RIC Proxy User Guide

Remote Instrument Control

The RIC Proxy allows a user interface control program to communicate via the Internet with an instrument deployed on NCAR aircraft . This document covers installation and operation of the software. System design and configuration are also addressed.

18 June 2015

Authors:

- Charlie Martin

NCAR
NATIONAL CENTER FOR ATMOSPHERIC RESEARCH

Contents

Figures

Tables

1 Quick Start

Substitutions:

- <HOME>: refers to the user's home directory. (On W7, this is *C:\Users\<user name>*.)
- <XXXX>: refers to the RIC software revision number, e.g. 6983.
- <INST>: is the name of your instrument, e.g. AMAX, or AVAPS.

The program will be installed in a folder created in the home directory. The configuration files will be created in a hidden folder[1] named *.ric*, also located in the home directory.

In the following, "*.ric/*" refers to *<HOME>/.ric.*

1.1 Software Installation

1. Fetch the RIC Proxy installer

   Download from: www.eol.ucar.edu/software/ric

   The file will be named Proxy-XXXX.exe or Proxy-XXXX.dmg.

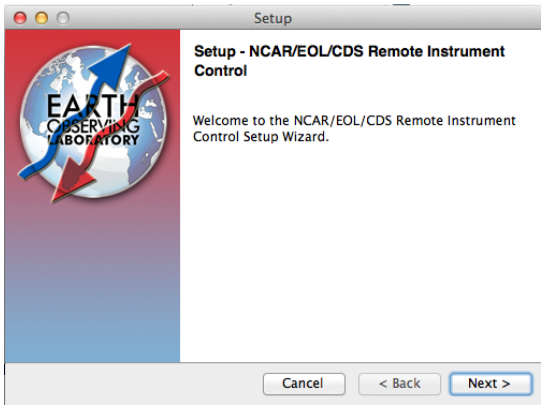
   (On OSX, double click to open the .dmg file, which will contain the installer file Profiler-XXXX.app)

2. Install the software

   Double click the installer and follow the instructions.

   The program will be installed in <HOME>/RIC-XXXX.

   Initial configuration directories and files will be created in *.ric.*

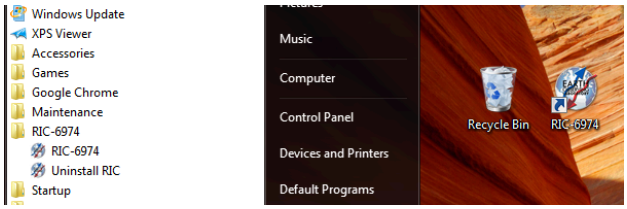


[1] Folder names that begin with a period are sometimes treated as "hidden", in that they won't appear in file browsers. Linux users will be familiar with this; OSX users less so. To edit a configuration file on OSX, start the text editor, and select the file open menu item. Simultaneously press Shift-Command-. to display the hidden folders.

[2] It is best to run the Proxy on the same computer that the user interface program is running on. The Proxy can be run on a different machine, but dynamic host naming can make this problematic. If the user control program machine has a static IP address or name, use that for

3. Shortcuts

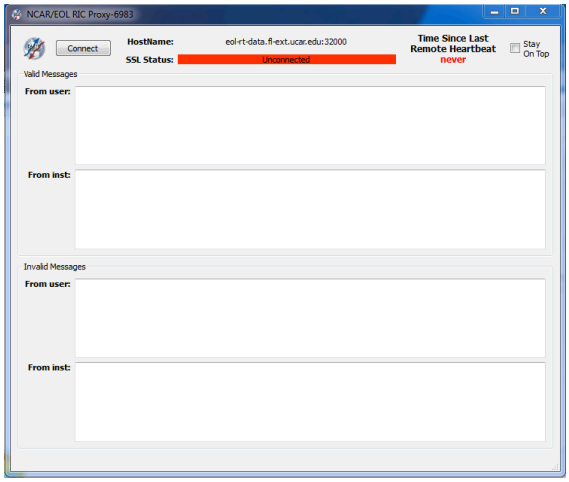
Windows: A shortcut for RIC-XXXX will be placed on the desktop, and a corresponding folder with shortcuts will be placed in the Start menu.



OSX: Drag the application name from the installation folder to the OSX dock.

4. Run the proxy

The main window will appear.



5. Click on the Connect button

The status bar at the bottom of the Proxy should read: “Remote host closed the connection”. This happens because your Proxy has not yet been configured with the correct certificates.

6. Obtain your key and certificates from NCAR

NCAR will arrange a secure method to deliver three files. These files must not be made public, as they will allow anyone to connect to the RIC servers.

Three files will be provided, e.g:

eol-rt-data.crt
AVAPS.crt
AVAPS.key

7. Authorize your Proxy

Copy the key and certificate files into .ric/Certs/

You may overwrite existing files of the same name.

8. Identify your instrument

Edit .ric/Proxy.ini

Change the configuration values to match your instrument name:
ProxyID: The instrument name.
and
InstrumentFile: The instrument file, which will be of the form <INST>.ini.

```
.
.
[General]
# The proxy identifier
ProxyID=AVAPS
.
.
.
InstrumentFile=AVAPS.ini
```

9. Create a profile for your instrument

Copy .ric/Instruments/SAMPLEINST.ini to .ric/Instruments/<INST>.ini

E.g., create .ric/Instruments/AVAPS.ini

10. Customize your instrument profile

Edit your instrument file:
(.ric/Instruments/<INST>.ini)

***InstName**:* the instrument name. This will match the instrument key and certificate file names (.ric/Certs/<INST>.key and .ric/Certs/crt).

The other *Inst…* parameters are not relevant to a Proxy configuration, and can be ignored.

***UserIncomingPort**:* the port that the user interface control program sends datagrams to. The proxy will listen on this port.

***UserDestPort**:* the port that the user interface control program receives datagrams on. The proxy will send messages here.

UserHostName: the host that the user control program is running on. Usually it is running on the same machine as the Proxy, in which case 127.0.0.1 should be used[2].

Add the message identifiers to the *[Messages]* section:
***ID**:* is the message identifier.
RateLimit: the time in seconds during which one message will be allowed. For instance, a value of *2* means that one message will be allowed through every two seconds. The value can be less than one. A value of zero disables rate limiting, allowing all messages through.

Be sure to set the ***size*** value to the number of messages defined in the *[Messages]* section.

```
[General]
InstName=AVAPS

# Instrument network parameters
(EmbeddedProxy)
InstIncomingPort=10120
InstDestPort=10121
InstHostName=127.0.0.1

# User network parameters (SSLProxy)
UserIncomingPort=10121
UserDestPort=10120
UserHostName=127.0.0.1

[Messages]
1\Broadcast=false
1\ID=AVAPS-GV
1\RateLimit=0

2\Broadcast=false
2\ID=AVAPS-ENGR-GV
2\RateLimit=5

3\Broadcast=false
3\ID=AVAPS-LOG
3\RateLimit=5

size=3
```

The configuration is now complete, and you should be able to connect to the NCAR/EOL server. Connection failures will occur if the certificates have expired, or do not match those on the server.

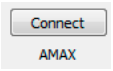
[2] It is best to run the Proxy on the same computer that the user interface program is running on. The Proxy can be run on a different machine, but dynamic host naming can make this problematic. If the user control program machine has a static IP address or name, use that for *UserHostName*. If the dynamic IP address is fairly stable, you can change the configuration as needed. Finally, if your network allows it, you can set the broadcast flags to true, and use the network broadcast address (usually setting the last octet to 255) to cause the messages to be broadcast to the user interface system.

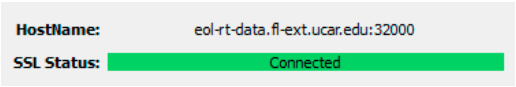
1.2 Proxy Operation

Click on the Connect button

The SSL Status should turn green and indicate a successful connection to the EOL server. If unsuccessful, an error message will appear in the status bar at the bottom.

It is possible to connect to the EOL server even when the aircraft is offline.





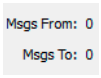
Heartbeat

The heartbeat timer indicates that communications to the aircraft are active, by showing how long since the last aircraft heartbeat was received. After 10 minutes, the elapsed time will turn red. If the aircraft is not connected, the elapsed time will show “Never”, or just continuously increment.

It is common for the satcom link to drop for significant time periods during a flight.

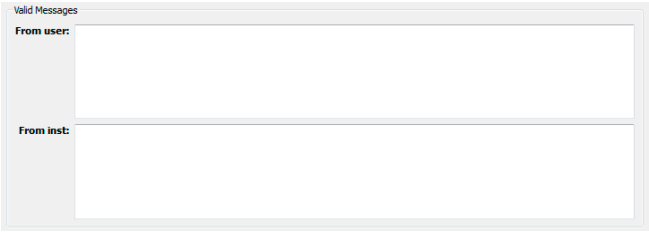
Msgs From / Msgs To

An updating count of messages coming from the instrument and sent to the instrument is displayed.



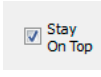
Message traffic

Messages that transit the proxy are displayed in the “Valid Messages” box. Unexpected messages are displayed in the “Invalid Messages” box.



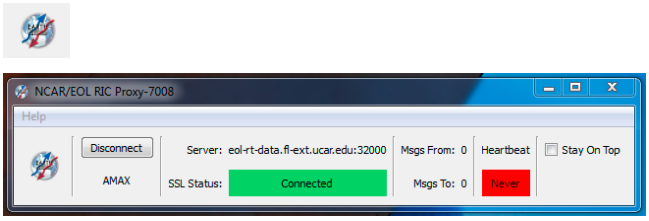
Stay on top

Selecting the “Stay On Top” checkbox will prevent the Proxy window from being hidden by other windows.



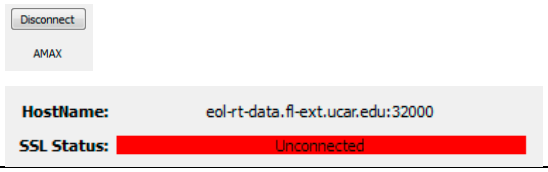
Mini-View Toggle

Click on the EOL logo to collapse or expand the message display panels.



Click on the Disconnect button

The Proxy will disconnect from the EOL server.



1.3 Software Uninstall and Reinstall

The RIC Proxy can be uninstalled by running the uninstaller application located in the installation directory (Windows and OSX), or by simply removing the installation directory (Linux).

The configuration files are not removed during the uninstall process.

Newer versions of the software can be installed at any time. The existing configuration files will be preserved. It is completely acceptable to have multiple versions installed at the same time. They will all use the same configuration files.

2 Background

The NCAR/EOL Remote Instrument Control (RIC) system provides a mechanism for users to control their scientific instruments via the Internet. The targeted use supposes that there will be at least one low bandwidth and unreliable communication link in the communication path, such as a satellite link to an aircraft.

Key design concepts of RIC include:

- The instrument control software is completely separate from the user interface program.
- The instrument and user communicate via “small” messages delivered via the network.
- The messaging protocol is “stateless”: neither the instrument nor the user need retain prior knowledge of each other’s state. (*More on this later*.)
- The developer is able to develop and test in their own lab, and then switch to remote control simply by running the RIC proxy program, and perhaps changing an IP address/port pair.

The target users of RIC are scientific instrument operators whose systems are capable of being remotely operated. RIC is appropriate for instruments that do not require a collocated human operator.

The instrument software should be capable of describing the operational status by transmitting a relatively compact message. Control is performed by sending short command messages to the instrument. The instrument can also transmit summary observational data.

Many instruments generate high resolution and bandwidth observations, which cannot be transferred via a slow link. However, it is usually possible to produce averaged and compressed summaries, or simple “snapshots”, that can convey the general phenomena being observed and the overall performance of the instrument.

RIC was initially designed for communications to the NCAR aircraft, but there is nothing preventing its use for other applications.

3 Design

3.1 Basics

RIC is essentially a secure message passing system, implemented with a few communicating software components.

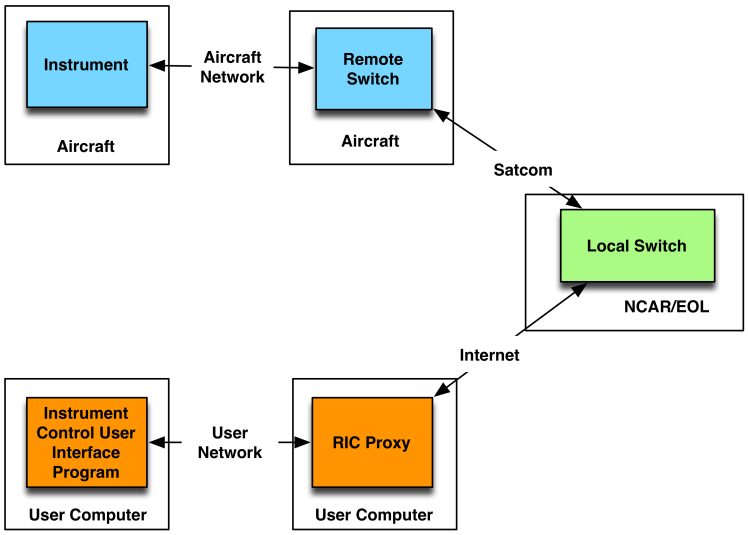


Figure 1. RIC Software Components

Some of the software components are provided by the instrument developer, and others are part of the RIC system. The functions of these components are:

- Instrument Control User Interface: The user interface program, provided by the instrument developer. It sends commands to the instrument and interprets messages received from the instrument.
- RIC Proxy: An NCAR provided program that masquerades as the instrument. It passes messages, via the Internet, between the user and ground based Local Switch. The proxy and the switch provide secure communications over the Internet.
- Local Switch/Aircraft Switch: This pair of programs provide secure message passing via the satellite communication link, between the ground and the aircraft.
- Instrument: The instrument software, running on the airborne instrument, sends and receives messages from the aircraft switch.
- Instrument: The remote instrument data acquisition and control software, created by the instrument developer.

Because messages are passing over the Internet, a secure method is required for the communication between the proxy and the local switch. This is achieved with the standard Secure Socket Layer (SSL) protocol. The proxy has its own user interface that is able to show when a successful connection has been made (or lost) to the local switch, monitor message traffic, and other functions.

The communication between the two switches is via a low bandwidth and relatively unreliable satellite link. This link can and often disconnects during regular operations, and the time that it takes to restore the link, a connection-oriented protocol such as SSL is prohibitive. For this reason, message passing between the two switches is implemented with encrypted datagrams. The satellite link can drop out, and the two switches will blindly continue to send datagrams to each other. *This explains the mandate that the instrument message protocol must be state-less:*

neither the instrument nor the user interface should assume that the opposite end has received their sent messages.

The two switches can also pass RIC status messages between them. One such message is a heartbeat message, which indicates that the satcom link is up. These heartbeats are passed onto the proxies, allowing them to indicate that there is a live and accessible aircraft system.

3.2 Typical Configurations

RIC was designed so that the deployment of the various software components is very flexible.

Another key requirement is that it be trivial for an instrument developer to migrate from development and testing of the instrument in a lab environment to a remote operation scenario. This is accomplished by using simple datagrams as the communication mechanism between the user and the instrument. We are simply substituting the proxy for the instrument during remote control. The following figure depicts various user/instrument combinations.

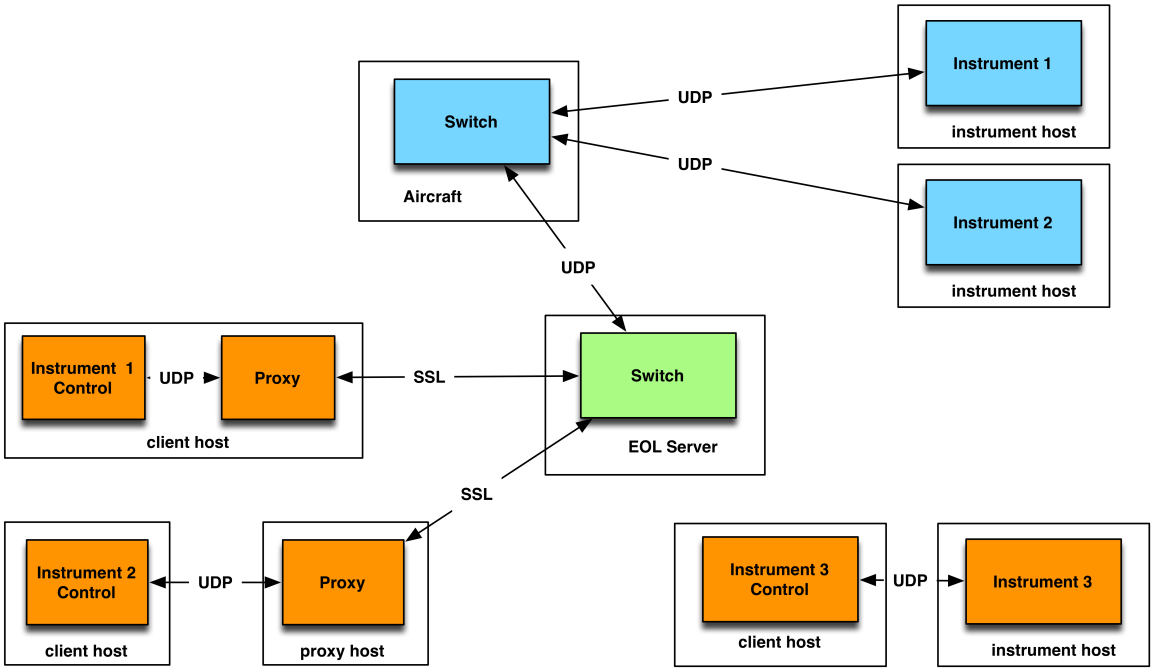


Figure 2. RIC Deployment Configurations

The lower right shows a typical lab development environment, where the control interface and the instrument talk directly to each other. Although two host computers are shown, both software packages are typically running on the same host.

Two remote scenarios are shown in the lower left, where the proxy has replaced the instrument. The only difference in the two cases is whether the control and proxy software are located on a common or separate computers. The proxy takes the place of the local instrument, and exchanges messages with the switch.

Onboard the aircraft, shown in the upper right, the instruments exchange messages with the switch. To the instrument, the switch appears as if it is the user instrument control program.

3.3 Switches

As shown in Figure 2, there is a single pair of switch programs that manage the messages between the ground and the aircraft, for all instruments. The switches are configured by EOL in order to authorize particular instruments. Operational limits (e.g. rate limiting) are part of the switch configuration. SSL certificates (with expiration dates) are managed by the ground switch and are used to establish secure communications with the proxies.

The switches also log proxy connections and message traffic statistics. The switch can be re-configured if necessary to lock out a previously authorized user, in order to limit disruptions caused by rogue instruments, control programs or hackers.

4 Guidelines for Instrument Developers

4.1 Development and Testing

All communication is done by UDP datagrams sent between your control program and your instrument. You can develop and test this in your lab. When it is time to deploy your instrument to the aircraft, you will simply point your control program at the RIC proxy. Likewise, on the aircraft your instrument messages will simply be directed to a computer on the aircraft. So, to go from lab testing to deployment, you may only need to change the destination IP address for the message datagrams.

4.2 Stateless Messaging Protocol

THIS IS CRITICAL: The command/status interaction should be stateless. This means that messages in either direction can be dropped, and the control program or instrument will "do the right thing" when they get the next message.

4.3 Message Definition

Communication between your control program and your instrument *must be text messages*, and each message must begin with a text identifier, followed by a comma. RIC does not care what is in the rest of the message; that is totally for your use.

The messages should reasonably small, simply to conserve bandwidth. Message sizes up to about 8000 bytes are currently supported. But messages of this size do not match the intentions of the system design; it is recommended to try to keep messages below 500 bytes. In any case, smaller is better.

Because satellite communication costs are significant, aggregate message bandwidth should be as low as possible. A couple of messages per second is a reasonable limit. The RIC system enforces message rate limiting, so that a message storm can't saturate the communication link. Remember that many users and systems are sharing the satcom link. The entire real-time operational data for the aircraft transit this link.

Have the instrument periodically send a status message that characterizes the health of the instrument. If there are a large number of parameters, message space can be saved by bit-mapping logical flags. Using comma-separated values can also make a status message more organized and easier to parse.

Create a few message identifiers, so that the rate limiting can be tailored to the communications situation. Standard practice is to have distinct message types for command, instrument status and summary data messages. Perhaps multiple summary data message types are appropriate. Think of ways to reduce the information to the bare minimum that allows you to assess the instrument performance as well as the observed phenomena.

Control messages from your instrument control program to the instrument should be succinct, and follow a logically designed protocol. Once again, both the control program and the instrument should take correct actions based on the most recently received message, with no knowledge of prior messages. *A useful exercise is to ask what your software will do if it receives any message, in any order.*

5 Proxy Configuration

This section describes the configuration scheme employed by RIC.

When the RIC Proxy software is installed, a configuration directory tree is created in the user home directory, /Users/<user name>/.ric. Sample configuration files are also installed:

.ric/

Certs/		
	eol-rt-data.crt	EOL server certificate
	SAMPLEINST.key	Sample user private key
	SAMPLEINST.crt	Sample user certificate
Instruments/		
	SAMPLEINST.ini	Sample instrument configuration
Proxy.ini		Proxy configuration

These configuration files are sufficient to run the Proxy, but it will not be able to connect to the EOL server. A few additional steps, described in the Quick Start (above) are required to configure the Proxy:

1. Change a few fields in Proxy.ini, to match your instrument name.
2. Obtain and install new EOL server certificate, as well as a key and certificate for your instrument.
3. Edit a new (and renamed) version of SAMPLEINST.ini, tailored to your instrument.

5.1 Proxy Configuration

Proxy.ini provides general configuration for the Proxy, and must always be located in the .ric/ directory, located in the user home directory.

Note that some fields (e.g. SSLCert) create a reference to a file with a very specific path. Fields that should not be modified from the distributed configuration files are indicated in the table.

Don't Change	Field	Default	Function
	ProxyID	SAMPLEINST	Assigns a name to the proxy. Change to the name of your instrument, e.g. AVAPS.
*	TopDir	~/.ric	The top directory for the configuration directory tree. "~" is shorthand for the user home directory.
*	SSLSwitchIP	eol-rt-data.fl-ext.ucar.edu	The EOL SSL server.
*	SSLPort	32000	The SSL port on the EOL server.
*	SSLCert	eol-rt-data	The name of the certificate for the EOL server. This identifies a certificate file named <TopDir>/Certs/<SSLCert>.crt.
	InstrumentFile	SAMPLEINST.ini	The instrument file, which should be named for your instrument, e.g. AVAPS.ini. This will identify an instrument file: <TopDir>/Instruments/SAMPLEINST.ini

Table 1. Proxy Configuration

5.2 Instrument Definition

The *InstrumentFile* entry in Proxy.ini specifies a file that defines the communication characteristics for your instrument. These are both networking parameters and message types.

The instrument file contains fields that are pertinent to the proxy as well as the switches, so that one file can be shared across RIC. Thus some fields will not be relevant to the proxy, but are used by one of the switches.

The following table describes the fields in an instrument definition file. The subsequent figure provides a graphical view of the example values.

Used by Proxy	Field	Example Value	
*	InstName	AVAPS	The instrument name. It should match the root name of the instrument definition file, and there must be corresponding key and certificate files in the Certs directory (e.g. AVAPS.key and AVAPS.crt)
	InstHostName	194.1.4.213	The IP name or address of the instrument.
	InstIncomingPort	10121	The port that instrument created messages will be received on.
	InstDestPort	10120	The port on *InstHostName* that will receive the user control messages.
*	UserHostName	127.0.0.1	The IP name or address of the user control computer. Typically it is the same system that the proxy is running on, and so 127.0.0.1 is specified.
*	UserIncomingPort	10120	The port the proxy will receive the user control messages.
*	UserDestPort	10121	The port on *UserHostName* that the instrument messages are sent to.
	[Messages]		
	1\Broadcast	false	If set to “true”, the message will be broadcast on the network.
	1\ID	AVAPS-STAT	The message identifier. Only messages that are identified will be processed by the system.
	1\RateLimit	5	The message rate limit, as the number of seconds between successive messages. Fractional second values are permitted. A value of 0 indicates no rate limiting. The switches perform the rate limiting.
	2\Broadcast	false	
	2\ID	AVAPS-CMD	
	2\RateLimit	0	
	size	2	The number of message types.

Table 2. Instrument Configuration

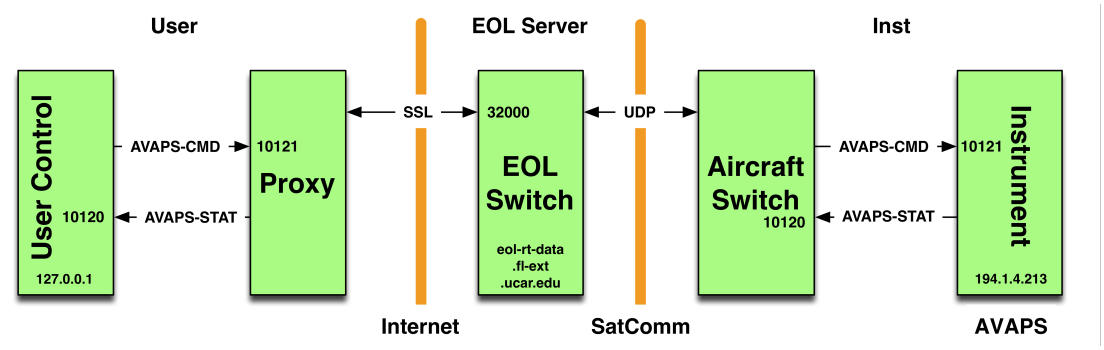


Figure 3. Example Network and Message Configuration