pRealm: Integrated Scoping of the MOOSDB

December 2020

Michael Benjamin, mikerb@mit.edu
Department of Mechanical Engineering
MIT, Cambridge MA 02139
`project-pavlab/appdocs/app_prealm`

1 Overview

The `pRealm` application is a tool to enable a shoreside multi-vehicle scope of the MOOSDB for all connected vehicles, as well as the shoreside MOOSDB. An instance of `pRealm` in each MOOS community, one per MOOSDB. It creates a shadow copy of the MOOSDB by registering for all known variables, i.e., those listed in the `DB_RWSUMMARY` published by the MOOSDB. The `pRealm` app only *listens* and keeps a copy of all the mail processed by the MOOSDB. Occasionally `pRealm` will receive and process a request to produce a report, a `REALMCAST`, showing the current values of a subset of known MOOS variables. Typically this subset is correlated to the subscriptions and publications of a particular app connected to the local MOOSDB. The typical usage scenario for pRealm is shown in Figure 1.

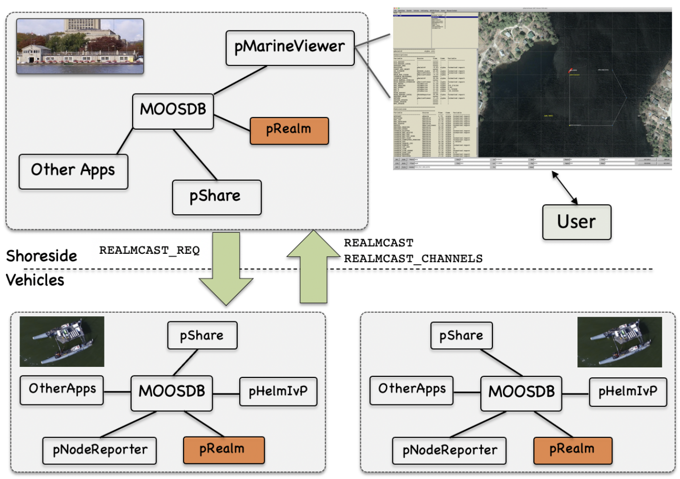


Figure 1: **Typical pRealm Topology:** A shoreside or topside community is receiving information from several deployed vehicles, in the form of node reports. The node reports contain time-stamped updated vehicle positions, from which the speed and distance measurements are derived and posted to the shoreside MOOSDB.

The goal of this application is to enable the user, situated at a shoreside console, to select any deployed vehicle and scope the MOOSDB from a variety of vantage points. Initially this means selecting a vehicle and process and seeing the publications and subscriptions of that process. This is especially powerful for confirming the flow of information between vehicles or between vehicles and the shoreside. The pMarineViewer app has been substantially augmented to capitalize on a set of MOOS communities that *realm enabled*. If pRealm is not enabled, pMarineViewer and all other apps will be unaffected.

The implementation of pRealm is similar to and based on *appcasting* which precedes pRealm by roughly seven years. It is similar to appcasting in that a realmcast is only generated when there is a client, e.g. pMarineViewer, with an open realmcast window, pointed to a particular vehicle and particular channel. In this case, pRealm on the chosen vehicle will only respond with a generated report solely for the selected channel. Like appcasting, the realmcast rate generation is immune time the MOOS Time Warp and will be generated roughly once per second. This regime, like with appcasting, minimizes bandwidth which may be limited in field operations.

Unlike appcasting, pRealm requires no augmentation to the participating apps. The only requirement is that an instance of pRealm is launched alongside all the apps otherwise launched in each MOOS community. No configuration is required. Although there are configuration parameters the user may adjust, the default parameters are fine in most cases.

2 Configuration Parameters for pRealm

The pRealm application may be configured with a configuration block within a MOOS mission file, typically with a `.moos` file suffix. Unlike most other MOOS apps, there will no warning generated if a configuration block is not provided. This is partly because the default values of parameters rarely need changing, and partly because not requiring a configuration block makes it even easier to add pRealm to existing missions. The following parameters are defined for pRealm.

Listing 2.1: Configuration Parameters for pRealm.

relcast_interval:	Time duration, in seconds, that must occur between the previously generated realmcast and a new one. The allowable range is [0.4, 15]. Values outside this range will be clipped to the range. The default value is 0.8 seconds.
summary_interval:	Time duration, in seconds, between auto-generated postings of REALMCAST_CHANNELS. The allowable range is [1, 10]. Values outside this range will be clipped to the range. The default value is 2 seconds.
wrap_length:	The number of characters, per line, when the user has opted to have the variable value field wrapped. Legal values are any integer value greater than zero. The default is 90.
trunc_length:	The number of characters that the value field of an output line will be truncated to, when the user has opted to enable truncated output. Legal values are any integer value greater than zero. It is recommended to choose a number that is a multiple of the wrap length. The default is 270.
msg_max_hist:	The number of MOOS mail messages held per variable. Currently multiple messages are only held for string messages. Legal values are any integer value greater than zero. The default is 10.

An Example MOOS Configuration Block

An example MOOS configuration block may be obtained from the command line with the following:

```
$ pRealm --example or -e
```

Listing 2.2: Example configuration of the pRealm application.

```
===============================================================
pHostInfo Example MOOS Configuration
===============================================================

ProcessConfig = pRealm
{
  AppTick   = 4
  CommsTick = 4

  relcast_interval = 0.8       //  [0.4, 15] Default is 0.8
  summary_interval = 2.0       //  [1, 10] Default is 2
  wrap_length = 90             //  [1,inf] Default is 90
  trunc_length = 270           //  [1,inf] Default is 270
  msg_max_history = 10         //  [1,inf] Default is 10
}
```

3 Publications and Subscriptions for pRealm

The interface for `pRealm`, in terms of publications and subscriptions, is described below. This same information may also be obtained from the terminal with:

```
$ pHostInfo --interface or -i
```

3.1 Variables Published by pRealm

- `APPCAST`: Contains an appcast report identical to the terminal output. Appcasts are posted only after an appcast request is received from an appcast viewing utility.
- `REALMCAST`: Contains a realmcast report, showing the current values of a set of MOOS variables. Typically the set of variables is related to a particular MOOS app running in the same community as `pRealm`. This is a multi-line text report that has been serialized into a single string. It will be unpacked as a multi-line text report by the receiving client.
- `REALMCAST_CHANNELS`: A list channels associated with this instance of `pRealm`. Typically this list contains the names of MOOS apps running in this MOOS community, but may contain other user-configured channels.
- `WATCHCAST`: Contains a watchcast report, containing the the information on the most recent post for a particular single MOOS variable.

3.2 Variables Subscribed for by pRealm

The `pRealm` application subscribes to the following MOOS variables:

- `APPCAST_REQ`: A request to generate and post a new apppcast report, with reporting criteria, and expiration.
- `REALMCAST_REQ`: A request to generate publications to `REALMCAST` for a specified channel, for specified duration.
- `DB_RWSUMMARY`: A report generated by the MOOSDB listing, for each app connected to the MOOSDB, the set of variables subscribed for by each app, and the set of variables observed to be published by each app.

Node that `pRealm` will also subscribe for all variables discovered from the contents of `DB_RWSUMMARY` publications.

3.3 Command Line Usage of pRealm

The `pRealm` application is typically launched with pAntler, along with a group of other modules. However, it may be launched separately from the command line. The command line options may be shown by typing:

```
$ pRealm --help or -h
```

Listing 3.3: Command line usage for the pRealm tool.

```
Usage: pRealm file.moos [OPTIONS]

Options:
  --alias=<ProcessName>
      Launch pRealm with the given process
      name rather than pRealm.
  --example, -e
      Display example MOOS configuration block
  --help, -h
      Display this help message.
  --HOSTIP=<HostIP>
      Force the use of the given IP address as the reported IP
      address ignoring any other auto-discovered IP address.
  --interface, -i
      Display MOOS publications and subscriptions.
  --version,-v
      Display the release version of pRealm.

Note: If argv[2] is not of one of the above formats
      this will be interpreted as a run alias. This
      is to support pAntler launching conventions.
```

4 Structure of RealmCast Reports

The primary output of pRealm is a realmcast, a posting to the MOOS variable REALMCAST. If pRealm is running in a vehicle MOOS community, then the realmcast is typically shared to the shoreside community. A realmcast is only generated on demand, to ensure minimal bandwidth, keeping in mind scenarios with large numbers of vehicles and limited physical communications bandwidth. In this section the structure of the realmcast message is described, how this message is serialized, and how the content of the realmcast message may be customized at run time to suit user preferences.

4.1 The RealmCast Structure

The structure of realmcast is simply a list of strings. While there is a distinct formatting of the report, the end result is the list of strings with inserted white space. The assumption is that whatever app is rendering the report will be using a fixed-width character output, which will preserve the columns and spacing of the report. An example is shown below.

```
=======================================
pProxonoi                      ben (33)
=======================================
Subscriptions
=======================================
Variable        Source         Time    Comm       Variable
--------------  -------------  ------  ---------  ---------------------------------------------------
APPCAST_REQ     uMAC_5533      111.44  shoreside  node=all,app=all,duration=3.0,key=uMAC_5533:app,thresh=
NAV_X           uSimMarineX    111.69  ben        5408
NAV_Y           uSimMarineX    111.69  ben        -1725
NODE_REPORT     uFldNodeComms  39.22   shoreside  NAME=abe,X=5327,Y=-1769,SPD=0,HDG=287,TYPE=kayak,COLOR=
PROX_CLEAR      -              never   -          -
PROX_POLY_VIEW  -              never   -          -

=======================================
Publications
=======================================
Variable             Source          Time    Comm  Variable
------------------   --------------  ------  ----  ---------------------------------------------------
APPCAST              uFldNodeBroker  26.28   ben   formatted report
PPROXONOI_ITER_GAP   pProxonoi       111.43  ben   1.011477
PPROXONOI_ITER_LEN   pProxonoi       111.43  ben   0.000436
PPROXONOI_STATUS     pProxonoi       110.93  ben   AppErrorFlag=false,Uptime=112.009,cpuload=0.1298,memory
PROXONOI_PID         pProxonoi       -0.42   ben   63498
PROXONOI_POLY        pProxonoi       111.43  ben   active=false,label=vpoly_ben
PROXONOI_REGION      pProxonoi       110.43  ben   pts={-500,2300:-3500,-1000:-3500,-4600:3100,-4600:8400,
```

The above example output is typical of the default format used when pRealm is generating a report for a particular application, the pProxonoi app in this case. On the top line, the app name is listed on the left, with the vehicle name on the right. The number in the parentheses is a counter showning how many reports for this app have been generated.

The body of the report shows the set of variables subscribed for by this application, followed by the variables published by this application. For each variable, the variable name, source, time of the publication, community and variable contents are shown, similar to other scoping utilities like uXMS.

4.2 Serialization of the RealmCast Structure

As mentioned above in Section 4.1, a realmcast message is comprised primarily of a list of strings. The entire message is serialized into a single string for posting to the MOOSDB. The structure of this message is:

```
node=<nodename>!Q!proc=<procname>!Q!count=<num>!Q!line!Z!line!Z!line!Z!....
```

It is comprised of four main components, each separated by "!Q!":

- The *node* is typically the name of the vehicle, or "shoreside" if this instance of pRealm is running in the shoreside community.
- The *proc* is typically the name of the application related to this report. A realmcast may be custom configured to contain information not strictly correlated with an app. In such a case the *proc* name is the name used for this custom configuration.
- The *count* represents the number of realmcast reports generated for this particular *node* and *proc* combination.
- The *line* components represent each line in the report, separated by the "!Z!" separator.

The serialization methods are defined on the C++ class `RealmCast`. There is a function defined to convert from the RealmCast class to a string:

```
RealmCast relcast;
... structure populated

string realmcast_report = relcast.get_spec();
```

And a function convert from a string to a RealmCast class instance.

```
string realmcast_report;
... string otherwise populated

RealmCast relcast = string2Realmast(realmcast_report);
```

Serialization is handled internally by `pRealm` and deserialization is handled internally by the client application, e.g., `pMarineViewer`.

4.3 When RealmCast Reports are Generated

The default state of `pRealm` is to be not generating any realmcast messages. Messages are only generated if there is a client requesting this information. A request comes in the form of the MOOS variable `REALMCAST_REQ`. Minimally this message specifies the `pRealm` channel of interest and the duration of time into the future during which it hopes to receive responding realmcast messages. The basic idea is shown in Figure 2.

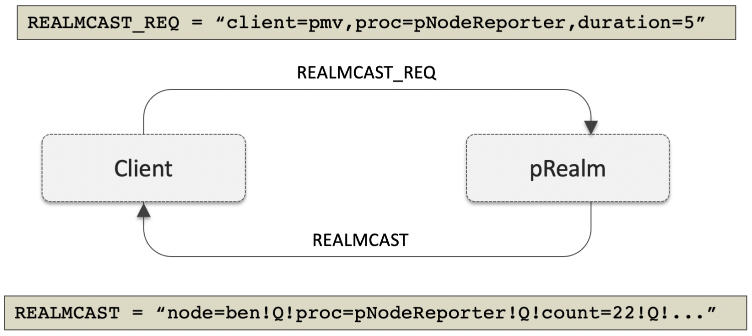

REALMCAST_REQ = "client=pmv,proc=pNodeReporter,duration=5"
REALMCAST_REQ
Client
pRealm
REALMCAST
REALMCAST = "node=ben!Q!proc=pNodeReporter!Q!count=22!Q!..."

Figure 2: **RealmCast Request:** A request is generated by a client, specifying a proc (i.e., channel or app), and a duration. For the next period in time specified by the duration, `pRealm` will produce realmcast reports at a fixed interval.

Note that if the client suddenly quits, or turns its attention elsewhere, after the short duration, `pRealm`, resumes being quiet. Note also that the rate of publications to `REALMCAST` by `pRealm` is determined by the `pRealm` configuration parameter `relcast_interval`. This interval by default is 0.8 seconds. So, in the above example, a *single* request with a duration of five seconds should result in six realmcast publications. A typical client, if interested in receiving a steady stream of realmcast reports over a period longer than five seconds, would simply send requests at a steady rate somewhat faster than the duration specified in each request. Each newly received request will reset the duration.

5 Altering the RealmCast Format to Suit the User

The default format for a typical realmcast report is shown in Section 4.1. There are seven simple ways to modify the output to further accommodate the preferences of a user. Each modification is a simple on/off state that can be toggled by the client application. As pMarineViewer is such a client, it has seven on-screen buttons for toggling the preferences. The seven methods are:

- Omission of the variable Source column
- Omission of the variable Community column
- Omission of the Subscriptions (top) block
- Masking out certain variables (described in Section 5.2)
- Wrapping the output of the variable Value column
- Truncating the output of the variabe Value column
- Showing time stamps in UTC format

Each of the above seven modifications come as part of the client's request for a realmcast report. A keyword is associated with each modification, and the absence of the keywork indicates that the modification is not requested. For example, a realmcast request with the Source and Community columns suppressed would look something like:

```
REALMCAST_REQ = client=pmv,duration=5,nosrc,nocomm,proc=pProxonoi
```

The keywords for the seven modifications are: `nosrc`, `nocom`, `nosub`, `mask`, `wrap`, `trunc` and `utc` respectively. Each incoming request sets the state for each of these mods. For example if the next request after the above one were:

```
REALMCAST_REQ = client=pmv,duration=5,nocomm,proc=pProxonoi
```

then the next outoing realmcast would no longer suppress the Source column for each variable.

5.1 Modifying the Column Output Based on Source and Community

By omitting the Source and Community columns the original output from Section 4.1 would look like that below, with both columns removed in both the Subscriptions block (top) and the Publications block (bottom):

```
=======================================
pProxonoi                      ben (33)
=======================================
Subscriptions
=======================================
Variable        Time    Value
--------------  ------  -------------------------------------------------------
APPCAST_REQ     111.44  node=all,app=all,duration=3.0,key=uMAC_5533:app,thresh=run_warning
NAV_X           111.69  5408
NAV_Y           111.69  -1725
NODE_REPORT      39.22  NAME=abe,X=5327,Y=-1769,SPD=0,HDG=287,TYPE=kayak,COLOR=dodgerblue,MODE=PARK,ALLSTO
PROX_CLEAR      never   -
PROX_POLY_VIEW  never   -

=======================================
Publications
=======================================
Variable            Time    Value
------------------  ------  -------------------------------------------------------
APPCAST             26.28   formatted report
PPROXONOI_ITER_GAP  111.43  1.011477
PPROXONOI_ITER_LEN  111.43  0.000436
PPROXONOI_STATUS    110.93  AppErrorFlag=false,Uptime=112.009,cpuload=0.1298,memory_kb=1768,memory_max_kb=
PROXONOI_PID        -0.42   63498
PROXONOI_POLY       111.43  active=false,label=vpoly_ben
PROXONOI_REGION     110.43  pts={-500,2300:-3500,-1000:-3500,-4600:3100,-4600:8400,2300},label=prox_opregi
```

The modified output simply allows more of the righthand Value columm to be more visible to the user.

5.2 Modifying the Row Output By Masking and Dropping Subscriptions

Notice the last two lines of Subscriptions section of the above output. These two variables have never been written to. They are included in the output because pRealm noticed in the DB_RWSUMMARY content that this particular app subscribes to those two variables. And at times it is helpful to see exactly the list of variables subscribed to for a given app. For some apps, however, these virgin variables can take big chunk of the report space, and it may be better to mask them out. This can be achieved by attaching the mask keyword to the realmcast request. The resulting output is shown below, with the two lines dropped.

```
=======================================
pProxonoi                      ben (33)
=======================================
Subscriptions
=======================================
Variable        Time    Value
--------------  ------  -------------------------------------------------------
APPCAST_REQ     111.44  node=all,app=all,duration=3.0,key=uMAC_5533:app,thresh=run_warning
NAV_X           111.69  5408
NAV_Y           111.69  -1725
NODE_REPORT      39.22  NAME=abe,X=5327,Y=-1769,SPD=0,HDG=287,TYPE=kayak,COLOR=dodgerblue,MODE=PARK,ALLSTO

=======================================
Publications
=======================================
Variable            Time    Value
------------------  ------  -------------------------------------------------------
APPCAST             26.28   formatted report
PPROXONOI_ITER_GAP  111.43  1.011477
PPROXONOI_ITER_LEN  111.43  0.000436
PPROXONOI_STATUS    110.93  AppErrorFlag=false,Uptime=112.009,cpuload=0.1298,memory_kb=1768,memory_max
PROXONOI_PID        -0.42   63498
PROXONOI_POLY       111.43  active=false,label=vpoly_ben
PROXONOI_REGION     110.43  pts={-500,2300:-3500,-1000:-3500,-4600:3100,-4600:8400,2300},label=prox_o
```

Furthermore, for certain apps, the number of variable subscriptions can be huge, and the user may be solely interested in monitoring one or more variable publications. The Subscriptions section can be omitted entirely by attaching the `nosubs` keyword to the realmcast request, resulting in the above output being further reduced to:

```
=======================================
pProxonoi                      ben (33)
=======================================
Subscriptions
=======================================
Variable        Time    Variable
--------------  ------  -------------------------------------------------------
APPCAST_REQ     111.44  node=all,app=all,duration=3.0,key=uMAC_5533:app,thresh=run_warning
NAV_X           111.69  5408
NAV_Y           111.69  -1725
NODE_REPORT      39.22  NAME=abe,X=5327,Y=-1769,SPD=0,HDG=287,TYPE=kayak,COLOR=dodgerblue,MODE=PARK,ALLSTO
PROX_CLEAR      never   -
PROX_POLY_VIEW  never   -
```

5.3 Modifying the Content of Very Large String Publications

Occasionally an app will publish very long string messages. In many cases, just seeing the first part of the message is sufficient for a user monitoring the system. As with the NODE_REPORT posting in the example above, the last part of the variable value is simply cut off in the output presented to the user. Rather than cutting off the line, it can be wrapped instead, by including the `wrap` tag in the realmcast request. Wrapping is shown in the example below.

```
=======================================
pProxonoi                      ben (33)
=======================================
Subscriptions
=======================================
Variable        Time    Value
--------------  ------  -------------------------------------------------------
APPCAST_REQ     111.44  node=all,app=all,duration=3.0,key=uMAC_5533:app,thresh=run_warning
NAV_X           111.69  5408
NAV_Y           111.69  -1725
NODE_REPORT      39.22  NAME=abe,X=5327,Y=-1769,SPD=0,HDG=287,TYPE=kayak,COLOR=dodgerblue,MODE=PARK,
                        ALLSTOP=ManualOverride,INDEX=81,TIME=3216098849.96,LENGTH=4
PROX_CLEAR      never   -
PROX_POLY_VIEW  never   -

=======================================
Publications
=======================================
Variable            Time    Value
------------------  ------  -------------------------------------------------------
APPCAST             26.28   formatted report
PPROXONOI_ITER_GAP  111.43  1.011477
PPROXONOI_ITER_LEN  111.43  0.000436
PPROXONOI_STATUS    110.93  AppErrorFlag=false,Uptime=112.009,cpuload=0.1298,memory_kb=1768,memory_max
                            _kb=1768,
PROXONOI_PID        -0.42   63498
PROXONOI_POLY       111.43  active=false,label=vpoly_ben
PROXONOI_REGION     110.43  pts={-500,2300:-3500,-1000:-3500,-4600:3100,-4600:8400,2300},label=prox_o
                            pregion
```

By default, the wrapping is done in 90 character increments. This can be changed with the pRealm configuration parameter, wrap_length.

Finally, in some cases wrapping may not be enough. If the string is thousands of characters long, it will dominate the realmcast output and make it hard to read anything else. The user has the option of also truncating the variable string value, by default, to 270 characters. This can be modified with the trunc_length parameter. To make the most of each line of output, it is recommended to set the trunc_length to be a multiple of the wrap_length.

5.4 Modifying the Time Format to UTC

Normally the posted time stamp is relative to the "start of the mission". There is no consensus on then the start of the mission occurs, although one could argue that it is the very instance the MOOSDB starts. However, in multi-vehicle missions, each MOOSDB starts at a slightly different time. In the case of pRealm, the start time used for calculating timestamps is simply the UTC time when pRealm started. This vagary is usually tolerated because we are simply trying to get a feel for the rough time or age of a variable post, and the nice small numbers help get a quick sense. However, at times, more precision may be preferred. The user has the option of requesting timestamps to be posted in absolute UTC time, i.e., the number of seconds since January 1st, 1970. In this way, all postings from all vehicles will be referencing the same start time, regardless of when the MOOSDB or other processes began.

A client application simply needs to add the utc component to the end of a realmcast request as such:

```
REALMCAST_REQ = client=pmv,duration=5,utc,proc=pProxonoi
```

5.5 How Does the User Actually Modify Output?

As described above, modification to the realmcast report content is achieved by adding tags to the incoming REALMCAST_REQ messages. For example, the Source and Community columns are removed when the request contains the `nosrc` and `nocom` tags:

```
REALMCAST_REQ = client=pmv,duration=5,nosrc,nocomm,proc=pProxonoi
```

And virgin variables and string content can be wrapped with the `mask` and `wrap` tags:

```
REALMCAST_REQ = client=pmv,duration=5,mask,wrap,proc=pProxonoi
```

In general the user does not need to be involved in formatting string messages with these tags. The pMarineViewer app is a client that interacts with the user and contains toggle buttons for the above content modifiers. Figure 3 shows the lefthand bottom corner of the pMarineViewer window when realmcast output is being viewed. (Also see Figure 7.) Each button is a toggle button.

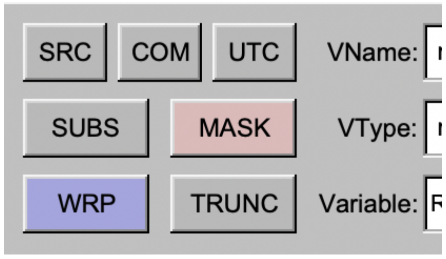


Figure 3: **Content Modifier Buttons:** The pMarineViewer app acts as a client requesting and receiving realmcast reports. The modifier buttons will toggle the seven modifiers affecting realmast formatting.

When pMarineViewer is in the mode to present realmcast content, it will generate a steady stream of REALMCAST_REQ messages. As the above seven content buttons are toggled, the next outgoing request message will be modified with the proper content tags.

6 WatchCasting: A Special Type of RealmCast

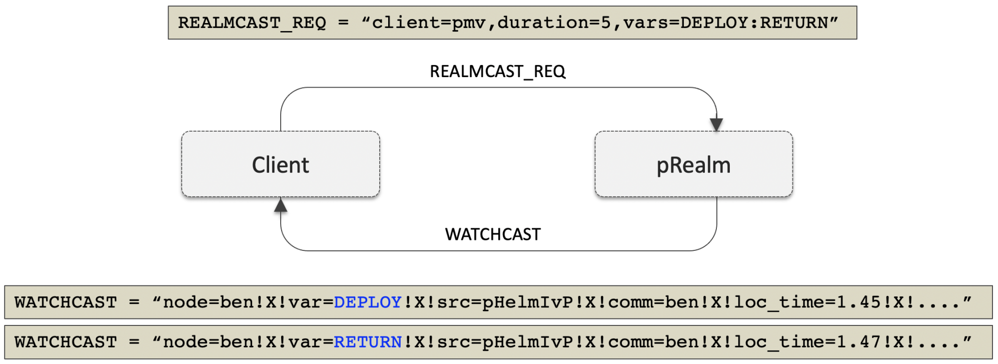
In addition to the outgoing REALMCAST message, pRealm supports a second type of message called a watchcast, posted in the variable WATCHCAST. This message contains the information about the most recent post to *one* MOOS variable. Watchcasts are generated when the most recent realmcast request names a set of MOOS variables. Here is a normal realmcast request discussed above, requesting the pub/sub information for pHelmIvP:

```
REALMCAST_REQ = client=pmv,duration=5,proc=pHelmIvP
```

And here is a realmcast request that specifies a particular set of MOOS variables:

```
REALMCAST_REQ = client=pmv,duration=5,vars=DEPLOY:RETURN:STATION_KEEP
```

When pRealm has been asked to produce this kind of content, it produces it in a watchcast message, one per variable. The idea is shown in Figure 4.



```
REALMCAST_REQ = "client=pmv,duration=5,vars=DEPLOY:RETURN"
```

```
WATCHCAST = "node=ben!X!var=DEPLOY!X!src=pHelmIvP!X!comm=ben!X!loc_time=1.45!X!...."
WATCHCAST = "node=ben!X!var=RETURN!X!src=pHelmIvP!X!comm=ben!X!loc_time=1.47!X!...."
```

Figure 4: **WatchCast Request:** A request is generated by a client, specifying a set of variables, and a duration. For the next period in time specified by the duration, pRealm will produce a watchcast report for each variable, each time the variable changes, but no more frequent than a fixed interval.

Watchcasting allows the client to configure a cluster of variables that may be important to the user. These variables may be involved in several different MOOS applications and thus hard to visualize and monitor. Clients of pRealm like pMarineViewer and uMACView are able to specify watch clusters and view a single report similar to:

```
Node    DEPLOY     RETURN    STATION_KEEP
----    ------     ------    ------------
abe     true       false     false
ben     true       false     false
cal     true       false     false
deb     true       false     false
eve     true       false     false
fin     true       false     false
gus     true       false     false
hal     true       false     false
```

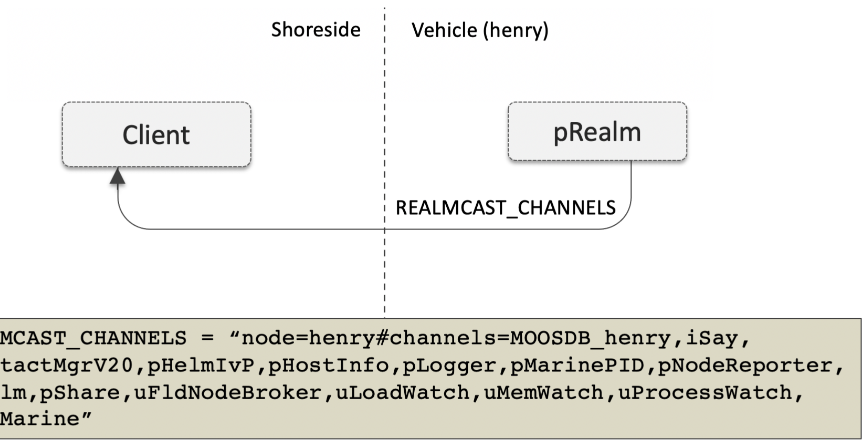
These may be the key variables in the autonomy of a particular mission, or they could be health montitoring variables for deployed platforms:

```
Node    BATT_VOLTAGE      CPU_TEMP    GPS_SATELLITES
----    ------------      --------    --------------
abe     17.2              111.2       11
ben     16.9              112.0       8
cal     17.4              111.9       9
deb     17.4              108.7       10
eve     17.0              101.2       11
fin     15.8              105.5       11
gus     16.4              114.2       9
hal     16.1              105.6       10
```

Since multiple pRealm clients may be run simultaneously, e.g., pMarineViewer and one or more instances of uMACView, then the user could have several such tables viewable. In multi-robot vehicle deployments, this may be extremely useful. Unlike appcast content, the above content is customizable by the user without any adjustments to code. Watch clusters are configured in pMarineViewer or uMACView configuration files. See the documentation for these app for more information, [1], [2].

7 How pRealm Informs Potential Clients

We have established how clients of pRealm may request reports on a given channel, Figure 2. But how does a client like pMarineViewer know there exists a pRealm to query, and which channels may be queried? To accomplish this, pRealm publishes its key information periodically for clients to discover. At a rate of once per five seconds, pRealm publishes to the MOOS variable REALMCAST_CHANNELS a report that identifies the name of the node/vehicle, and all available channels for querying. This idea is conveyed below.



```
REALMCAST_CHANNELS = "node=henry#channels=MOOSDB_henry,iSay,
pContactMgrV20,pHelmIvP,pHostInfo,pLogger,pMarinePID,pNodeReporter,
pRealm,pShare,uFldNodeBroker,uLoadWatch,uMemWatch,uProcessWatch,
uSimMarine"
```

Figure 5: **RealmCast Channels:** The pRealm app periodically posts information about itself. This information is used later by clients to request realmcast reports from specific nodes and specific channels.

By default, these messages are posted once every 5 seconds. This value can be changed with the summary_interval parameter. There are two exceptions to this interval. For the first minute after launch, the summary will be posted every half second. The second exception is applied when a new app has been detected. The set of apps participating in the local MOOS community normally doesn't change after the initial startup. But if a new app is later detected by pRealm, the REALMCAST_CHANNELS variable will be published immediately with this new information.

A final note: the time interval between summaries is real time, not time warp time.

8 Terminal and AppCast Output

The pRealm application produces some useful information to the terminal on every iteration of the application. An example is shown in Listing 4 below. This application is also appcast enabled, meaning its reports are published to the MOOSDB and viewable from any uMAC application or pMarineViewer. The counter on the end of line 2 is incremented on each iteration of pRealm, and serves a bit as a heartbeat indicator. The "0/0" also on line 2 indicates there are no configuration or run warnings detected.

Listing 8.4: Example terminal or appcast output for pRealm*.*

```
 1  ==================================================================
 2  pRealm gilda                                             0/0(3074)
 3  ==================================================================
 4  Configuration:
 5  --------------------------------------------
 6    RelCast Interval: 0.55
 7    Summary Interval: 7.60
 8    Wrap Length:  90
 9    Trunc Length: 270
10    Max Msg Hist: 10
11
12  MOOS Community State:
13  --------------------------------------------
14    MOOSDB Name: MOOSDB_gilda
15     Known Apps: 15
16     Known Apps: MOOSDB_gilda,iSay,pContactMgrV20,pHelmIvP,
17                 pHostInfo,pLogger,pMarinePID,pNodeReporter,
18                 pRealm,pShare,uFldNodeBroker,uLoadWatch,
19                 uMemWatch,uProcessWatch,uSimMarine
20    Total SVars: 196
21    Total PVars: 137
22    Unique Vars: 206
23    UTC Time:    12874301098.43
24    Local Time:  1605.84
25    Summaries:   40
26
27  Recent RealmCasts or WatchCasts:
28  --------------------------------------------
29    Total RealmCasts: 238
30    Total WatchCasts: 6
31  --------------------------------------------
32  Count     Time  Client  Content
33  -----  -------  ------  -------------------------------------------
34  (165)  1604.28  pmv     pHelmIvP
35  (47)    829.04  pmv     iSay
36  (1)     424.30  pmv     vars=DEPLOY
37  (1)     137.69  pmv     vars=AVOID,DEPLOY,LOITER,RETURN,STATION_KEE
```

```
38  (26)    132.98  pmv     iSay
39
40  Currently Active Clients:
41  --------------------------------------------
42  Client  Active  Content
43  ------  ------  --------
44  pmv     21.37   pHelmIvP
45
46  ===================================================================
47  Most Recent Events (8):
48  ===================================================================
49  [831.62]: Client pmv, new pipe: pHelmIvP
50  [533.97]: Client pmv, new pipe: iSay
51  [386.22]: Client pmv, new pipe: vars=DEPLOY
52  [336.66]: Client pmv, new pipe: vars=AVOID
```

Lines 6-10 of the output show the current values of the five pRealm configuration parameters listed in Section 2.

Lines 12-22 relate state of the MOOSDB, including the name of the MOOSDB which is always the community name at the end of the MOOSDB_ prefix. The number of and list of know apps connected to the MOOSDB are shown in lines 15 and 16. Line 20 shows the number of MOOS variables involved in a subscription by at least one app. Line 21 shows the number of unique MOOS variable publications. Line 22 shows the number of unique variables known to the MOOSDB involved in either a subscription or publication. Line 33 shows the current UTC time, with the time warp multiplier applied. Line 24 shows the local time, i.e., the time since pRealm was started. Line 25 shows the total number of realmcast summaries posted, to the variable REALMCAST_CHANNELS.

Lines 27-38 indicate the most recent realmcasts or watchcasts. First the total of each is shown on lines 29 and 30. Lines 34-38 show the five most recent content settings. The first column, Count, shows the number of successive outgoing messages. The Time column shows the local time of the most recent message in that group. The Client column shows which app is requesting the content. This will show as "pmv" when pMarineViewer is the client. Finally, the Client column indicates the content of the outgoing message. When the content begins with "vars=", this indicates that it is a watchcast.

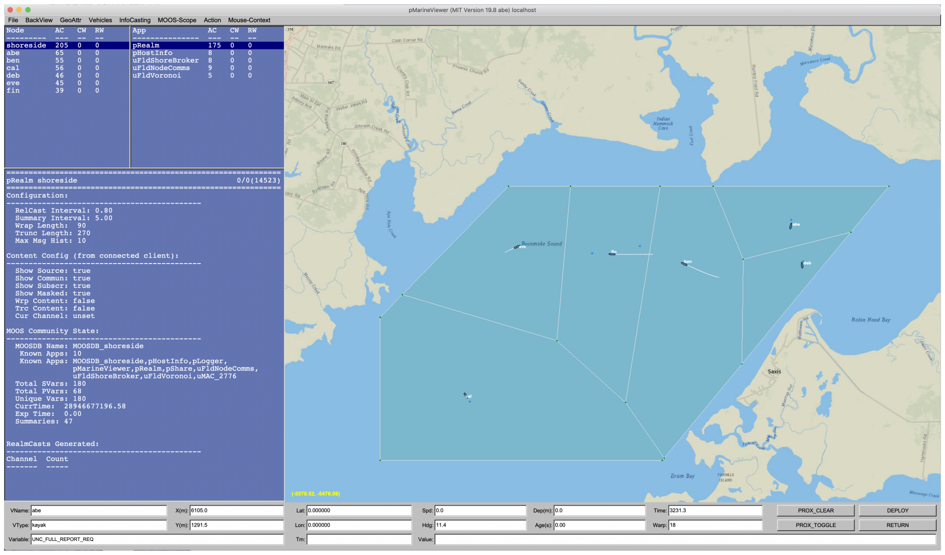
Lines 40-44 show the status of active clients. A client is active if pRealm has recently received a realmcast request. As discussed earlier, each request has a duration. The duration implies a time remaining before the request expires. This time is shown in column two. Column three shows the content being requested in the realmcast request.

Lines 46-56 contain the typical recent events block for all appcasting apps. In the case of pRealm, an event is posted each time a realmcast request has been received that contains a change in requested content.

9 A Preview of Using pRealm Information within pMarineViewer

In most cases the user won't give another thought about pRealm beyond simply including it in the pAntler launch lists. The user experience is through pMarineViewer. Initially this is the only app that is able to handle realmcasts and present them to a user. So here is a quick preview of the pMarineViewer experience.

Normally when the a mission is launched with the pMarineViewer the window shows the three "infocast" panes shown on the left. The infocast panes are either showing appcast content, or realmcast content. The infocast panes show appcast content by default. Unless the user changes the default color schemes, appcasting is distinguished by an indigo background as in Figure 6.



```
Node       AC   CW  RW     App              AC   CW  RW
---------  ---  --  --     ---------------  ---  --  --
shoreside  205  0   0      pRealm           175  0   0
abe        65   0   0      pHostInfo        8    0   0
ben        55   0   0      uFldShoreBroker  8    0   0
cal        56   0   0      uFldNodeComms    9    0   0
deb        46   0   0      uFldVoronoi      5    0   0
eve        45   0   0
fin        39   0   0
```

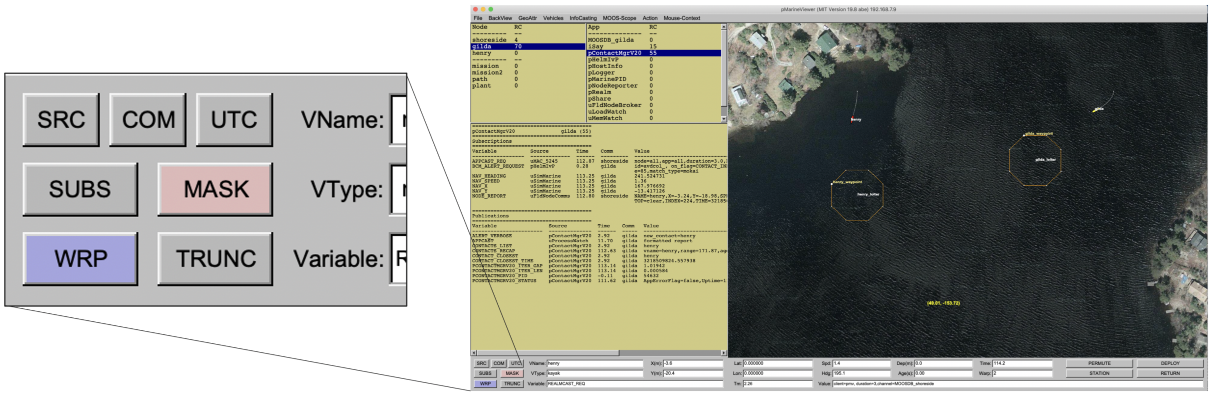
```
==============================================================
pRealm shoreside                                   0/0(14523)
==============================================================
Configuration:
--------------------------------------------
  RelCast Interval: 0.80
  Summary Interval: 5.00
  Wrap Length:  90
  Trunc Length: 270
  Max Msg Hist: 10

Content Config (from connected client):
--------------------------------------------
  Show Source: true
  Show Commun: true
  Show Subscr: true
  Show Masked: true
  Wrp Content: false
  Trc Content: false
  Cur Channel: unset

MOOS Community State:
--------------------------------------------
  MOOSDB Name: MOOSDB_shoreside
   Known Apps: 10
   Known Apps: MOOSDB_shoreside,pHostInfo,pLogger,
               pMarineViewer,pRealm,pShare,uFldNodeComms,
               uFldShoreBroker,uFldVoronoi,uMAC_2776
  Total SVars: 180
  Total PVars: 68
  Unique Vars: 180
  CurrTime:  28946677196.58
  Exp Time:  0.00
  Summaries: 47


RealmCasts Generated:
--------------------------------------------
Channel  Count
-------  -----
```

VName: abe | X(m): 6105.0 | Lat: 0.000000 | Spd: 0.0 | Dep(m): 0.0 | Time: 3231.3
VType: kayak | Y(m): 1291.5 | Lon: 0.000000 | Hdg: 11.4 | Age(s): 0.00 | Warp: 18
Variable: UNC_FULL_REPORT_REQ | Tm: | Value:

Figure 6: **The pMarineViewer AppCasting Mode:** When pMarineViewer starts, normally it is in appcasting mode, with the left three appcasting panes rendered in indigo showing appcast content in the lower pane.

The appcasting panes let the user select the node in the upper left pane, e.g., the shoreside or one of several vehicles. The upper right pane lets the user select the MOOS app running on the selected node. The bottom pane shows the appcast content for the node and app selected in the top two panes.

The realmcast mode is very similar. To toggle between modes, the 'a' key is used. Unless the user changes the default color schemes, the realmcasting is distinguished by a beige background as in Figure 7. As in the appcasting mode, the top two panes offer virtually the same options based on nodes and processes. The bottom middle pane switches however to show realmcast content.



```
Node       RC        App             RC
---------  --        --------------  --
shoreside  4         MOOSDB_gilda    0
gilda      70        iSay            15
henry      0         pContactMgrV20  55
---------  --        pHelmIvP        0
mission    0         pHostInfo       0
mission2   0         pLogger         0
path       0         pMarinePID      0
plant      0         pNodeReporter   0
                     pRealm          0
                     pShare          0
                     uFldNodeBroker  0
                     uLoadWatch      0
                     uMemWatch       0
```

```
=======================================
pContactMgrV20                gilda (55)
=======================================
Subscriptions
=======================================
Variable           Source         Time    Comm       Value
-----------------  -------------  ------  ---------  ------------------------------
APPCAST_REQ        uMAC_5245      112.87  shoreside  node=all,app=all,duration=3.0,
BCM_ALERT_REQUEST  pHelmIvP       0.28    gilda      id=avdcol_, on_flag=CONTACT_IN
                                                     e=85,match_type=mokai
NAV_HEADING        uSimMarine     113.25  gilda      241.524731
NAV_SPEED          uSimMarine     113.25  gilda      1.36
NAV_X              uSimMarine     113.25  gilda      167.976692
NAV_Y              uSimMarine     113.25  gilda      -13.417126
NODE_REPORT        uFldNodeComms  112.80  shoreside  NAME=henry,X=-3.24,Y=-18.98,SP
                                                     TOP=clear,INDEX=224,TIME=32185

=======================================
Publications
=======================================
Variable                 Source          Time    Comm   Value
-----------------------  --------------  ------  -----  ------------------------------
ALERT_VERBOSE            pContactMgrV20  2.92    gilda  new_contact=henry
APPCAST                  uProcessWatch   11.70   gilda  formatted report
CONTACTS_LIST            pContactMgrV20  2.92    gilda  henry
CONTACTS_RECAP           pContactMgrV20  112.63  gilda  vname=henry,range=171.87,ag
CONTACT_CLOSEST          pContactMgrV20  2.92    gilda  henry
CONTACT_CLOSEST_TIME     pContactMgrV20  2.92    gilda  3218509824.557938
PCONTACTMGRV20_ITER_GAP  pContactMgrV20  113.14  gilda  1.01942
PCONTACTMGRV20_ITER_LEN  pContactMgrV20  113.14  gilda  0.000584
PCONTACTMGRV20_PID       pContactMgrV20  -0.11   gilda  54632
PCONTACTMGRV20_STATUS    pContactMgrV20  111.62  gilda  AppErrorFlag=false,Uptime=1
```

VName: henry
VType: kayak
Variable: REALMCAST_REQ
X(m): -3.6
Y(m): -20.4
Tm: 2.26

Figure 7: **The pMarineViewer RealmCasting Mode:** The pMarineViewer user may toggle into realmcasting mode using the `'a'` key, with the left three panes rendered in beige.

When in realmcast mode, pMarineViewer will display the seven buttons shown above. These buttons allow the user to modify the content modes discussed in Section 5.

10 Additional Notes

Some additions planned for the future:

- General masking: Currently masking is only done on virgin variables. The plan is to also allow the user to mask out entire apps, or a list of variables for an app, or list of variables across all apps.
- Configurable channels: Currently the only channels are connected apps, plus an artficial channel for the MOOSDB. Like uXMS, the plan is to allow the user to define their own channel with a custom list of variables to watch.
- Global channel: Currently there is no channel that simply lists all variables.
- Messaging channel: A channel specifically for monitoring inter-vehicle messaging.

References

[1] Michael R. Benjamin. pMarineViewer: A Tool for Mission Monitoring and Control. http://oceanai.mit.edu/ivpman/apps/pMarineViewer.

[2] Michael R. Benjamin. The uMAC Utilities. http://oceanai.mit.edu/ivpman/apps/uMAC.