Developing a Graphical User Interface for Improved ION DTN Node Operations

Matthew Wu
Mentors: Leigh Torgerson, Philip Tsao

August 25, 2016

Abstract

Disruption-tolerant networking (DTN) is an approach to networks that can withstand drops in connectivity and long transmit times. One important application of DTN is an interplanetary Internet among the numerous spacecraft in the Solar System. The Interplanetary Overlay Network (ION) is an implementation of DTN developed at JPL. To allow for network monitoring, ION nodes write status outputs to local files and `stdout`. Previously, ION network monitoring was done by remotely viewing each node's status outputs in terminal windows, which quickly became cumbersome for large networks. In this project, we prototyped a more effective system for monitoring ION networks. First, each node sends its status outputs to a remote machine via TCP. Then, an application on the remote machine reads the status outputs from TCP, stores them in a database, and displays them in a flexible, user-friendly graphical user interface. The application was built using Logstash (log pipeline), Elasticsearch (database) and Kibana (graphical front-end), which are open-source and support Windows, Mac, and Linux.

1 Introduction

Disruption-tolerant networking (DTN) is an approach to creating networks that are resistant to delays and disruptions. The protocols that underlie the terrestrial Internet, like TCP/IP, perform poorly in networks with many disruptions [1]. For example, if a complete path between sender and receiver is not available, any packets on the way are dropped. DTN uses special protocols such as the Bundle Protocol (BP), documented in RFC 5050 [2], to achieve delay and disruption tolerance. For example, BP provides store-and-forward behavior: when the sender is unable to connect to the next hop in the route to the receiver, the sender can store the packets until a connection to the next hop is reestablished.

A major application of DTN is space communication [1]. Currently, virtually all communication with spacecraft in deep space is point-to-point or through a single relay satellite [1]. However, an Internet-like communication network among these spacecraft

may provide several benefits, such as more uplink and downlink routes to any individual spacecraft, and reuse of communication software and hardware across missions [3]. Since deep space presents both long delays (significant light-time delays between distant spacecraft) and frequent disruptions (say, a planet or asteroid moves in the way), DTN is necessary to realize such an interplanetary Internet.

DTN also has many potential terrestrial applications, such as in military situations when radio signals are jammed, or in any situation where communications are strained [1, 4].

Several software implementations of DTN exist. The Interplanetary Overlay Network (ION) is a DTN implementation specifically designed for embedded systems such as spacecraft computers [5]. ION is developed by the Jet Propulsion Laboratory (JPL) and written in C.

1.1 Network monitoring and control

Network monitoring and control is an important component of any network, including disruption-tolerant networks (DTNs). "Network monitoring" includes ensuring that nodes and links are operational and measuring network performance, while "network control" includes changing network topology and performing troubleshooting [6].

A forthcoming standard for monitoring *and* controlling DTNs is the Asynchronous Management Protocol (AMP), documented in the Internet Draft draft-birrane-dtn-amp-03 [7]. However, as of the time of this project, an implementation had not yet been released [8]. Thus, only non-standard methods of monitoring DTNs were available.

In ION, each node produces two types of status outputs, logs and watch characters, which can be used for network monitoring. Logs are one-line text messages about ION's current operational status, written to the node's local file `ion.log`. Watch characters are individual characters that represent steps in the DTN protocols and are printed to `stdout` as the steps are completed.

Previously, ION network monitoring was done by using terminals to SSH (Secure Shell) into each node to view `ion.log` and `stdout` [8]. Because multiple terminal windows were needed for each node, this quickly became cumbersome for large networks. Also, this implies SSH privileges into each node are required to monitor the network. Such a high level of privileges should only be entrusted to a few users, but many more people may be interested in monitoring the network.

The goal of this project is to develop an improved system for monitoring ION networks.

2 Methods

Our design for an improved network monitoring system can be divided into two parts. First, each ION node sends its logs and watch characters over TCP (i.e. the terrestrial Internet) to a remote machine. Second, an application on the remote machine receives the logs and watch characters from TCP and displays them in a GUI.

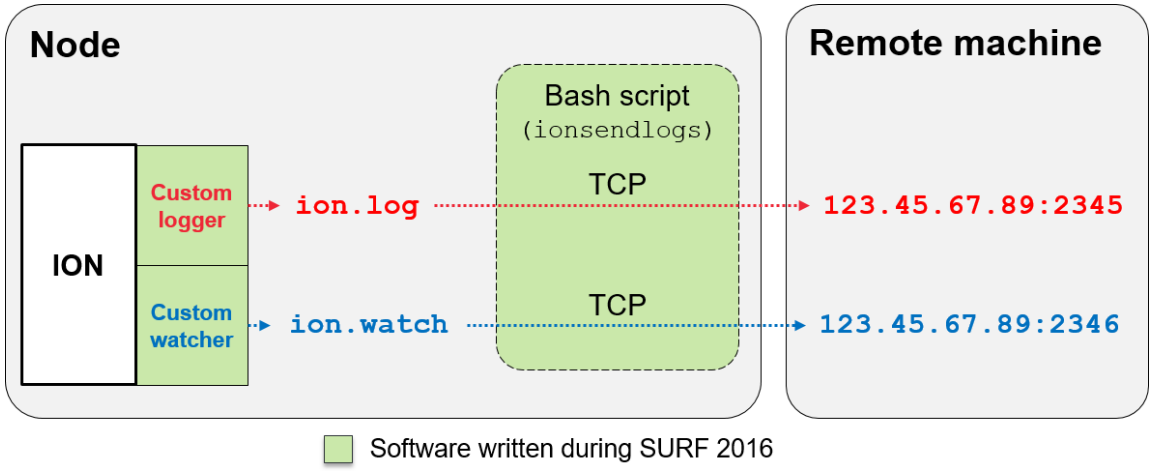

Figure 1: Block diagram for how each node's logs and watch characters are sent via TCP to a remote machine.

Since logs and watch characters are sent using the terrestrial Internet, this project is intended for monitoring ION nodes which are still on Earth, and not nodes in space.

2.1 Sending each node's logs and watch characters over TCP

In normal ION, all log messages are passed to a *logger*, which writes the logs to the local file `ion.log`. Likewise, all watch characters are passed to a *watcher*, which prints the characters to `stdout`. However, one can use a custom-written logger and watcher by recompiling ION with the custom logger/watcher files included and some compile flags set. (Editing or patching the ION source code is not necessary.)

Our first attempt was to use a custom logger and watcher that sends logs and watch characters directly to TCP sockets. However, ION is single-threaded, so the node's DTN operations must wait for the logger and watcher to finish sending to TCP. If the connection is broken (e.g. the remote machine goes down), the node's DTN operations may hang.

Our current approach is depicted in Figure 1. The custom logger writes to the local file `ion.log`, while the custom watcher writes to the file `ion.watch`. A separate Bash script monitors these files and sends newly appended lines over TCP. This way, TCP issues may halt the Bash script, but the node's DTN operations will continue.

The custom logger writes logs to `ion.log` in the format "`<unix-ms> <text>\n`". `<unix-ms>` is the Unix time in milliseconds, `<text>` is the log message text, and `\n` indicates that log messages are separated by newlines. For example, "`1468429103596 [i] rfxclock has ended.`"

The Bash script prepends a string called the *node id* to each log message before it is sent. For instance, if the node id is `27`, the previous example is *sent* as "`27 1468429103596 [i]`

Table 1: Format of log and watch messages in `ion.log` and `ion.watch`, and when being sent

Type	Format
Log, in `ion.log`	`<unix-ms> <text>\n`
Watch, in `ion.watch`	`<unix-ms> <char>\n`
Log, when sent	`<node-id> <unix-ms> <text>\n`
Watch, when sent	`<node-id> <unix-ms> <char>\n`

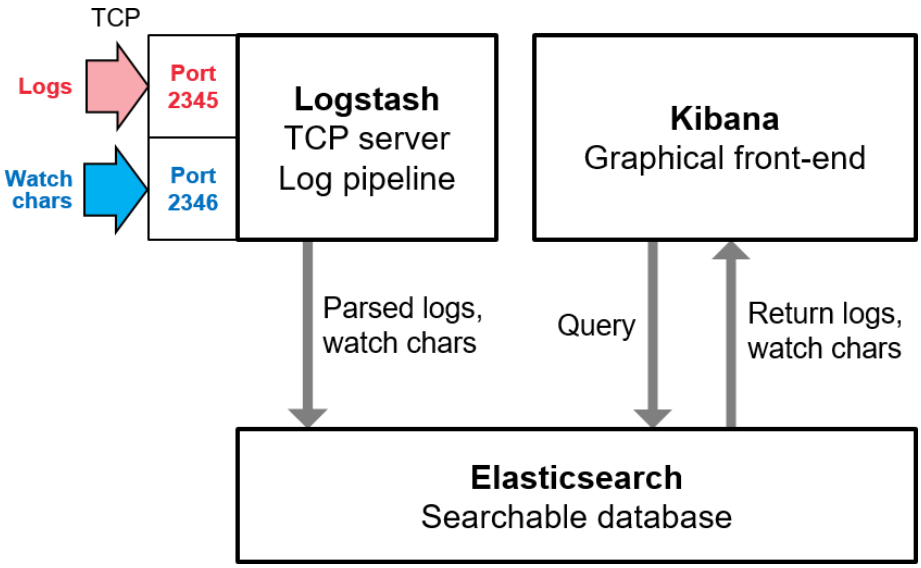

Figure 2: Block diagram for the network monitoring application on the remote machine.

`rfxclock has ended.`" The node id is provided as an argument to the script, and each node should use a unique node id. Since many nodes may send logs to the same TCP socket on the remote machine, the node id allows each message's sender node to be identified.

Watch characters are written to `ion.watch` in the format `<unix-ms> <char>\n`, and the Bash script also prepends the node id to each watch message before sending it.

The formats of log and watch messages are summarized in Table 1.

2.2 Network monitoring application on remote machine

For the network monitoring application, we used the Elasticsearch, Logstash, and Kibana (ELK) stack. These are open-source, cross platform (Windows, Mac, Linux) tools developed by the Elastic company.

- Elasticsearch is an easily searchable document store.

- Logstash is used to receive incoming logs/watch characters from TCP sockets, parse them, and store them in Elasticsearch.
- Kibana is a graphical front-end, used to visualize log and watch character data pulled from Elasticsearch.

A diagram for how they work together is depicted in Figure 2.

Elasticsearch, Logstash, and Kibana had been used successfully for JPL's SMAP mission [9] and seemed suitable for our needs. By using already-existing tools, we avoided the difficulties of developing a custom application on our own.

3 Results

We prototyped an application for monitoring ION nodes, with a database and a graphical user interface.

3.1 Database

Elasticsearch is used as a database. Logs and watch characters are stored in Elasticsearch in individually dated indices of the format `log-<yyyy.mm.dd>` and `watch_char-<yyyy.mm.dd>` (e.g. `log-2016.08.22` and `watch_char-2016.08.22` for logs and watch characters from August 22nd, 2016).

3.2 Graphical user interface

Kibana, querying data from Elasticsearch, provides the graphical user interface.

Two dashboards, or windows with pre-arranged visualizations, were created in Kibana. The "Overview" dashboard (Figure 3a) lists incoming log messages and counts certain watch characters, giving an overview of network usage. The "Watch char" dashboard (Figure 3b) plots counts over time of almost all watch characters, with separate plots for each node.

Dashboards can be restricted to displaying logs and watch characters from any time interval in the past, or an interval relative to "now" such as "the last 10 minutes." Kibana can also auto-refresh, or re-query Elasticsearch for data, at regular intervals such as every 5 seconds. If a now-relative time interval is used with auto-refresh, dashboards will appear to update automatically to incoming logs and watch characters.

It is easy to switch between dashboards and create new ones. In a dashboard, visualizations can easily be moved, resized, modified, or removed. New visualizations can easily be created, including area, line, pie, and bar charts.

Kibana's Discover tab allows Google-like full-text search for logs and watch characters stored in Elasticsearch.

Much more information about using Kibana is available in the Kibana User Guide on the Elastic website [10].

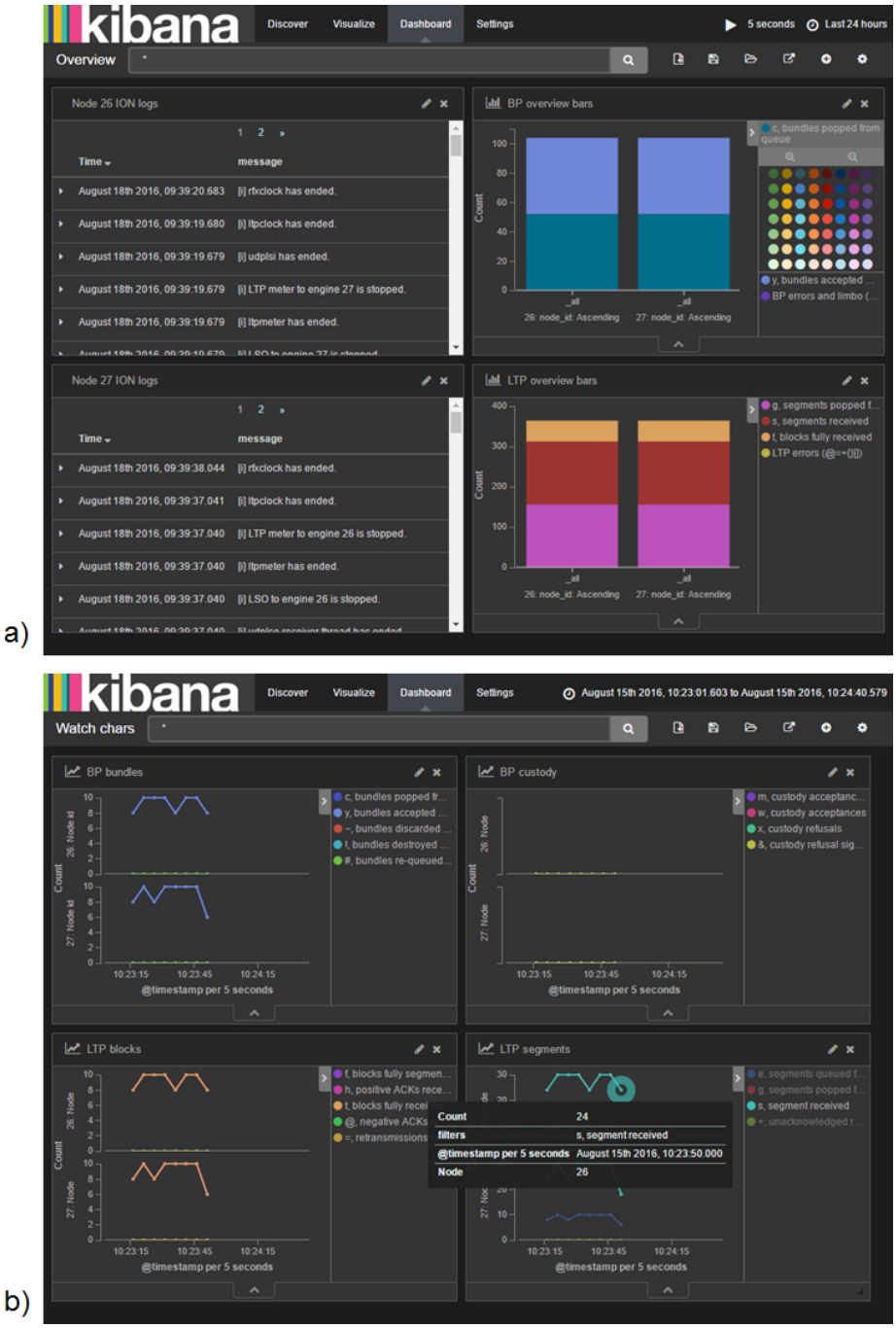

Figure 3: Screenshots of a) Overview and b) Watch char dashboards in Kibana.

4 Discussion

Our system represents an improved, more user-friendly way to monitor ION networks. Because each node's logs and watch characters are sent to a remote machine, there is no need to SSH into each node to view `ion.log` and `stdout`.

Since logs and watch characters are stored in a database, they can be searched and referenced in the future. If the database gets too large, older data can easily be archived, since data from separate days are stored in separate indices. We suggest using the open-source Node.js tool `elasticdump` [11] to archive old data.

Kibana provides an attractive and user-friendly GUI to monitor ION networks. In particular, Kibana's line and area charts are a much more effective way to display watch characters than viewing a node's `stdout` in a terminal window, which usually results in an overwhelming stream of letters.

The "Overview" and "Watch char" dashboards were created as starting points and examples of what can be done in Kibana. One of the tool's greatest strengths is its flexibility, and visualizations and dashboards should be customized to meet the specific needs of each network.

5 Conclusion

Previously, ION network monitoring was done by using terminals to SSH into each node to view logs and watch characters. We prototyped an improved network monitoring system, where all logs and watch characters are sent to a central machine, which displays them in a user-friendly GUI. Overall, our system is easier to use and provides advanced features. Because of Kibana's ease of customization, our system can adapt to many DTN setups.

Future work includes continued deployment and testing to refine the system. Additionally, it may be easy to extend the system to monitor not only ION logs and watch characters, but also node CPU usage, network usage, and other statistics that may be useful to those monitoring the network. We may also seek to extend the system to support AMP when its implementation is released.

6 Acknowledgments

Thanks to Leigh Torgerson, Philip Tsao, Scott Burleigh, Greg Miles, Rick Borgen, Loren Clare, Allen Kim, and Terry Suh for their support.

Thanks also to the JPL Education Office and Caltech's Student-Faculty Programs for making this summer opportunity possible.

References

[1] Jason Soloff. Disruption Tolerant Networking (DTN): An Architecture for Challenged Communications. IEEE WISEE 2015, December 2015.

[2] Keith Scott and Scott C. Burleigh. Bundle Protocol Specification. RFC 5050, October 2015.

[3] Disruption Tolerant Networking Project, NASA. `http://techport.nasa.gov/view/11772`. Accessed: 2016-08-23.

[4] Forrest Warthman. Delay- and Disruption-Tolerant Networks (DTNs): A Tutorial. Technical report, Warthman Associates, September 2015.

[5] Scott Burleigh. Interplanetary Overlay Network (ION): What's New. InterPlanetary Networking Conference, May 2015.

[6] J. Leigh Torgerson. Network Monitor and Control of Disruption-Tolerant Networks. In *Proceedings of the SpaceOps 2014 Conference*, 2014.

[7] Edward J. Birrane and Jeremy Pierce-Mayer. Asynchronous Management Protocol. Internet-Draft draft-birrane-dtn-amp-03, Internet Engineering Task Force, June 2016. Work in Progress.

[8] Communication with Leigh Torgerson.

[9] Communication with Rick Borgen.

[10] Kibana User Guide. `https://www.elastic.co/guide/en/kibana/current/index.html`. Accessed: 2016-08-22.

[11] elasticdump. `https://github.com/taskrabbit/elasticsearch-dump`. Accessed: 2016-08-23.