Poster Abstract: Rime — A Lightweight Layered Communication Stack for Sensor Networks

Adam Dunkels, Swedish Institute of Computer Science
adam@sics.se

***Abstract*— Early work in sensor networks found traditional layered communication architectures too restrictive and proposed cross-layer optimizations. Recent work in data aggregation, however, argues that the complexity of cross-layer optimizations may lead to fragile and unmanageable systems. This has inspired us to create Rime, a layered communication stack for sensor networks, with much tinner layers than traditional architectures. Rime is designed to simplify the implementation of communication protocols. A preliminary evaluation suggests that Rime may be able to significantly reduce the implementation complexity of sensor network protocols with only a small increase in resource requirements, hinting that a layered stack may be a suitable communication abstraction even for sensor networks.**

I. Introduction

Early work in sensor networks found that traditional layered communication architectures were too restrictive for sensor networks [5]. Instead, radical cross-layer optimizations were investigated, where e.g. high-level abstractions were implemented at a low level [8]. Recent work in data aggregation [6], however, argues that complex cross-layer optimizations for data aggregation leads to fragile and unmanageable systems. Instead, a more traditional, layered, architecture is proposed and found to be nearly as efficient as cross-layer aggregation architecture. This move towards a more traditional architecture for data aggregation have inspired us to revisit layered communication abstractions for sensor networks.

The contribution of this paper is Rime, a lightweight layered communication stack for sensor networks. Rime is different from traditional layered network architectures such as the Internet architecture in that the layers in Rime are unusually thin. Rime draws heavily from communication abstractions for distributed programming [7] where layers of simple abstractions are combined to form powerful high-level abstractions.

The purpose of Rime is to simplify implementation of sensor network protocols and facilitate code reuse. We have implemented Rime in Contiki [3] and a preliminary evaluation suggests that Rime can significantly simplify protocol implementation with only a small increase in resource requirements. The code footprint of Rime is less than two kilobytes and the data memory requirements on the order of tens of bytes.

Rime is designed to be much simpler than existing proposals for modular communication abstractions for sensor networks [4], [9]; Rime does not allow for a fully modular structure where any module can be replaced, but enforces a strict layering structure where only the lowest layer and the application layer can be replaced.

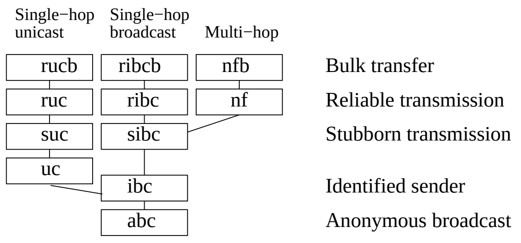

Fig. 1. The current Rime stack. More protocols and layers may be added.

II. Rime

Rime is organized in layers as shown in Figure 1. The layers are designed to be extremely simple, both in terms of interface and implementation. Each layer adds its own header to outgoing messages. Because Rime layers are simple, individual headers are very small; typically a few bytes each.

The thin layers in Rime enable code reuse within the stack. For example, reliable communication is not implemented in a single layer but divided into two layers, one that implements acknowledgments and sequencing, and one that resends messages until the upper layer tells it to stop. We call the latter layer *stubborn*. A stubborn layer is not only used by reliable protocols but also by protocols that send periodic messages such as neighbor maintenance for routing protocols and repeated transmission of messages in Rime's network flooding layer (nf). Figure 2 shows how a hop-by-hop reliable data collection protocol implemented with Rime's stubborn identified best effort broadcast, sibc, and reliable unicast, ruc.

The lowest level primitive in Rime is anonymous best-effort broadcast, abc. The abc layer provides a 16-bit channel abstraction but no node addressing; it is added by upper layers. The identified sender best-effort broadcast, ibc, adds a sender identity header field and the unicast abstraction, uc, adds a receiver header field.

An underlying MAC or link layer may chose to implement parts of the Rime stack, such as the abc, ibc, or uc layers, in hardware or firmware.

A. Shifting the Burden from Applications to the System Core

One of the basic ideas of Rime is to shift the burden, in terms of memory footprint, from protocol implementations to Rime. By making Rime part of Contiki's system core, which is always present in memory, loadable programs are made smaller. Consequently, the energy consumption for program loading [2] is reduced.

```
sibc_received(node_id from) {
  neighbor_add_or_update(from, signal_strength());
  recalculate_hops_from_sink();
}

ruc_received(node_id from) {
  if(we_are_the_sink) {
    print_to_serial(packet_data);
  } else if(we_have_a_route) {
    ruc_send(neighbor_best(), packet_data);
  }
}

datacollection_process() {
  ruc_setup(DATA_CHANNEL);
  sibc_setup(NEIGHBOR_CHANNEL);
  sibc_send(hello_message);
  while(1) {
    wait_until(sensor_event);
    ruc_send(neighbor_best(), sensor_data);
  }
}
```

Fig. 2. Hop-by-hop reliable data collection protocol implemented with Rime.

B. *Buffer Management*

To reduce memory footprint Rime uses a single buffer for both incoming and outgoing packets similar to uIP [1]. Layers that need to queue data, e.g. a stubborn protocol or a MAC layer, copy the data to dynamically allocated queue buffers.

III. Preliminary Evaluation

A. *Memory Footprint*

The code memory footprint of individual Rime modules is small. The smallest module currently is ibc, identified sender best-effort broadcast, with a footprint of only 100 bytes. Stubborn unicast and reliable unicast are the largest with a code footprint of 226 bytes. The current total footprint of Rime is less than two kilobytes but will increase with more features.

Each Rime layer requires between two and four bytes of RAM per connection. The stubborn layers currently need an extra 18 bytes of RAM because of the retransmission timer, which currently requires 16 bytes of state. Much of this is, however, due to the current timer implementation in Rime not being optimized for a low memory footprint.

B. *Hop-by-hop Reliable Data Collection Routing Protocol*

We evaluate Rime by reimplementing Treeroute, Contiki's hop-by-hop reliable data collection routing protocol, with Rime. We compare the resulting lines of code and footprint with the existing implementation. The results suggest that Rime can significantly reduce the complexity of protocol implementations for sensor networks.

For this preliminary evaluation, the reimplementation does not fully conform to the existing implementation. The existing implementation uses implicit acknowledgments while the reimplementation uses explicit acknowledgments because we have not yet implemented implicit acknowledgments in Rime.

The results of the reimplementation are shown in Table I. Lines of code are measured without comments. Both implementations are compiled with GCC 3.2.3 for the MSP430 microcontroller. The code footprint for the reimplemented protocol does not include the code footprint of the Rime modules.

TABLE I
EXISTING IMPLEMENTATION AND REIMPLEMENTATION OF CONTIKI'S TREEROUTE DATA COLLECTION ROUTING PROTOCOL.

	Existing	With Rime
Lines of code	439	179
Code footprint (bytes)	2064	772
Memory footprint (bytes)	180	110
Packet header size (bytes)	8	10

The memory footprint does, however, include the memory footprint of the Rime modules. The table shows a significant reduction in program size and a slight increase in header size. One of the extra header bytes for the reimplementation is due to the use of explicit acknowledgments. The second extra byte is because the ruc layer uses an entire byte for a single-bit flag. We expect that a header packing mechanism that we plan to implement will reduce the header size.

Although the reduction in code and memory footprint for the reimplementation of the data collection protocol module is significant, the total code footprint for the data collection protocol and the Rime modules is slightly larger than that of the existing data collection implementation. In a system using Rime the footprint of the Rime modules is, however, amortized over all protocols and applications using Rime. Therefore we expect that, overall, Rime will reduce the total memory and code footprint for systems using Rime.

IV. Conclusions

Our preliminary evaluation suggests that Rime may be able to significantly reduce complexity of sensor network protocol implementations, with only a small increase in resource requirements. If these results would continue to hold true for a more thorough evaluation, this suggests that a layered stack could be a suitable communication abstraction even for wireless sensor networks.

Acknowledgments

This work was partly financed by VINNOVA and the European Commission under contract IST-004536-RUNES.

References

[1] A. Dunkels. Full TCP/IP for 8-bit architectures. In *MobiSys'03*, 2003.
[2] A. Dunkels, N. Finne, J. Eriksson, and T. Voigt. Run-time dynamic linking for reprogramming wireless sensor networks. In *ACM SenSys'06*, 2006.
[3] A. Dunkels, B. Grönvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *IEEE Emnets-I*, 2004.
[4] C. Ee, R. Fonseca, S. Kim, D. Moon, A. Tavakoli, D. Culler, S. Shenker, and I. Stoica. A modular network layer for sensornets. In *Proceedings of OSDI*, November 2006.
[5] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. In *Proceedings of ACM/IEEE MobiCom*, August 1999.
[6] O. Gnawali, K. Jang, J. Paek, M. Vieira, R. Govindan, B. Greenstein, A. Joki, D. Estrin, and E. Kohler. The tenet architecture for tiered sensor networks. In *ACM SenSys '06*, November 2006.
[7] R. Guerraoui and L. Rodrigues. *Introduction to Reliable Distributed Programming*. Springer, 2006.
[8] J. S. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan. Building efficient wireless sensor networks with low-level naming. In *Symposium on Operating Systems Principles*, 2001.
[9] J. Polastre, J. Hui, P. Levis, J. Zhao, D. Culler, S. Shenker, and I Stoica. A unifying link abstraction for wireless sensor networks. In *SenSys*, 2005.