

# Re: Coordinating TCP projects

---

*Source:* <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/arch/2007-12/msg00113.html>

---

- *From:* Lawrence Stewart <[lawstewart@xxxxxxxxxxxxx](mailto:lawstewart@xxxxxxxxxxxxx)>
  - *Date:* Thu, 20 Dec 2007 02:50:21 +1100
- 

Hi Robert,

Comments inline.

Robert Watson wrote:

Dear all,

It is rapidly becoming clear that quite a few of us have Big Plans for the TCP implementation over the next 12–18 months. It's important that we get the plans out on the table now so that everyone working on these projects is aware of the larger context. This will encourage collaboration, but also allow us to manage the risks inevitably associated with having several simultaneous projects going on in a very complex software base. With that in mind, here are the large projects I'm currently aware of:

## Project Flag Wavers Status

---

TCP offload Kip Macy Moving to CVS and under review and testing; one supporting device driver.

TCP congestion control Sam Leffler, At least one prototype  
Rui Paulo, implementation, to move to p4  
Andre Oppermann,  
Kip Macy,  
Lawrence Stewart,  
James Healy

TCP overhaul Andre Oppermann Glimmer in eye, to move to p4.

TCP lock granularity/ Robert Watson Glimmer in eye, to occur in increased parallelism p4.

TCP timer unification Andre Oppermann, Previously committed, and to Mike Silbersack be reintroduced via p4.

Monitoring ABI cleanup Robert Watson Glimmer in eye, to occur in

## Re: Coordinating TCP projects

p4.

Looking at the above, it sounds like a massive amount of work taking place, so we will need to coordinate carefully. I'd like to encourage people to avoid creating unnecessary dependencies between changes, and to be especially careful in coordinating potentially MFCable changes. There are (at least) two conflicting scheduling desires in play here:

- A desire to merge MFCable changes early, so that they aren't entangled with un-mergeable changes. This will simplify merging and also maximize the extent to which testing in HEAD will apply to them once merged to RELENG\_7.
- A desire to merge large-scale infrastructural changes early so that they see the greatest exposure, and so that they can be introduced incrementally over a longer period of time to shake each out.

Both of these are valid perspectives, and will need to be balanced. I have a few questions, then, for people involved in these or other projects:

(0) Is your project in the above list? If not, could you send out a reply talking a bit about the project, who's involved, where it's taking place, etc.

Rui@ recently posted a TCP ECN patch that probably belongs in the list (<http://lists.freebsd.org/pipermail/freebsd-net/2007-November/015979.html>) unless it has already recently been committed.

Jim and I recently discussed the idea of implementing autotuning of the TCP reassembly queue size based on analysis of some experimental work we've been doing. It's a small project, but we feel it would be worth implementing. Details follow...

### Problem description:

Currently, "net.inet.tcp.reass.maxqlen" specifies the maximum number of segments that can be held in the reassembly queue for a TCP connection. The current default value is 48, which equates to approx. 69k of buffer space if MSS = 1448 bytes. This means that if the TCP window grows to be more than 48 segments wide, and a packet is lost, the receiver will buffer the next 48 segments in the reassembly queue and subsequently drop all the remaining segments in the window because the reassembly buffer is full i.e. 1 packet loss in the network can equate to many packet losses at the receiver because of insufficient buffering. This obviously has a negative impact on performance in environments where there is non-zero packet loss.

With the addition of automatic socket buffer tuning in FreeBSD 7, the ability for the TCP window to grow above 48 segments is going to be even more prevalent than it is now, so this issue will continue to affect connections to FreeBSD based TCP receivers.

We observed that the socket receive buffer size provides a good indication of the expected number of bytes in flight for a connection, and can therefore serve as the figure to base the size of the reassembly queue on.

## Re: Coordinating TCP projects

Basic project description:

- Make the reassembly queue's max length a per-connection variable to appropriately tailor the reassembly queue buffer size for each connection
- Piggyback automated reassembly queue sizing with the code that resizes the socket receive buffer
- The socket buffer tuning code already has the required infrastructure to cap the max buffer size, so this would implicitly limit the size of the reassembly queue
- If the socket buffer sizes were explicitly overridden using sockopts (e.g. to support large windows for particular apps), the reassembly queue would grow to accommodate only connections using the larger than normal receive buffer.
- The `net.inet.tcp.reass.maxsegments` tunable would still be left intact to ensure users can set a hard cap on the max amount of memory allowed for reassembly buffering.

(1) What is your availability to shepherd the project through its entire cycle, including early prototyping, design review, development, implementation review, testing, and the inevitable long debugging tail that all TCP projects have.

We should be able to run the reassembly queue project full cycle.

(2) When do you think your implementation will reach a prototype phase appropriate for an expanded circle of reviewers? When do you think it might be ready for commit? Keep in mind that we're now a month or so into the 18-month cycle for 8.0, and that all serious TCP work should be completed at least six months before the end of the cycle.

To be safe, I'll say we should have a prototype ready by the end of Feb 2008, though I suspect we'll have something ready sooner than that. Commit ready code should follow very shortly after that (few weeks at most), as we anticipate that the patch will be very simple.

(3) What potential interactions of note exist between your project and the others being planned. Are there explicit dependencies?

The "TCP Overhaul" project would possibly alter the location of the changes, but shouldn't affect the essence of the changes themselves. It's unlikely any of the other projects would affect this one.

Re: Coordinating TCP projects

(4) Do you anticipate an MFC cycle for your work to RELENG\_7?

Yes. A munged version could also be made available for RELENG\_6.... it just wouldn't be based on automatic receive buffer tuning, and would probably be based on a static calculation during connection initialisation.

I'd like for us to create a wiki page tracking these various projects, and pointing at per-project resources. Once the discussion has settled a bit, I can take responsibility for creating such a page, but will need everyone involved to help maintain it, as well as to maintain pages (on the wiki or elsewhere) regarding the status of the projects. I think it also makes a lot of sense for participants in the projects to send occasional updates and reports to net@/arch@ in order to keep people who can't track things day-to-date in the loop, and to invite review.

Sounds fair.

[snip]

Cheers,  
Jim and Lawrence

---

freebsd-arch@xxxxxxxxxxx mailing list

<http://lists.freebsd.org/mailman/listinfo/freebsd-arch>

To unsubscribe, send any mail to "freebsd-arch-unsubscribe@xxxxxxxxxxx"