Folks:
Adam Roach; Ben Campbell; Cullen Jennings; Daniel Petrie; Dean Willis; Gonzalo Camarillo; Jonathan Rosenberg; Robert Sparks

Conclusion :
Identified we need to tear this apart addressing the problem of having to take a long time to generate a response separately from the problem of something that was going to respond went away.

Summary:
The "Random Notes" below are the stuff I typed in the meeting. Right here I am going to describe what I took out of the meeting - this could be a totally warped impression of what happened - I was pretty tired. So read the random notes too and make your own summary.

The problem we are trying to solve is how to clean up state on all the elements. The major two cases we need to deal with are 1) something died and will never respond 2) something is taking a long time to respond. Case 2 splits into two sub cases. The first 2a where it is a automata that is holding up the response and 2b where something is waiting for input from a human before responding.

Some things to keep in mind when evaluating solutions are: systems use different values of T1. DNS SRV failures complicate things. A similar problem might be able to happen in INVITE transactions where things fork or CANCEL.

We brainstormed a few possible approaches to this problem.

- Set an expire time of some sort on the transaction
  - If this is absolute, then need synchronized time - Yuck
  - If we ignore the network time (the hops) and only count the processing time at each node (the hipitty) then absolute time would not be needed
- Make the time start at 64*T1 for the first node and shrink as the message traverses nodes
  - won't work
- Make the time at each node be large enough (i.e. current max hops * 64 * T1) that downstream nodes will timeout first.
  - 64*T1*70 = 37 minutes * 1000 transaction per second is a lot of state
- Have any transaction that is going to take awhile, return some code that more or less amounts to try back later with the same request and I will give you the answer then.
  - Not clear how this work when nodes fail instead of just being slow

The reason I came to the meeting was to say that no node should have to hold state forever and that it is reasonable to put some upper bound on how long things are allowed to take and still be a single transaction. I don't think the problem is solvable without this constraint.

Random Notes:

First question – should we kill proposal B?

> Considered how HTTP deals with the problem. Feeling was HTTP did not deal with it yet but would need to in the future with SOAP.
>
> Decided that right now we don't have enough information to kill proposal B – need to brainstorm some options.

Jonathan posed the problem as: Imagine the timeout is very long then figure out how to clean up state. There is state in user agent, proxies.

> Idea: allow the proxy to go stateless at some point.
>
> Consider invite stuff. Proxy may have to maintain state arbitrarily long. Solution was to have uas send provisional to refresh timers. Not sure this was true.
>
> If we make things arbitrarily long, how deal with crashed things…

The ideas of shrinking the timer: The uac will have timer 70*64*t1. Each proxy have max hops * 64 * t1. uas has timer 64 *t1 or could possibly max hops *54 *t1. This results in keeping state 37 minutes – all agree 37 minutes does not sound good.

Invite has this same problem if the cancel fails you can get this cascaded.

Really two problem here – one is failure – other is device is just slow doing it's backend processing or something.

Important point – not all systems have t1 at 500 ms – this can mess up things and we have to take it into account.

The DNS SRV fail over needs to be considered.

Disused the idea of saying you have this much time to process this message. It can get decremented along the way.

Is the whole discussion purely that the timer number is too low.

If we want to deal with human interaction, then it is a different problem that time for automata to respond.

If the UAS timer is large enough, then the only time that there is no response is when something failed.

One proposal is make timer bigger than t1*64 and then toss out the 408 somehow.

Do we let the client continues to allow the transaction to processed using some sort of message back or do we redo the request at some future time when the far side as completed the answer?