The Case for a Flexible-Header Protocol in Power Constrained Networks

Ignacio Solis and Katia Obraczka
{isolis,katia}@cse.ucsc.edu
Computer Engineering Department
University of California, Santa Cruz

Abstract— **This paper evaluates FLIP, a flexible header protocol for power-constrained, heterogeneous networks. We show that FLIP can improve the energy efficiency of a system considerably, prolonging the system's lifetime. For example, when employing FLIP to implement an existing sensor network communication paradigm, we obtain 50% energy savings when compared to the paradigm's original implementation. Further, we use a sample data gathering application which calculates the average of a sensor network's attribute (e.g., average temperature) and show FLIP's energy efficiency when compared to static header approaches.**

I. Introduction

Networks have expanded and now cover a wide range of devices. To address the needs of heterogeneous networks, we have proposed *FLIP*, a flexible network-layer protocol [1], which incurs minimal overhead when run by simple devices, while still providing a range of functions that can be performed by more sophisticated devices.

In this paper, we focus on how FLIP addresses the challenges posed by networks where most devices are power-anemic. Sensor networks are typical examples: they consist of an arbitrarily large number of sensing devices which rely on relatively short lifetime batteries. Sensor network applications usually imply that sensors will be left on the field unattended for extended periods of time and must conserve energy in order to maximize the whole network's operational time.

A sensor network will rarely need information such as fragmentation, hence these fields are wasteful. On the other hand, designing and optimizing a protocol for a single application/network scenario is prone to many problems. Several protocols will likely have to coexist in the same network and interoperability will be challenging. Besides, in a production network, the cost of redeployment to enable new features might be prohibitive (e.g., unmanned space mission or a sea-bottom monitoring sensor network).

FLIP tries to balance between providing complete functionality and optimizing its overhead by allowing application developers to customize the protocol header. As a result, FLIP's overhead (both in terms of per-packet overhead and protocol complexity) is dependent on the functionality needed by the application.

We demonstrate FLIP's power conservation capability in a number of scenarios. We start by evaluating how well FLIP matches the needs of *directed diffusion* [2], while still being power-efficient. *Directed diffusion* (described in more detail in Section III) is a communication paradigm designed for data gathering applications in sensor networks. Using an optimized FLIP architecture we were able to save more than half the energy consumed by the unoptimized use of *diffusion*.

We then consider a sample network application, namely running average calculation of sensed data. We implement this data gathering application using different protocol header models: FLIP's adaptive header and static headers represented by two models, complete and minimal headers. Our simulation results show that FLIP outperforms static headers by as much as 12% while providing full functionality.

Finally, we add data aggregation to the data gathering protocol and show its energy-saving effects. Adding such new features is part of protocol evolution and can be easily accomplished in the case of a flexible-header protocol like FLIP. Our results show a reduction of 30% in energy consumption when data aggregation is employed.

The remainder of this paper is organized as follows. We describe the FLIP architecture in Section II. Section III evaluates FLIP in the context of directed diffusion. Section IV describes our data gathering protocol for the sensor network running average calculation application. Data aggregation in the context of the running average calculation application is considered in Section V. Sections VI and VII present related work and concluding remarks, respectively.

II. The FLIP Architecture

FLIP is a network protocol designed to interconnect devices with varying power, communication, and processing capabilities. FLIP's flexible headers can provide full functionality in one extreme, or a barebone protocol well-suited for limited capability devices. Because of its customizable headers, FLIP offers close to optimal overhead.

The FLIP header is composed of two parts, the *meta header* and the header fields. The meta header (Figure 1) consists of a bitmap containing an entry for each possible header field. If a field's meta header entry is set to one, then the corresponding field is present in the header; otherwise, the header does not contain that field.

Fields come in the same ordering as the meta header bitmap. The ordering of the meta header bitmap is not arbitrary. Commonly used fields appear near the beggining and infrequently used fields appear last. The current header field ordering was determined based on a somewhat limited set of applications. We plan to further investigate field ordering.

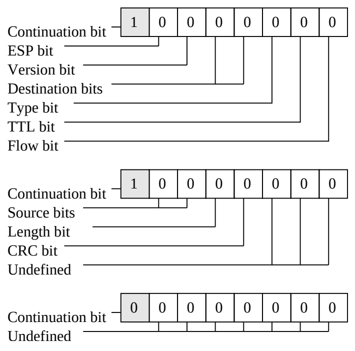

Fig. 1. The FLIP meta header bitmap

The *continuation bit*, the first bit of every meta header byte, defines whether the meta header continues onto the next byte. Currently the meta header is composed of 3 bytes, however not all meta header bits have been defined. Undefined fields are for future use and allow the protocol to evolve. After the third meta header byte, if the continuation bit is set, *user-specified* fields follow. This feature allows FLIP to carry application-specific fields. These fields can be useful for some applications but are not general enough to be included in the predefined meta header.

Usually, FLIP header fields are fixed size. For example, *TTL* is always 1 byte and *length* is always 2 bytes long. Addresses are a special case. FLIP addresses are 2 bytes long since we expect that we will not need globally unique addresses (i.e., system-unique addresses will suffice). However, we also want to allow IP address compatibility. Thus, FLIP addresses can be either 2, 4 or 16 bytes long. That is why the meta header address field is 2 bits. If a packet does not include destination address, it is assumed to be a broadcast packet. Figure 2 shows a sample FLIP packet.

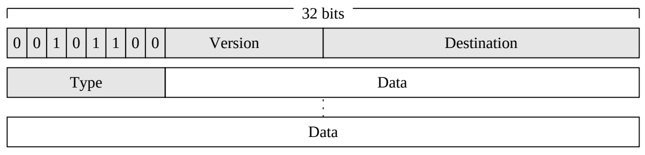

Fig. 2. Sample FLIP packet

FLIP's *Extra Simple Packet*, or *ESP* mode targets simple data exchange. If the second bit of the meta header is set, the packet is considered an ESP. ESPs consist of 1 or 2 bytes, depending on the first continuation bit, and carry 6 or 14 bits of data respectively. ESPs address the requirements of applications that need only exchange small amounts of data with almost no header information. Sensor network scenarios are a good example: sensors can broadcast small data related to what they are sensing.

FLIP allows developers to customize its header by selecting fields required by the target application. Allowing direct manipulation of header fields by applications can be considered a violation of layered system design. However, exposing network-layer features to higher layers allows for protocol optimization, which is especially critical in power-constrained environments.

Rather than addressing protocol design issues, this paper focuses on demonstrating FLIP's energy efficiency in the context of different power-constrained network scenarios.

III. Directed Diffusion and FLIP

Directed diffusion [2] is a communication paradigm especially targeted at data-centric sensor networks. A sink node sends an *interest* for a certain data. This interest will be broadcast to the whole network which will set up a *gradient* along the path. If a node has relevant information to that interest, it will send the data along the gradient back to the sink.

Originally, directed diffusion was implemented as a very specialized application-level protocol. Diffusion implementors' main goal was to develop a working architecture for data-driven sensor networks, rather than build a generic network-layer protocol. In this section, we evaluate the tradeoff between FLIP's flexibility and efficiency as the network-layer protocol underlying diffusion.

We implemented "diffusion-over-FLIP" in two ways. In the first approach, we constructed diffusion's complete header using FLIP and evaluated the resulting protocol's overhead when compared to "pure" diffusion. In other words, we left every diffusion field intact and did not perform any sort of optimization. Secondly, we implemented diffusion from scratch assuming FLIP as the underlying protocol. In this case, we optimized where possible.

A. Diffusion over FLIP

We use the diffusion packet definition from the 3.0 beta release [3]. Figure 3 shows a normal diffusion packet. Most diffusion fields can be directly translated to FLIP header fields: `last_hop` is mapped to `source`, `next_hop` to `destination`, etc. More specialized diffusion fields are carried in the payload. The resulting FLIP packet is 2 bytes longer than the original diffusion packet since we have the overhead of the meta-header.

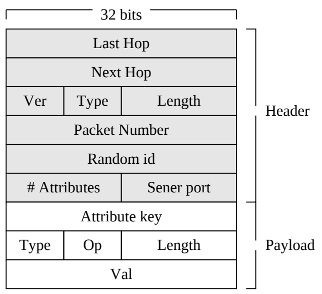

Fig. 3. Diffusion packet for an `int` attribute

B. *Optimizing Diffusion with FLIP*

The second approach to evaluating FLIP in the context of diffusion is to address the following question: how would one re-design diffusion assuming FLIP as the underlying protocol? We consider diffusion's different packet types: `interest`, `reinforcement`, and `data`. As expected from a fixed-header protocol, all packet types use the same packet header. The question then becomes: can one take advantage of FLIP's flexible headers to optimize diffusion's exchanges?

In the case of `interest` packets, the header only needs to carry `source`, `flow (packet id)`, and `type` fields. This customization reduces `interest` headers to 11 bytes, including the meta header overhead. The payload part of this type of packet can be reduced to 10 bytes for simple interests. That is, interests that have only one attribute and that can deduce the type of data from the data key. The total packet size for interests will be 21 bytes, in contrast to 36.

In addition to `interest` header fields, `reinforcements` also require a destination field because they reinforce a specific path. This results in 25-byte packets as we are using 4 byte addresses.

Data packets flow in the opposite direction to interests. Similarly to `reinforcements`, they carry both `source` and `destination` fields because they need to leave a trail for reinforcements. In their payload, we are able to save one byte used for comparing attributes, making it 9 bytes long for an `int` attribute type interest. The total packet length will be 24 bytes. Figure 4 shows the resulting optimized diffusion packets. Shaded and unshaded areas denote header and payload, respectively.

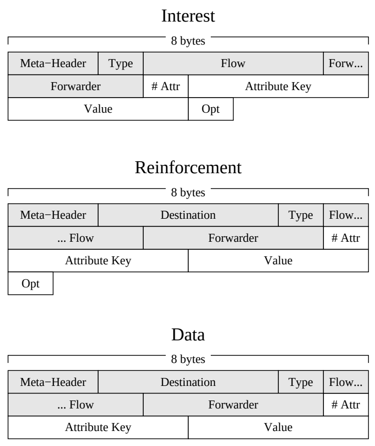

Fig. 4. FLIP-optimized diffusion headers

C. *Simulation Results*

To evaluate these FLIP-based diffusion variants, we modified the original diffusion code in the `ns-2` network simulator from the VINT project [4]. In our experiments, we use sensor networks consisting of 300 nodes scattered across a 2000 x 2000 meter area. 802.11 is the underlying MAC protocol. The energy values were based on the original diffusion evaluation values, i.e., 395mW in reception and 660mW when

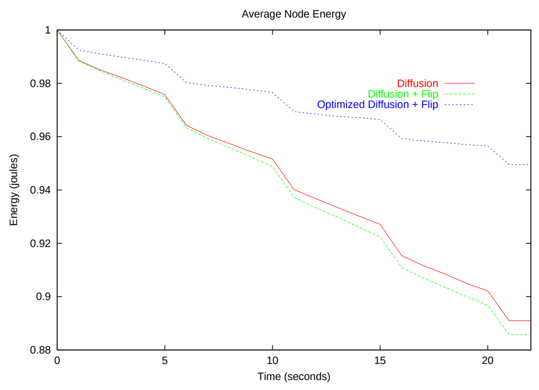

Fig. 5. Energy levels over time for different diffusion variants.

transmitting. Nodes don't move in the sensor network and have a 250m transmission range. To accentuate the difference between diffusion variants we reduce idle energy dissipation to 0, which suppresses the effects of lower layer (data link and physical) overhead. Node failures were not considered.

Figure 5 shows average node energy over time simulation. Node starting energy level is 1.0 Joules. Data points represent averages over 10 runs with different random topologies. We use one sink, one source and a data rate of 10 packets/second. The graph's "step" shape is due to how the diffusion algorithm operates: it resends interests every 5 seconds. Simulations were run for 21 seconds.

As expected, *diffusion over FLIP* consumes slightly more energy than diffusion since the packets are 2 bytes longer. i *Optimized diffusion* on the other hand does save a lot of energy when compared to the other diffusion variants. It uses less than half the energy for the same period of time. This means that FLIP-optimized diffusion could double the lifetime of a sensor network when compared to "pure" diffusion. Table I summarizes energy consumption results for the different diffusion variants.

TABLE I
DIFFUSION VARIANT ENERGY CONSUMPTION

	Energy consumed	packet size
Diffusion	0.109	36
Diffusion + flip	0.114	38
Optimized Diffusion	0.050	varies (21 - 25)

IV. FLEXIBLE HEADERS EFFECTS

In this section, we demonstrate FLIP's energy efficiency in the context of another data gathering application for sensor networks. The specific scenario we consider is temperature running average calculation.

We designed a simple data gathering protocol which works as follows. A requesting node sends a query for a certain variable, for example temperature. Each node then sends back an answer reporting their current temperature measurement. The requesting node calculates the average over each round of reported temperature values. Nodes perform two basic

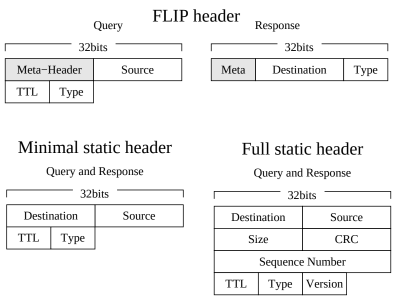

Fig. 6. Header Models

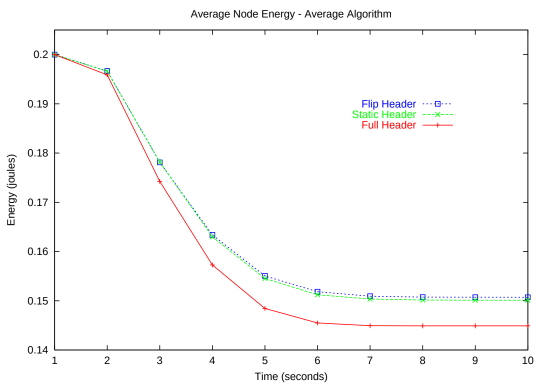

Fig. 7. Energy level in data gathering (temperature averaging) application

exchanges: the query that originates at requesting nodes, and the replies that come back from the network. The `TTL` is decremented at each hop. We will explain the use of the `TTL` in Section V. We describe the different packet formats below.

A. Header Models

We compare FLIP's flexible headers with two static header models: *minimal* and *full* headers. Figure 6 show the three header models considered.

In FLIP, the query packet header consists of `source` (2 bytes), `TTL` (1 byte), and `type` (1 byte). The response header includes `destination` (2 bytes) and `type` only. Query and response header sizes (including meta-headers) are 6 and 4 bytes, respectively. The payload in the two cases are 2 and 4 bytes long. For query packets, it carries the query id; in the case of response packets, the data being reported is also included. This makes the packets 8 bytes long.

Minimal static headers are 6 bytes long. They consist of the union of all FLIP header fields, i.e., `source`, `destination`, `type`, and `ttl`. The corresponding query packet is 8 bytes, like in FLIP. But responses are 10 bytes long.

Full header mode includes all fields typically present in "traditional" network-level protocols: `version`, `source`, `destination`, `type`, `TTL`, `size`, `CRC`, and `sequence number`. The total header size is 15 bytes which makes queries 17- and responses 19 bytes long.

In this particular application scenario, since flows in different directions need to carry different information, FLIP's flexible headers are able to minimize protocol overhead, while not limiting protocol functionality. As our simulation results show, FLIP yields highest energy efficiency even when compared to the minimal header model case.

B. Simulation results

The graph in Figure 7 shows how average node energy varies over time for the temperature average calculation application. Similarly to the directed diffusion experiments, we use a 300-node sensor network spanning a 2000 x 2000 meter area. We also use `ns-2`'s 802.11 MAC protocol and the same energy consumption parameters, i.e., 395mW, 660mW, and 0 for receive, send and idle, respectively. As before, these parameters are based on the empirical values used in the original evaluation of diffusion [2].

Like our previous experiments, we set energy dissipation during idle periods to 0. Reported data points are averages over 10 runs. Initial node energy is 0.2 Joules. The experiment consists of running the average calculation algorithm once, where a node sends a query and waits for the responses. There are no delivery guarantees of any kind. The simulation did not consider node failures.

At the initial part of the graph average energy consumption is low since nodes are only sending the query packet away from the requester node. As more and more nodes reply to this packet and forward the responses, energy consumption increases. After a few seconds the energy levels off as packets arrive at the requester or are lost due to collisions. The number of readings collected by the requester were similar for all three models with an average of 266 readings out of a possible 300.

TABLE II
TOTAL ENERGY CONSUMPTION FOR DATA GATHERING

	Energy consumed	Query size	Response size
FLIP header	0.0493	8 (7)	8
Minimal static header	0.0499	8	10
Full static header	0.0551	17	19

Table II summarizes the energy consumption results for the different header models. The minimal static header model consumed 1.2% more energy than FLIP, while full headers consumed 11.8% more. Both the minimal header protocol and FLIP provide exactly the same functionality, which is optimized for the data gathering application. The full static header model on the other hand includes the usual fields present in "traditional" network-layer protocols. This added but unnecessary functionality results in almost 12% additional energy consumption when compared to FLIP. For applications that need some or all the functionality provided by a full header model, FLIP could easily add the required fields.

One can argue that it is possible to use a different static header for each protocol exchange. To this end, the protocol still needs a way to differentiate between the different packet types. For example, nodes can use the packet type type field to figure out how to process a packet. However, we claim that if protocol designers are willing to make packet processing more complex, they will be better off using FLIP, which is fully customizable. As demonstrated by our results, FLIP's meta header provides an efficient way to define which fields are included in the header.

V. Effects of Data Aggregation

In this section we demonstrate that FLIP's ability to incorporate new functionality may lead to a more (power-)efficient protocol. We modify the data gathering protocol described in Section IV to include data aggregation.

The main idea behind our data aggregation technique is that information from nodes closer to the requester is considered more important than information from far away nodes. The objective then becomes to optimize energy efficiency while still delivering important data in a timely fashion.

Our data aggregation mechanism works as follows. The requester node defines an area it considers important. It does so by setting the TTL of the query packet, which defines the hop count of the *importance area*. At every hop, the TTL is decremented by one. Once it reaches zero, it means the packet left the importance area. After this point the packet no longer needs the TTL field since it already knows it is far away from the requester. Figure 8 shows a sample scenario. The central grey node is the requester and the dashed circle defines its transmission range. The shaded area is an approximation of a 2-hop importance area. The black, or *ring* nodes delimit the importance area.

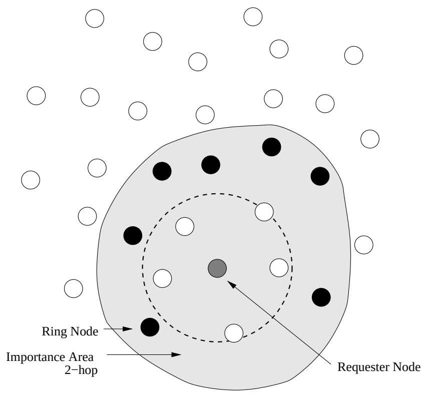

Fig. 8. Sample ring aggregation scenario

Nodes reply as soon as they get a request. Nodes inside the importance area will forward these replies immediately so they reach the requester as soon as possible. Nodes outside the importance area will reply and forward other nodes' replies at their leisure. In our experiments, outside nodes also reply immediately. However, instead of forwarding immediately, ring nodes aggregate replies into a single packet which they forward to the requester when new data is received. Of course the tradeoff is that information from farther away nodes is delayed. However, since this information is not considered critical, the added delay is tolerated.

In these experiments, we used the same simulation parameter values as described in Sections III and IV. The radius of the importance area (number of hops between requester and ring nodes) was set to 4. Periodic messages from ring nodes to requester are sent every 0.5 seconds.

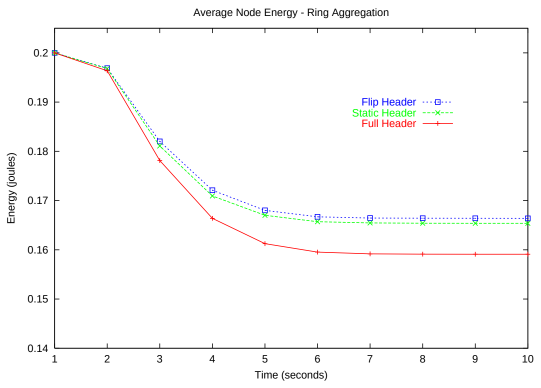

Fig. 9. Energy level in data gathering (temperature averaging) application with data aggregation.

Figure 9 shows the effects of data aggregation when applied to the average calculation protocol. Note that the resulting energy level graphs for data gathering with and without aggregation exhibit similar shape. However, data aggregation results in lower power consumption as nodes inside the importance area end up sending fewer packets when aggregation is used. Table III shows energy consumption with aggregation for the different header models. The requester in all models collected a similar number of readings with an average of 284 out of 300. To show the effect aggregation has on energy consumption we have plotted the results of the experiments that used FLIP in Figure 10.

TABLE III
TOTAL ENERGY CONSUMPTION FOR DATA GATHERING WITH AGGREGATION

	Energy consumption		Energy
	no aggregation	aggregation	savings
FLIP header	0.0493	0.0336	31.8%
Small Static header	0.0499	0.0346	30.7%
Full Static header	0.0551	0.0409	25.8%

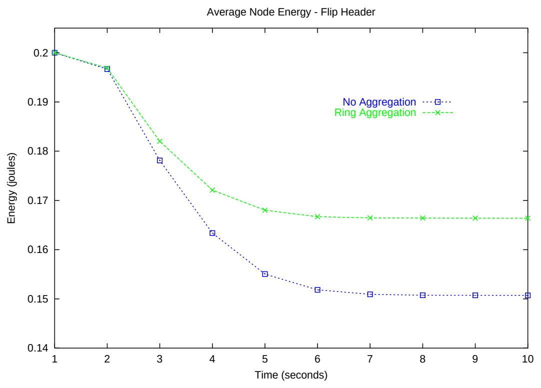

Fig. 10. Data Aggregation effect

We used aggregation as an example functionality that can be incorporated into an existing protocol. FLIP allows the higher protocols to add or remove functionality as needed. In the case of ring aggregation the TTL field was used. When it was no longer needed it was removed, decreasing the header overhead.

With the results from aggregation we have seen that if we oversimplified the header, we could not have provided a feature like aggregation, and would not be able to provide these energy savings, as much as 30.7% with a static header (31.8% with the FLIP header, but that would have provided the ability to add the aggregation functionality). On the other hand, aggregation might not be done all of the time and we might not want to incur in the overhead it requires.

VI. Related Work

Research in communication protocols has been an active area of research. However, to our knowledge, the way we address hererogeneity and flexibility with FLIP is a new approach.

In the Scalable Coordination Architectures for Deeply Distributed Systems (SCADDS) project [3], nodes loose their individuality and the focus lies in the data generated by the whole system. In this context, the *directed diffusion* [2] architecture was developed to convey data from information sources (e.g., sensors) to information sinks. Other research has recently focused on aggregation protocols [9], [10].

Another initiative in the sensor networks area is WINS, Wireless Integrated Network Sensors [11]. They describe their basic sensor network environment and present a sensing device architecture based on layered processing.

There has also been work on header compression. The recently proposed Unified Header Compression Framework [12] aims at creating a standard way in which protocols in general can define header compression. Previous work [13], [14] targets specific protocols such as TCP/IP. Unlike FLIP, these schemes assume a standard point-to-point communication scheme. They are also vulnerable to out of order delivery and drops since the state kept by nodes needs to be updated by every packet to reconstruct the full headers.

VII. Conclusions and Future Work

In this paper we showed the energy efficiency properties of FLIP, a flexible header protocol in a number of power-constrained network scenarios. Flexible headers allow FLIP to customize its functionality based on the target application and the devices running it. We evaluated FLIP in the context of the directed diffusion communication paradigm for sensor networks. In the first set of experiments, we performed direct translation between diffusion and FLIP header fields. We observe a slight increase in the resulting protocol's overhead due to FLIP's meta headers.

We then re-designed diffusion assuming FLIP as the underlying network protocol. Using FLIP's flexible headers, we were able to provide just the required functionality incurring minimal protocol overhead. Simulation results show that *optimized diffusion* can be 50% more energy efficient than original diffusion.

Data gathering applications in sensor networks were the other scenario we used to evaluate FLIP. We used two static header models: *full* headers include most fields present in "traditional" network-layer protocols, while minimal headers, which are optimized for the target application, only carry required fields. FLIP outperforms optimized static headers by a small margin and still has the additional advantage of being able to accommodate other devices if needed. FLIP is able to match the functionality of the full header model and yet yields 12% higher energy efficiency.

We also show FLIP's ability to evolve seamlessly and include new protocol functionality as needed. To this end, we enhanced the simple average calculation protocol with aggregation. When compared to the original version of the protocol, data aggregation reduced the system's overall energy consumption by as much as 30%. The addition of this feature required the use of the TTL field. TTL (or any other fields) could be easily incorporated into the FLIP header. If the optimized static header had not anticipated the need for this field, the feature could not have been added.

No matter how much protocol designers plan, they are not able to predict all possible features a protocol should have. A clear example is IP addresses. Designers predicted that 32bit addresses would last a long time. Flexible protocols like FLIP permit a more seamless protocol evolution.

References

[1] I. Solis and K. Obraczka, "FLIP: a flexible protocol for efficient communication between heterogeneous devices," in *IEEE Symposium on Computers and Communications*, July 2001.

[2] C. Intanagonwiwat, R. Govindan and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in *6th International Conference on Mobile Computing and Networking (MobiCom)*. ACM, August 2000.

[3] USC/ISI, "http://www.isi.edu/scadds/," May 2000.

[4] The VINT Group, "VINT:virtual internet testbed," http://netweb.usc.edu/vint, 1996.

[5] "Packet radio topics in professional journals, http://www.tapr.org/tapr/html/biblio.html," August 2000.

[6] DARPA, "http://www.darpa.mil/ito/solicitations/glomo/glomobrief.html," February 1995.

[7] IETF, "http://www.ietf.org/html.charters/manet-charter.html," July 2000.

[8] "The bluetooth project," Available from http://www.bluetooth.com/.

[9] C. Intanagonwiwat, D. Estrin, R. Govindan and J. Heidemann, "Impact of network density on data aggregation in wireless sensor networks," Under submission, November 2001.

[10] W. Heinzelman, J. Kulik and H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," in *Proceedins of the International Conference on Mobile Computing and Networking (MobiCom)*, August 1999.

[11] G. Pottie and W. Kaiser, "Wireless integrated network sensors," *Communications of the ACM*, vol. 43, Issue 5, May 2000.

[12] J. Lilley, J. Yang, H. Balakrishnan and S. Seshan, "A unified header compression framework for low-bandwidth links," in *6th International Conference on Mobile Computing and Networking (MobiCom)*. ACM, August 2000.

[13] V. Jacobson, "Compressing TCP/IP headers for low-speed serial links,RFC-1144," February 1990.

[14] M. Degermark, M. Engan, B. Norgreen and S. Pink, "Low-loss TCP/IP header compression for wireless networks," in *International Conference on Mobile Computing and Networking (MobiCom)*. ACM, November 1996.