Constellation {binhle, jennyxue, joshh}

In the following is a list of general comments, which I created after reading all your reports. I fully understand that you may not had sufficient space or time to address all of these concerns for the preliminary report. Rather the comments are meant as a guideline for the final report and potentially your presentation. Also not all comments might apply to your report. Therefore, the last section of this document contains a more specific list of issues I noticed when reading your report.

Presentation issues:

A good technical description starts with an overview.

- What are the key features of your design
- What interesting design decisions did you make
- What do you think makes your design different than others
- What assumptions do you have
- How do you solve the most important constraints/requirements

It is less helpful to reiterate the different device types, etc. (I know that the example preliminary report did this, but still…)

I also suggest for your final report you include the following 3 things:

1) Abstract/executive summary (1-2 paragraphs): by just reading the executive summary one should be able to understand your most important design decisions and what your design project does differently than others
2) Project summary (roughly a page): The project summary should provide a high level overview of your design and how things work together. For example, in the summary you describe the general idea of your routing but not the details. You talk about the key assumptions, constraints and how you solved them.
3) Detailed description: Here you go into the details: your message design, how many bits you use for what, how you plan to implement the concurrent access of data on the FCS, etc.

Use visualization and tables to convey the key points. Here are 3 suggestions (none of them are a must and it highly depends on your DP if they are useful or not)

- An overview figure(s) compactly visualizing the main design decisions,
- A visual example how a routing table looks like, and how the routing is done from the sensor to the FCS and back
- The design of the FCS (could be a state-diagram, flow-short, a figure visualizing threads/queues/forks, etc.)

They might be other useful visualizations. The same visualizations are probably also very useful as a hand-out for our presentations.

At the same time, do not add visualization which add little content. Good visualizations show many facts, explain complex connections, etc at once while being simple. For example, many of you used a perfect building with equally shaped offices to show how sensors/repeaters are placed. While useful, I would have loved to see some more real examples (e.g., how would your design look like for Stata's 1st floor?).

A general suggestion is also to create a table of requirements (e.g., one frame per second of video data, not more than 16 active connections per repeater, etc.) and how those requirements are achieved in your design. It is all about to make your key design decisions more visible as well as to ensure that you covered all the cases. On a side note: a good example for this is the "Dynamo: Amazon's Highly Available Key-value Store" paper. Just take a look at their figures and tables, and how they are used to explain different concepts.

Think about your audience (mainly me :-): I am reading a lot of similar proposals and I want to learn about the most important differentiators of your proposal first, before I try to understand all the details. In addition there are a lot of proposals. Make yours stand-out. Nice visualizations, funny examples, have an animation for your routing protocol in a link,.... Nothing of this is a must - I am just giving you some ideas and try to point out, what you have to do if this would be a real system proposal for a contract competition or grant.

Terminology: Many of you start to introduce additional terminology (e.g., node). You should precisely define what that means (e.g., only smart devices and repeaters/gateways, but not the FCS?)

Design Justifications:

Many design decisions appear rather random and much more details and justification is needed. For example, you probably want to include some back of the envelope calculations to show how many nodes each gateway / repeater will handle, if the bandwidth is sufficient to handle all requirements, etc. Avoid numbers without justification, or missing numbers to understand topology, or ensure that topology has sufficient bandwidth to deliver all data.

Discuss limitations and alternative designs. For example, what would be viable alternative to your overall design, routing protocol, main thread and why did you not use it. Sometimes the answer can be there are equally good.

You should also provide some analysis of costs and the tradeoff between $$ and reliability or ability to collect all data. What is the projected cost of your design and what would alternative cost. If you want to show, why your design is overall more cost-effective you can make assumptions about personal cost.

You should study and describe for all design decisions the worst case scenario. For example, what is the worst-case latency vs. average latency without failures. What happens during failures. When is it possible that the number of active connections are not sufficient and devices can not connect. How likely are those scenarios and is it OK to tolerate them (if they are not, maybe you should change your design). What is the worst case for your physical layout strategy (e.g., isolated small buildings, very long floors, etc). Does MIT have this scenario? To help with that, try to play devil's advocate within your team-mates. When I am reading a paper, most of the time I am thinking of all the potential problems the design might have and under which scenario it falls apart. This is very good practice in case in the future you end up writing research papers, project proposal (internal/external), grants,....

Physical layout:

Your system should be able to work regardless of the layout of various buildings. As an example, a design that assumes that every room is a square, that repeaters can be placed in a way to NEVER receive beacons from the floor below/above, etc. are all resting on a very unreasonable assumptions (just go into building 32 :)

You should think about how your system performs in lots of different types of buildings: those with long hallways, with large classrooms such as 26-100, with virtually no right angles as in building 32, etc. You should also think about how your system handles the fact that MIT's main campus area is not a rectangle.

As mentioned above it is probably also a good idea to test your physical layout on some real floor plans as case studies and for self evaluation.

Networking issues:

Full link-state routing probably isn't a reasonable choice, at least not running on the repeaters or sensor nodes. In most designs, each node has a short, fixed path to the root.

Think about whether links state or path vector protocols are really necessary, given that all-to-all communication isn't a requirement. If you sue such a protocol, what fraction of the total node RAM will be occupied with routing tables? Does it fit into the devices? What fraction of your network traffic will be spent on routing messages?

Reliability: the BLE network is inherently unreliable, but simply running TCP between smart devices and gateways is impossible (the TCP header is too large for a BLE network).

Furthermore, TCP is designed to handle constraints of the Internet, in particular high levels of congestion. It's likely a poor choice as a BLE routing protocol due to small BLE packet sizes, low-latency links. as well as very low loss rates in the network you have.   Dynamically sizing windows, backoff, etc probably aren't really needed in this network.

Naming/Hierarchy: Be careful about excessive use of hierarchy -- do node names really need to have a fixed hierarchical schema? Is it really important to assign FCS threads to specific buildings? Such designs limit flexibility (can't move nodes / balance work) and scalability (limit the maximum number of nodes). A design with a rigid hierarchy may be reasonable but you should argue for its use.

Relatedly, overly static network designs, e.g., where some external process configures the network topology once and it never changes, or where a person has to enforce complicated topological by hand properties is probably not reasonable. Such static configuration does simplify routing but can't be ominously difficult to maintain over time.

How does a device learn its identifier? Is it fixed at configuration time? Note, that the DP description now states that the identifier might be incorrectly configured.

Messages: You should specify all messages in detail. Very important: what messages are send upstream (to the FCS) and down-stream. How do the devices know which direction to go for which message.

Routing: Your routing protocol should describe in detail how to route data to the FCS and from the FCS to the smart devices. Create a visual? example to explain it. Show what is stored where, how this information is used, etc.

Can there be congestion in your network / queue build up? Do you drop old or new data when needed?

How, if at all do you use local storage on repeaters and gateways?

Designs need to handle the fact that campus is multiple levels and can have strange shapes.

Does your design clearly specify communication protocols spec, format of network messages, etc? Some specific things to think about: designs need to specify how data is segmented into 20 byte BLE packets / 64 bit headers, and how these packets are reassembled into larger data blocks when needed. Specific packet format considerations, like different types of messages, bits to ensure deliver of high priority messages, etc should be included. Designs should specify how 20 byte BLE packets are re-encapsulated into 1500 byte gateway packets. Most importantly, how is a video frame split up into packets and re-assembled. What happens if a packet is dropped. What happens if the camera fails during the submission. In some designs the main thread would probably stall forever.

Design should argue that the network connects all nodes, without using too many redundant repeaters (of either type).

For the final report, does your design include calculations for end-to-end latency of data delivery?   Does it describe / quantify possible queuing delays / bottlenecks and propose a solutions to them if they can occur? What about calculations for total data storage on nodes and FCS? When is data dropped / discarded?

!!!!!!!!!!!   Very very important - Device limitations   !!!!!!!!!!!
There are several strong device limitations mentioned throughout the proposal. For example BLE repeater can have at most 16 connections active at a time AND if there are N connections active on a repeater, each connection will get at most 1/Nth of the total bandwidth. Does your design deal with those limitations? How many active connections does every repeater/gateway have? What happens if there are no free connections? Does your design deal with the 1/Nth limitation (e.g., large chains might have a problem with that). You absolutely need some back-of-the-envelope calculations here.

FCS Issues

Software: Your design should specify the software you deploy on the FCS (Section 2.3) as well as on the smart device (Section 2.1). Incidentally, the software abstractions given in Tables 1-3 are function calls that are available on the smart devices themselves, not on the FCS.

If you have shared state and concurrency in your FCS, is shared state properly protected by locks?

How does you protocol give priority to nodes in crisis mode? Does it ensure timely delivery of all crisis mode data? Again back-of-the-envelope calculations and visualizations can help a lot here.

Be precise about exactly what physical data storage structure you use — a JSON file? A collection of database tables? A collection of files and directories? When when choosing a representation, think about what types of access you will need to the data (e.g., reading an individual value vs reading many historical values), and whether your physical representation supports this.

Some detail about control flow / main loop on the FCS is needed. How does it service all incoming requests? Pseudocode, flow chart, or a state-diagram is a good idea.

How many threads do you expect during normal operation and in the worst-case. A lot of active threads might cause thrashing. If you use a database how many transactions per second do you need to handle. Note, that storing large variable length binaries in many database systems is a problem.

What happens with firmware updates during crises mode. How are they buffered? When are they deployed?

Is it possible that a thread stalls forever because of a failure (e.g., the video frame is reconstructed but the camera failed during the transmissions or packets got lost)

Mapping device to rooms: Many proposals did not clarify how the mapping from device to rooms is done. What happens if two temperature/motion sensors in the same room report different temperatures/motion events.

Failure Scenarios

You should discuss various failure scenarios. What happens if a repeater or gateway fails. If a smart device is wrongly configured. When are the routing table updated. How much data can be lost?

Individual Comments

Your back-of-the-envelope cost calculations are great. BTW: to really make the case for a project like this, people often also point out how much savings the project would provide (e.g., XX in Energy savings, XX in less security personal, etc). Not really needed for the DP though.

What would be the worst case cost scenario for you physical layout? A separate very narrow building? How well would your design work on some of MITs buildings. If you want to keep that design, you need to make a very strong argument for it.

It is nice that the average numbers look all good, but it is unrealistic to achieve them. You can not perfectly place devices (e.g., a building might have a weird shape (Stata???) and you are not allowed to place devices outside of the building). So you have to talk about what happens in those scenarios as well.

The device ID might not be correctly set (see errata to DP). What happens then?

Link state routing is probably overkill (see general comments)

How video frames are sent it not fully clear. How does your design deal with the different package sizes, Bluetooth vs TCP.

Section 3.24: How exactly do cameras guarantee five fps? How does the camera know the network is overloaded? Can it happen?

Might it be possible that the temperature is constantly decreased with every minute without action?