the bundle will probably have traversed at least three different lower-layer communication stacks — between lander and orbiter, from orbiter to Earth station, and from there, via the Internet, to the scientist's desktop.

The overlay network approach, as demonstrated by the Bundle Protocol, represents the mainstream DTN approach as regards the number of people working on its specification and development. You can find details about the protocol in two main documents: the DTN architecture document, which introduces the general overlay architecture, puts it in context in terms of applicability, and introduces key architectural terminology;[3] and the Bundle Protocol Specification, which specifies the formatting of bundles and the processing rules associated with sending and receiving them.[4] A few subsidiary documents related to security also exist, including an overview[5] and a document that defines security extensions for the Bundle Protocol.[6]

A DTN node is an entity that runs an instance of the Bundle Protocol and can thus, in principle, send and receive bundles — although some exceptional nodes can only ever transmit (such as a simple sensor), and, more commonly, some nodes might not be able to both transmit and receive simultaneously. We identify nodes via endpoint identifiers (EIDs), which are the bundling equivalent of addresses. EIDs are syntactically uniform resource identifiers (URIs), and each refers to one or more bundle nodes.

Clearly, some component of each DTN must map from EIDs to lower-layer addresses when a node

decides how to forward bundles. The DTN architecture calls for this to follow the late-binding principle so that the URI resolves into a lower-layer address as close to the final destination as possible.

The next concept to consider is the contact, which presents the idea that not all nodes will be in contact at any given moment. This is in contrast to the Internet trend in which we increasingly assume that addressable entities are always online. In DTNs, however, we must explicitly consider that communication is only possible at certain times — perhaps with additional time-varying constraints.

Because bundles must traverse lower-layer networks, they're ultimately subject to whatever restrictions exist on those networks in terms of maximum packet sizes. On most Internet Protocol (IP) networks, for example, it's safest to assume that single packets should be less than 1,500 bytes long. Some DTNs will clearly be subject to extreme constraints in this respect — for instance, if some nodes near the network edge are running over an extremely low bit-rate radio link (as was the case with the Galileo probe to Jupiter, where, due to a malfunction, bit rates in the tens to hundreds of bits per second were common [see www.parkes.atnf.csiro.au/people/jsarkiss/galileo/galileo.html]).

Other DTNs might be able to support forwarding much larger bundles but could be subject to disruption of the lower-layer connections. Of course, in many cases, such as those in which the lower layer runs over TCP, the bundle won't, in fact, be fragmented (at the bundle layer) thanks to the retransmissions of lost IP packets being handled by the TCP layer.

The LTP Protocol

The Bundle Protocol addresses many problems that arise in DTNs. However, sometimes a need exists for delay tolerance at a lower network layer — basically, to handle cases with very high delays between one host and the next. The classic example here is the connection between the orbiter and the Earth station we just mentioned. Some terrestrial applications require similar behavior if, for example, no contact between two machines will ever be sufficiently long to complete an application-layer exchange. In such cases, we need a way to handle forwarding data one part at a time when a delay of hours might exist between each partial transmission. Essentially, we need a delay- and disruption-tolerant point-to-point protocol.

LTP tackles delay tolerance and disconnection in a point-to-point environment with an emphasis on operation over single — but typically very long-delay — links. Such links can suffer from long light-trip times, occultations, and Earth-station scheduling restrictions. If an orbiter is about to be eclipsed behind its planet, for example, it might still send a block of LTP data; knowing that it won't receive an acknowledgment until it's no longer eclipsed, the orbiter can freeze all the timers that drive the protocol for the duration of the eclipse. Once out of the eclipse, the spacecraft can restart these timers. This concept of frozen or "punctuated" timers is a crucial aspect of LTP.

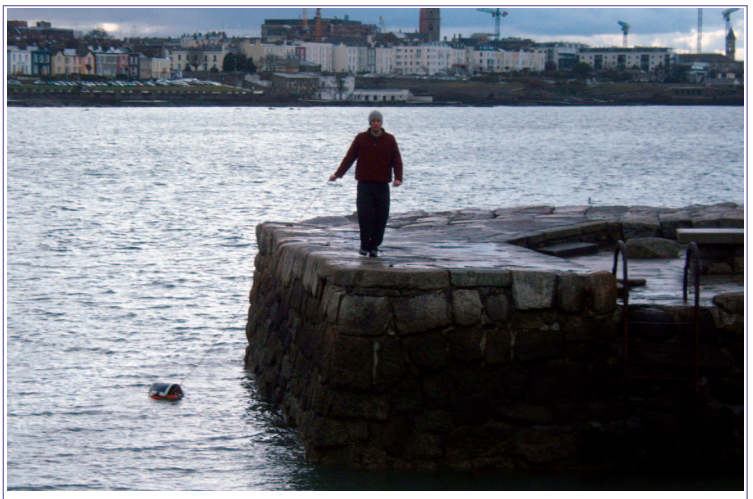
From this description, it might seem like LTP is useful only for space communications, but it can also be useful with terrestrial applications for which disruption is highly likely. Applications dealing with disruptive environments can either be conventionally structured so that the application handles the expected errors, or, using a protocol such as LTP, we can essentially isolate the application from all of this complexity by having a communications daemon that handles all disruptive events, such as retransmissions required after a host shuts itself down. In the case of a sensor network, the resulting sensor code can be very simple, yet have highly reliable transmission over disrupted connections.

It's important to understand that LTP is a point-to-point protocol, so there are no routing or congestion issues to consider — bytes are simply transferred between two peers with no intermediaries. Three documents describe LTP, one covering the motivation for the protocol,[7] one specifying the base protocol itself,[8] and one on protocol extensions.[9]

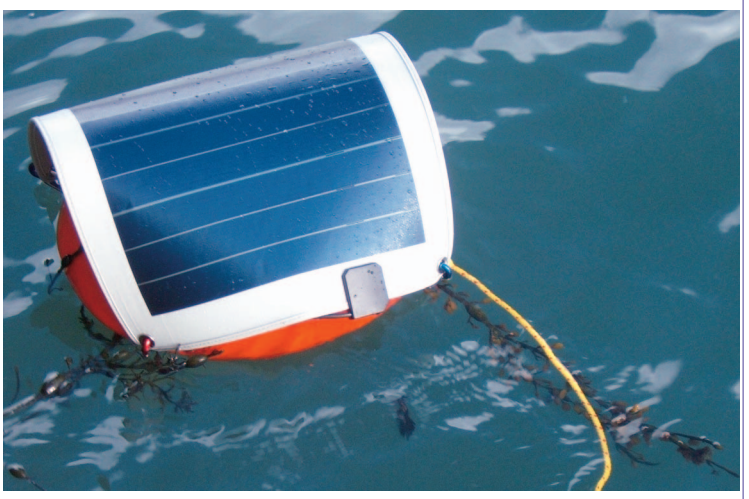
LTP is designed to be a potential convergence layer to support the Bundle Protocol — although we can also use it in other contexts. For instance, we've seen uses for LTP/UDP between nodes in a sparse sensor network.

As we've observed, to operate in such environments, a protocol can't be chatty like TCP is because requiring any round trips before application data flows just isn't an option. Consequently, LTP effectively has no negotiation, and nodes must agree on all parameters required for interoperability before a contact occurs. LTP is thus highly stateful, requiring relatively large amounts of information about previous and upcoming contacts.

LTP separates the handling of protocol exchanges (such as automatic request for retransmission, or ARQ) from issues related to when and how much to transmit or receive. Traditional reli-



(a)



(b)

Figure 3. Delay-tolerant networking (DTN) experimentation. (a) We tested a DTN node by taking it for a walk to check waterproofing and radio range. (b) A close-up of the node in the water.

able protocols (like TCP) handle these issues in an algorithmic manner — in TCP's case, using sliding windows, slow-start, and other mechanisms. Although binding together ARQ and network performance has obviously been enormously successful for Internet transport, several problematic scenarios exist, including deep-space communications and some sensor networks for which TCP and similar protocols are simply unusable.

While LTP uses a fairly standard set of protocol primitives for providing ARQ, data integrity, origin-authentication, reliability, and other on-the-wire issues, it's all done without multiple (or any) round trips prior to sending application data. Where LTP really differs, due to its deep-space heritage, is in its concept of lower-layer cues supporting scheduled communications. We can think of an LTP implementation sitting on top of a separate "layer" that knows the network state sufficiently well to tell each peer when and how much to receive and transmit. As it turns out, this is also a great way to handle a sensor network using data mules.

In addition, if a sender can only communicate with a receiver once each hour for one minute, and the sender expects an acknowledgment message from the receiver within two minutes of sending a message, then the appropriate timer to use is one that will take two hours to expire. The two minutes represent what we might call punctuated time, not elapsed time, where punctuated time is the continually-being-interrupted duration of the scheduled contacts with the peer in question.

Once nodes maintain such punctuated timers independently for each LTP peer and in each direction, then we have already achieved a fairly high degree of delay tolerance. High-latency cases are thus handled by ensuring that the lower-layer cues reflect the communication's in-transit or scheduled latency, which, in the case of deep-space contact, will be dominated by the light-trip times between the peers in question.

Other Protocols

A few other protocols, some historic and some current, are potentially relevant in considering how to meet DTN requirements — perhaps the most frequently quoted of these being the "IP over avian carriers"[10] or carrier-pigeon IP protocol. This was originally a joke, but was eventually implemented and is actually a fairly nice demonstration that IP[11] (although not TCP) can run successfully in many previously unexpected environments.

Regardless of how many times someone says that something good can be done, doing it in the real world is better. This makes attempting DTN deployments quite interesting. At Trinity College Dublin for example, we are building a pilot DTN sensor network (http://down.dsg.cs.tcd.ie/sendt/) for lake-water-quality monitoring and have learned, and continue to learn, many lessons (some obvious, some less so) from this work. Figure 3 shows some of the testing we performed on these sensor nodes prior to deployment, which involved ensuring that the enclosure was waterproof and that radio range wasn't affected so close to the water's surface.

Much as we'd like to, we've yet to attract the level of funding required to attempt a space mis-

Implementations

With something as complex as the Bundle Protocol, the potential clearly exists for us to specify many seemingly sensible protocol features that turn out to be overly complex, useless, or simply broken. In the absence of running code, protocol designers will, in fact, include all three types of error in any reasonably complex protocol. Due to a recognition of this as fact, the Bundle Protocol has a reference implementation that has several useful consequences. First, as we just pointed out, it keeps the protocol designers honest by acting as a reality check whenever paper-only plans get out of control. Second, the reference implementation makes it easier for people to experiment with the Bundle Protocol. An excellent example occurred during the IETF meeting in Dallas in 2006, where a vendor had a special offer for attendees on a new handheld wireless device. During the meeting, an attendee (Well done Jörg Ott!) successfully ported the delay-tolerant networking (DTN) reference implementation to this (relatively) new platform, while still, apparently, paying attention to the proceedings!

The Bundle Protocol reference implementation is freely available for download (look for "code" below www.dtnrg.org) and is currently maintained by a small team (basically, Michael Demmer), but with contributions from other DTN coders. To find out more, download the code and start working with it. There is also a separate NASA Jet Propulsion Laboratory implementation that is tailored for use in spacecraft and which was described at a recent Delay-Tolerant Networking Research Group meeting (http://www3.ietf.org/proceedings/06mar/DTNRG.html).

There are also currently three Licklider Transmission Protocol (LTP) implementations, although only one has been released to date. The Ohio University LTP implementation is available for download at http://masaka.cs.ohiou.edu/ocp/ and is a Java implementation of LTP. We will also release the Trinity College Dublin LTP implementation (in C) in the near future at http://down.dsg.cs.tcd.ie/ltplib/.

sion, so we're limited to modeling how a DTN might work in that context. Of course, others are luckier when it comes to funding space missions, and in fact, the NASA JPL has now implemented a version of the Bundle Protocol intended for use in future space missions. (See the "Implementations" sidebar for information on other implementations of these protocols.)

In the end, DTN is a new technology, and much work remains in a couple of fairly large areas before we can deploy it on anywhere near the scale of even part of the current Internet. By far, the biggest issue is routing, for which we're only now starting to see proposals that we can evaluate. Luckily, a good review of DTN routing has recently been published.[12] The various routing proposals described there could in fact form a sort of continuum, so that future DTN routers (meaning almost all DTN nodes) should perhaps be able to route bundles using one of several schemes, depending on current circumstances. Although other concerns exist, such as the unmet need for some basic delay-tolerant cryptographic key management schemes, routing is by far the biggest hole in today's DTN story.

Fortunately, DTN is a technology that is gaining momentum, due partly perhaps to the attractive nature of the applications it addresses and partly to the amount of work done and remaining to be done, which attracts network researchers (and project funders). Our expectation is that the number of DTN pilots will continue to grow over the coming years, and that, in the not-too-distant future, some commercial DTN applications will appear. Whether DTN as a technology joins the mainstream remains to be seen, but hopefully this article will help you decide whether you think that should happen.

References

1. J. Postel, *Transmission Control Protocol*, IETF RFC 793, Sept. 1981; www.ietf.org/rfc/rfc793.txt
2. S. Farrell and V. Cahill, *Delay and Disruption Tolerant Networking*, Artech House, to be published in Oct. 2006.
3. V. Cerf et al., "Delay-Tolerant Network Architecture," IETF Internet draft, draft-irtf-dtnrg-arch, work in progress, Mar. 2006.
4. K. Scott and S. Burleigh, "Bundle Protocol Specification," IETF Internet draft, draft-irtf-dtnrg-bundle-spec, work in progress, May 2006.
5. S. Farrell, S. Symington, and H. Weiss, "Delay-Tolerant Networking Security Overview," IETF Internet draft, draft-irtf-dtnrg-sec-overview, work in progress, Mar. 2006.
6. S. Symington, S. Farrell, and H. Weiss, "Bundle Security Protocol Specification," IETF Internet draft, draft-irtf-dtnrg-bundle-security, work in progress, Mar. 2006.
7. S. Burleigh et al., "Licklider Transmission Protocol – Motivation," Internet draft, draft-irtf-dtnrg-ltp-motivation, work in progress, Mar. 2006.
8. M. Ramadas et al., "Licklider Transmission Protocol – Specification," Internet draft, draft-irtf-dtnrg-ltp, work in progress, Mar. 2006.

9. S. Farrell et al., "Licklider Transmission Protocol – Extensions," Internet draft, draft-irtf-dtnrg-ltp-extensions, work in progress, Mar. 2006.
10. D. Waitzman, *A Standard for the Transmission of IP Datagrams over Avian Carriers*, IETF RFC 1149, 1 Apr. 1990; www.ietf.org/rfc/rfc1149.txt.
11. J. Postel, ed., *Internet Protocol — DARPA Internet Program Protocol Specification*, IETF RFC 791, Sept. 1981; www.ietf.org/rfc/rfc791.txt.
12. Z. Zhang, "Routing in Intermittently Connected Mobile Ad Hoc Networks and Delay Tolerant Networks: Overview and Challenges," *IEEE Comm. Surveys and Tutorials*, vol. 8, no. 1, 2006, pp. 24–37.

Write for Spotlight

Spotlight focuses on emerging technologies, or new aspects of existing technologies, that will provide the software platforms for Internet applications.

Spotlight articles describe technologies from the perspective of a developer of advanced Web-based applications. Articles should be 2,000 to 3,000 words. Guidelines are at www.computer.org/internet/dept.htm.

To check on a submission's relevance, please contact department editor Siobhán Clarke at siobhan.clarke@cs.tcd.ie.

Stephen Farrell is a research fellow in the departments of mechanical engineering and computer science at Trinity College Dublin. His research addresses delay- and disruption-tolerant networking and security. Farrell has a joint hons. BSc in mathematics and computer science from University College Dublin. He is cochair of the Internet Research Task Force's Delay Tolerant Networking Research Group and of the IETF's Domain Keys Identified Mail (DKIM) working group and has been involved in Internet standards and security for far too many years. Contact him at stephen.farrell@cs.tcd.ie.

Vinny Cahill is a professor in the department of computer science at Trinity College Dublin. His research addresses middleware and programming language support for distributed computing. He is currently investigating dependable sentient computing for applications ranging from intelligent vehicles to independent living. Cahill has a BA, MSc, and PhD in computer science from Trinity College Dublin. He is a member of the ACM and the IEEE Computer Society. Contact him at vinny.cahill@cs.tcd.ie.

Dermot Geraghty is a lecturer in the department of mechanical engineering at Trinity College Dublin. His research interests include implementation of finite element solvers using field programmable gate array (FPGA) modeling and the design of surface acoustic wave strain sensors, signal processing and instrumentation for weight in motion systems, instrumentation for traffic studies, and sensor network hardware. Geraghty has a BA BAI in mechanical engineering and an MSc in mechanical engineering from Trinity College Dublin. Contact him at tgerghty@tcd.ie.

Ivor Humphreys is a research fellow with the department of mechanical and manufacturing engineering at Trinity College Dublin and is involved in the design and development of electronic hardware/software for data acquisition and control within a variety of research projects. His interests include embedded software, PCB design, data acquisition, and wireless networking. Humphreys has a BA BAI and an MSc in mechanical engineering from Trinity College Dublin. Contact him at humphri@tcd.ie

Paul McDonald is a postgraduate research student focusing on wireless sensor networks at Trinity College Dublin. His interests include RF engineering in wireless networks, sensors and transducers, and CAD. McDonald has a BA BAI in mechanical and manufacturing engineering from Trinity College Dublin. He is a member of the Institute of Engineers of Ireland. Contact him at pmcdonal@tcd.ie.