Puck/SIAM Review Comments from John Graybeal
11/18/03

Instrument Puck

1. What if there's a node that we want to change instruments on, but which doesn't have a way to command it to do port scans? (e.g., bottom of the ocean)
2. What does non-isolated mean? (The term also shows up in Hardware presentation—does it always mean electrical isolation of the RS-232 circuit?)
3. Is the device info on the Puck space limited in any way? (The specification of byte sizes doesn't mean there's a concern with space, does it?)
4. I notice the example (Metsys.jar) doesn't include an XML file—what's up with that? I concur with the gist of the discussion that instrument description information (i.e., the XML) is useful.
5. I'd recommend changing the name of the ID variable. Calling it an "ISI ID" emphasizes the 'MBARI provincial' perspective, at the cost of appealing to folks as a broader solution.
6. What is the puck 'breakout' command sequence, to get from pass-through (sensor) mode back to command mode?
7. How can you make command set less likely to interfere with instrument command set? This is important for the case of an embedded puck.

Instrument Puck Hardware

1. Is there a "go to sensor mode" other than the PUCK command?
2. What's the term "Executive" mean in the PUCK context?
3. What is the difference between embedded and external pucks from the standpoint of going into/out of sensor mode? What are the implications?
4. What's the interface that is used to burn the puck?
5. Is there a command to burn the puck (through its RS232)? Is there any reason there couldn't be? I think you may have answered this in the discussions, but I wasn't sure.
6. Why can you write only 32 bytes at a time when writing to memory?

Discussion

1. Process for making changes to the PUCK (its usage life cycle) not well covered.
2. Consider making the spec more general as far as PUCK vs sensor command interference – can both be available simultaneously, at least in concept?
3. Policies are the bigger issue/challenge to deal with than technology.
4. A reference site will be critical to a widely accepted implementation.
5. 1451.4 schemes bear further investigation. I concur that agreeing on the critical data to put in the PUCK header is an important step. (And that XML is one component of the overall content that is critical, but I think the core header should be binary, as you have it already.)
6. Love the idea of "first command must be X" in order to minimize puck interference.

SIAM Development Process

1. SIAM is getting pretty close to benefiting from the IDEs. People could try Eclipse for starters, since it's free, has pretty flexible editor support (much better than Together in that regard), and CVS support too.
2. The frequency of the words 'manager' and 'service' in the OO diagrams may be worth looking at. My reading and experience (FWIW) suggests these types of objects lead to a more procedural SW organization, rather than OO.
3. Might need to look more about overhead costs of Java, particularly the latencies involved (run-time environment, RMI overhead, …?). I remain fearful about how this will scale. At a minimum, you'll need to explain this pretty clearly to the developers implementing GUIs (which Java features they can use or should evolve).
4. The Node appears to incorporate both "mechanical" (power, serial ports, etc.) and "logical" (data/software) interfaces. It may prove useful at some point to split those up (maybe make them Interfaces?), since the mechanical interfaces may not extend to all types of nodes. (Or, to put it another way, the logical interfaces will be useful in other places, like some future AUV, where the mechanical wouldn't apply. (This concern may qualify as a nit.)
5. Consider publish/subscribe (even home-spun). Although now that I think about it, threading could become a significant resource drain as the moorings become more complicated/data-rich.

Sample Acquisition and Logging

1. What is the purpose of the on-board data store and how is that purpose met? A: It's a wet-side cache, intended for physical recovery. Mechanism for data ingest from this component is still TBD, so that's a significant pending activity.

Shore Side Portal

1. The leaseListener is much more a lease controller (or just a Lease) than a listener. From the name, I thought the listeners were subscribers who wanted to know the status of the lease, but instead it is actually the thing that takes action on all the lease requests, right? Consider renaming it.

Metadata

1. Important concept that Dan mentioned is the ability to get metadata in machine-readable form from the web sites/interfaces that publish the data. (I hadn't thought about that before!) I think we can do this.
2. Another key point about the dictionaries is the need for a reference authority that everyone uses for their standards.

Shore Side Data System

1. How big/permanent are the "Logs" that are in the Portal box? Are they caches? For how big/how long?
2. Device packet header: Is sourceID the same as ISI_ID? If so, please use the same term in all places.

3. Can we make each metadata packet contain one and only one of the items listed on the "Metadata packet contents" slide? Life gets more complicated when one packet contains multiple objects.
4. I am concerned that the SeqNo representing the "current XML description" is incrementing when the underlying XML hasn't changed. I see better why you want to do this, but it still creates a headache on the part of anyone (shore side, deployed, or operations) who is trying to manage data streams and understand when they change. (Idea for a compromise solution: Is it possible that the packet header metadata ID can stay fixed until the original XML actually changes? Sorry if that's a confusing question, I might need a white board to explain it.)
5. Agree that only collecting part of the metadata during a conport session isn't as robust as just recording all of it. Storage on the shore side is cheap (compared to bandwidth needed to go execute more queries).
6. Should system-wide metadata (e.g., comments on a platform) be associated to components on that platform? I'd suggest this association be made through the parent relationship (e.g., user says "show me all comments/changes to parent") rather than by having a mechanism to promulgate a single message to all the children. (If the *action* to a parent clearly *impacts* the children, promulgating the action might make sense.)

Instrument Service Framework

1. The notion of backtracking through a stream to find the "parent packets" (mentioned by several questioners, and not contradicted by the presenters) is broken. It implies everyone who wants to make use of a data packet from an instrument has to save the entire history of the stream (or cache the potential parent packets). This isn't feasible, let alone wise. The service providers should have the responsibility of providing the "parent packets" on request, which can be done with some simple caching. (Although because all the metadata sequence numbers will have to be cached, this could turn into a fair amount of storage.)

Device Synchronization

1. (Jeff) Because the things being measured are opaque in the infrastructure right now, you have limited ability to automate the systems. (John G) It is early days to be worried about automating (in the infrastructure!) the ability to synchronize systems—there are lots simpler ways to get this done.

Transition Plan

1. Concur that end-to-end know-it-alls are needed.

Overall

1. (Dale) Reference implementation is needed (particularly of Puck) that people could use to jump-start their own efforts.
2. (Jeff) System level concern with use of RMI over low-bandwidth links, reconsider that. Consider a shore-side proxy with the data from ALL the wet-side systems. Don't require repeated transactions via the proxy link to pull up what you want. (Brent) Consider including 'age limits' in the requests.

Connor's Talk

1. Some coordination between our in-house schema work, the Marine Metadata Initiative terminology effort, and the companies would be a good thing. Would it be helpful to send a link to the MMI work to your reviewers, and invite their participation? I am also talking to people individually as it comes up, but maybe you'd like to do a more systematic approach?

Power Management

1. The interactions of the power management elements might be better presented as a sequence or activity diagram. Hearing about each component in isolation forces us to anticipate and remember what all the other components do.
2. Need to take latencies into account when waking the system to sample events.
3. (Jeff) Look at state behaviors (race conditions, deadlocks) for power/lease management. (John G) Centralizing system-level functions deserves more consideration in general.

Events

1. Why wouldn't distributed events' delivery be guaranteed, even if the network fails? When the network comes up the subscribed data should be delivered, shouldn't it?
2. Concur with Jeff's assessment that a generic mechanism for dealing with measurements (and making events another form of measurement) is valuable. But not sure why he distinguishes measurements from data – I think they're both the same thing. (Later discussion confirmed this.) Every data token (measurement in his terms) should be well described.
3. Having trusted timestamps is a very valuable mechanism, useful for many purposes, including ordering. If we make that system reliable, it could make a lot of other systems work better (and we could potentially delete the sequence number).

Multi-Node Moored Networks

1. Will there be radio bandwidth or collision problems with multiple moorings?
2. In a multi-mooring environment, will there be contention between portals (if 1 portal per mooring) or within a portal (if 1 portal goes to multiple moorings)?
3. I agree you would like to be able to direct requests to specific nodes without waking up all the other components. The cost of waking up all the nodes intuitively seems expensive. For what my intuition is worth in this case, which isn't much.

Security

1. Security is more critical than was given credit by the level of attention we've paid attention to it. I thought Kent did a good job presenting the status, but that some serious discussions will be needed to get anywhere near a resolution of these issues.

2. In general, taking 15 minutes to give generic information to a high-level group of reviewers disrespects the time those participants are giving you.

Fault Tolerance

1. Portal logging policies will be useful to take a cut at. (Unless you want to encourage people to go to the portal for their data history, which will eventually start loading the portal more than I'd guess you'd want).

Bug Tracking/Revision Control

1. Eventually you will want a more sophisticated bug tracker than phpbt, IMHO.
2. Are you keeping track of component revision numbers when you enter bugs? That's one place phpbt starts becoming a struggle.
3. Nice work tagging all the user meetings in the software.

Discussion

1. Agree that development process for drivers has to be generalized so that more people can do it. Lots of hand-holding needs to be a part of this process.
2. Should definitely log all transactions to/from deployed systems.