Change proposal to improve efficiency of ITS traffic

Background

Priority one messages are currently transmitted over the Iridium Short Burst Message system. Messages are associated with a given service name. Using a dataset of twenty days of ITS data the number of messages sent per service name look like this:

```
Start: 2012-11-30 23:58:08.118194      End: 2012-12-19 00:32:30.249736

DB 7387
GammaFollowUp 7456
I3DAQDispatch 7384
I3MoniDomMon 5190
I3MoniDomSn 5189
I3MoniDomTcal 5189
I3MoniMover 5190
I3MoniPhysA 5189
None 495
OpticalFollowUp 10066
PFClient.* 1689
PFFiltDispatch 7388
PFFiltWriter 7610
PFRawWriter 7389
PFServer1 7382
PFServer2 7383
PFServer3 7380
PFServer4 7381
TFRateMonitor 8650
livecontrol 3637
meteorology 2594
pdaq 44882
sndaq 5174
temperature 7821
uptimer 5190

Total messages: 190285
Service State Count:  118847 (62%) (varname == ‘state’)
‘Running’ updates:  117821 (value == ‘RUNNING’)
Percent run state: 0.619182
```

Conclusion: over 61.9% of ITS priority one messages are used to report that

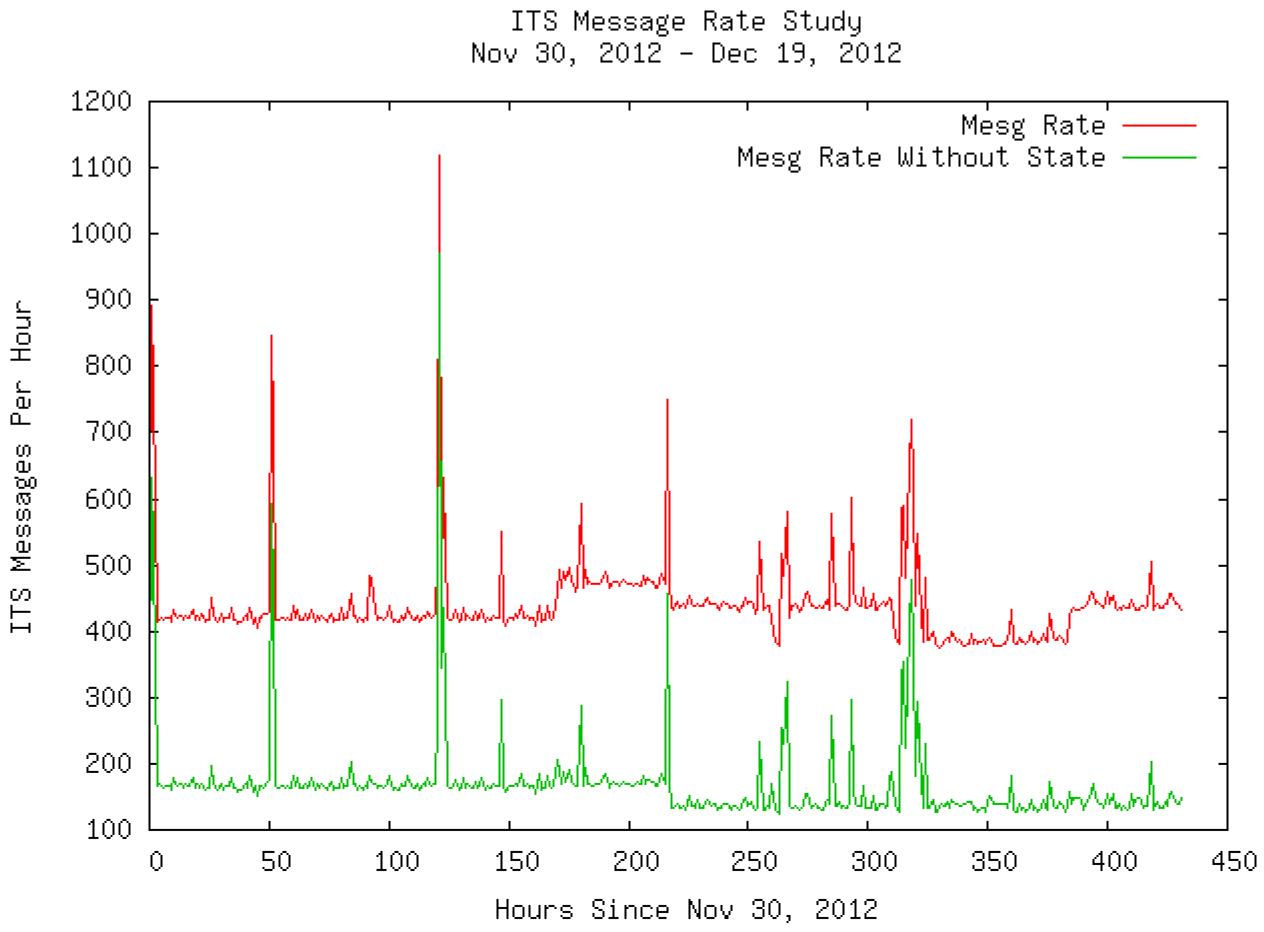
components are in a ‘running’ state.

Add up the raw string representation for the above 61.9% messages, 37060248 bytes of a total 63813214 bytes (58% of bytes).

Currently components are responsible for reporting their state at some interval. This is not super clean as there is no guarantee the component actually will do that, or that every component will use the same interval.

Proposal

1. **State Collection** - LiveControl will continue to collect state information as sent by components.
    a. LC will send a list of controlled components (LCC)
        i. once per hour
        ii. whenever a new component is ‘livecmd control’led
        iii. this message will be versioned (‘component version number’ or CVN)
    b. LC will send a dictionary of all components *not in the running state* (CNORS)
        i. every 30 minutes
        ii. whenever a component changes state
        iii. this message will also report the latest CVN
2. **State Receiving** - DBServer will
    a. upon receiving LCC, update its component list (models.Component), including the new CVN (need migration for this)
    b. upon receiving CNORS, update the component list *if and only if the CVN matches the latest*.



Background: Summary of Discussion on Skype

Matt - For a time slice of about 20 days (end of november through dec 20 last year) 62% of the total priority one messages where.. "service": <name of live component>, value: "RUNNING".. was wondering why if we have heartbeats why we couldn't adjust the algorithm so if there has been a heartbeat and no error message then 'running' would be assumed.

John - which kind of heartbeats do you mean?

Matt - Oh I mean like a 'I'm alive' message.. There are the ones that show up in the warehouse, but I suppose we could send a simple one over its.. (sends plot to John)
(plot of the resulting message rates.. each point is a count of messages in a 1 hour bin.
The line with the higher total rates is the real traffic, the lower one is if you filter out 'running' messages
the peaks are either pfclient crashes or pdaq issues and a storm of livecontrol messages.

John - I don't understand your proposal. How are the components going to report their state, exactly?

Matt - No worries.. If a component has a 'running' state it would report nothing. If it had a state other than running it would report it (ie error, recovering etc). On a regular basis live would send a single heartbeat message north over its. On the north side for some time window if there was no error message, and a recent heartbeat then 'running' would be assumed. If no recent heartbeat or error message then put a '*' next to the component state as we do now. If there is a recent heartbeat and an error message report it..

John - How would you distinguish a component which is quietly running from a component which is completely crashed/missing?

I.e., say live is running fine, and talking to the north, but sndaq is completely AWOL. how will that information be detected and communicated?

Matt - Ahh, an architectural misunderstanding... So right now a component is the software that reports it's state?

if the south collects status information.. the same piece of code that generates the heartbeat would fill in the missing error message..

John - I guess (without having really planned it this way) that a "component" is something you can "control" (acquire, start, stop, recover). As opposed to a "service" which is a distinct entity which can send one or more moni vars.
Most "components" report their state occasionally; also live reports their state if, say, an operator changes it, or tries to and it raises an exception.
As of now it's up to the components to report their state unless operators take action. But I don't really like this design. Probably livecontrol should just poll all the components it knows about.
Then there's the issue of how to communicate that to the North. One change we could do is to report a dictionary of state changes since the last report.

for example,

{sndaq: RUNNING, uptimer: STOPPED}
{}
{}
{sndaq: ERROR}

means what you really had was:

Matt - Sure, that could work, except if for some reason you have all the components changing state at one wack..

John - sndaq running, uptimer stopped
sndaq running, uptimer stopped
sndaq running, uptimer stopped
sndaq error, uptimer stopped

if all components changed at one whack you'd still be OK w/ this scheme

Matt - yeah, I get your example. You might throw in a full state update every once in awhile..

John - sure, that'd be fine too, and easy to do

but the first order of business would be to poll the components from livecontrol

then maybe get the components themselves to shut up about their state

then implement the deltas

you could drastically reduce the bandwidth this way

Matt - (yes, except "running" + "name of component" inside a json string gets really close to 1800 bytes. okay so the compression would take care of a lot, but scalability becomes an issue.

if in the future you add a number of new components...

John - you could also solve that one simply by setting up an indexed list of components and states and then just report the component IDs and state IDs. Ugly but compact.

Matt - Sure, makes sense.. I do think aggregating the messages makes lots of sense. The overhead on a single json string per state update message is pretty high..

John - yes, we're not using the resource efficiently at all, i'll grant you that.

Matt - besides.. the priority queue scheme in case of too much traffic throws away data from the noisiest service first.. Right now these state messages are over half the traffic but they are split across many many different service names..
John - well, and since the updates in the new scheme would come from livecontrol (which presumably isn't going to be the noisiest, else we have bigger problems)...

Matt - pdaq is by far and away the busiest on average over the course of the month. but not too far behind pdaq.. especially depending on the update rate for these status reports.. Might make sense to put them in their own service.. Livecontrol can have some interesting messages like

"run fail" etc.
John - sure. so, should we organize these ideas in one place, maybe a gDoc, and then break out tickets for them?

Matt - sounds reasonable..

John - you up for starting it?

Matt - I've got to leave before too long but I can write something up for a little while..

John - we could even do a little discussion at the collab meeting during the ops session