Item 318
git_comments:

1. If trans is already a *THeaderTransport, it will be returned as is.

git_commits:

1. **summary:** THRIFT-4612: Avoid double wrapping THeaderTransport
   message: THRIFT-4612: Avoid double wrapping THeaderTransport Client: go Previously the library didn't check against double wrapping, so when NewTSimpleServerN was used with both THeaderTransportFactory and THeaderProtocolFactory, inside THeaderProtocolFactory the transport was double wrapped with THeaderTransport. Worse, the transport still appeared to work, because THeaderTransport is backwards compatible with TBinaryProtocol and TCompactProtocol so the outer layer of THeaderTransport wrapper (the one directly accessible from the protocol) would assume the client doesn't support THeader and fallback. So when double wrapping happened, it appeared like everything was fine, except you couldn't get the headers from the protocol (because they were in the inner THeaderTransport, not the outer one that's directly accessible from the protocol), making it very hard to debug. This commit adds protection against such double wrapping. This closes #1839.
   label: code-design

github_issues:

github_issues_comments:

github_pulls:

1. **title:** THRIFT-4612: Avoid double wrapping THeaderTransport
   body: Client: go Previously we did not check against that, so when people using NewTSimpleServerN with both THeaderTransportFactory and THeaderProtocolFactory, inside THeaderProtocolFactory we will actually double wrap the transport with THeaderTransport. What make things worse is that in that case, the transport still appear to work, because THeaderTransport has backward compatible to TBinaryProtocol and TCompactProtocol so the outer layer of THeaderTransport wrap (the one directly accessible from the protocol) will only think that the client does not support THeader, and fallback. So when double wrapping happens, it will appear that everything works, except you cannot get the headers from the protocol (because they are actually in the inner THeaderTransport, not the outer one that's directly accessible from the protocol), makes it very hard to debug. This change adds protection against double wrapping.
2. **title:** THRIFT-4612: Avoid double wrapping THeaderTransport
   body: Client: go Previously we did not check against that, so when people using NewTSimpleServerN with both THeaderTransportFactory and THeaderProtocolFactory, inside THeaderProtocolFactory we will actually double wrap the transport with THeaderTransport. What make things worse is that in that case, the transport still appear to work, because THeaderTransport has backward compatible to TBinaryProtocol and TCompactProtocol so the outer layer of THeaderTransport wrap (the one directly accessible from the protocol) will only think that the client does not support THeader, and fallback. So when double wrapping happens, it will appear that everything works, except you cannot get the headers from the protocol (because they are actually in the inner THeaderTransport, not the outer one that's directly accessible from the protocol), makes it very hard to debug. This change adds protection against double wrapping.
3. **title:** THRIFT-4612: Avoid double wrapping THeaderTransport
   body: Client: go Previously we did not check against that, so when people using NewTSimpleServerN with both THeaderTransportFactory and THeaderProtocolFactory, inside THeaderProtocolFactory we will actually double wrap the transport with THeaderTransport. What make things worse is that in that case, the transport still appear to work, because THeaderTransport has backward compatible to TBinaryProtocol and TCompactProtocol so the outer layer of THeaderTransport wrap (the one directly accessible from the protocol) will only think that the client does not support THeader, and fallback. So when double wrapping happens, it will appear that everything works, except you cannot get the headers from the protocol (because they are actually in the inner THeaderTransport, not the outer one that's directly accessible from the protocol), makes it very hard to debug. This change adds protection against double wrapping.

label: code-design
4. **title:** THRIFT-4612: Avoid double wrapping THeaderTransport
   body: Client: go Previously we did not check against that, so when people using NewTSimpleServerN with both THeaderTransportFactory and THeaderProtocolFactory, inside THeaderProtocolFactory we will actually double wrap the transport with THeaderTransport. What make things worse is that in that case, the transport still appear to work, because THeaderTransport has backward compatible to TBinaryProtocol and TCompactProtocol so the outer layer of THeaderTransport wrap (the one directly accessible from the protocol) will only think that the client does not support THeader, and fallback. So when double wrapping happens, it will appear that everything works, except you cannot get the headers from the protocol (because they are actually in the inner THeaderTransport, not the outer one that's directly accessible from the protocol), makes it very hard to debug. This change adds protection against double wrapping.
   label: code-design
5. **title:** THRIFT-4612: Avoid double wrapping THeaderTransport
   body: Client: go Previously we did not check against that, so when people using NewTSimpleServerN with both THeaderTransportFactory and THeaderProtocolFactory, inside THeaderProtocolFactory we will actually double wrap the transport with THeaderTransport. What make things worse is that in that case, the transport still appear to work, because THeaderTransport has backward compatible to TBinaryProtocol and TCompactProtocol so the outer layer of THeaderTransport wrap (the one directly accessible from the protocol) will only think that the client does not support THeader, and fallback. So when double wrapping happens, it will appear that everything works, except you cannot get the headers from the protocol (because they are actually in the inner THeaderTransport, not the outer one that's directly accessible from the protocol), makes it very hard to debug. This change adds protection against double wrapping.
6. **title:** THRIFT-4612: Avoid double wrapping THeaderTransport
   body: Client: go Previously we did not check against that, so when people using NewTSimpleServerN with both THeaderTransportFactory and THeaderProtocolFactory, inside THeaderProtocolFactory we will actually double wrap the transport with THeaderTransport. What make things worse is that in that case, the transport still appear to work, because THeaderTransport has backward compatible to TBinaryProtocol and TCompactProtocol so the outer layer of THeaderTransport wrap (the one directly accessible from the protocol) will only think that the client does not support THeader, and fallback. So when double wrapping happens, it will appear that everything works, except you cannot get the headers from the protocol (because they are actually in the inner THeaderTransport, not the outer one that's directly accessible from the protocol), makes it very hard to debug. This change adds protection against double wrapping.
   label: code-design
7. **title:** THRIFT-4612: Avoid double wrapping THeaderTransport
   body: Client: go Previously we did not check against that, so when people using NewTSimpleServerN with both THeaderTransportFactory and THeaderProtocolFactory, inside THeaderProtocolFactory we will actually double wrap the transport with THeaderTransport. What make things worse is that in that case, the transport still appear to work, because THeaderTransport has backward compatible to TBinaryProtocol and TCompactProtocol so the outer layer of THeaderTransport wrap (the one directly accessible from the protocol) will only think that the client does not support THeader, and fallback. So when double wrapping happens, it will appear that everything works, except you cannot get the headers from the protocol (because they are actually in the inner THeaderTransport, not the outer one that's directly accessible from the protocol), makes it very hard to debug. This change adds protection against double wrapping.
8. **title:** THRIFT-4612: Avoid double wrapping THeaderTransport
   body: Client: go Previously we did not check against that, so when people using NewTSimpleServerN with both THeaderTransportFactory and THeaderProtocolFactory, inside THeaderProtocolFactory we will actually double wrap the transport with THeaderTransport. What make things worse is that in that case, the transport still appear to work, because THeaderTransport has backward compatible to TBinaryProtocol and TCompactProtocol so the outer layer of THeaderTransport wrap (the one directly accessible from the protocol) will only think that the client does not support THeader, and fallback. So when double wrapping happens, it will appear that everything works, except you cannot get the headers from the protocol (because they are actually in the inner THeaderTransport, not the outer one that's directly accessible from the protocol), makes it very hard to debug. This change adds protection against double wrapping.
9. **title:** THRIFT-4612: Avoid double wrapping THeaderTransport

body: Client: go Previously we did not check against that, so when people using NewTSimpleServerN with both THeaderTransportFactory and THeaderProtocolFactory, inside THeaderProtocolFactory we will actually double wrap the transport with THeaderTransport. What make things worse is that in that case, the transport still appear to work, because THeaderTransport has backward compatible to TBinaryProtocol and TCompactProtocol so the outer layer of THeaderTransport wrap (the one directly accessible from the protocol) will only think that the client does not support THeader, and fallback. So when double wrapping happens, it will appear that everything works, except you cannot get the headers from the protocol (because they are actually in the inner THeaderTransport, not the outer one that's directly accessible from the protocol), makes it very hard to debug. This change adds protection against double wrapping.

10. **title:** THRIFT-4612: Avoid double wrapping THeaderTransport
    body: Client: go Previously we did not check against that, so when people using NewTSimpleServerN with both THeaderTransportFactory and THeaderProtocolFactory, inside THeaderProtocolFactory we will actually double wrap the transport with THeaderTransport. What make things worse is that in that case, the transport still appear to work, because THeaderTransport has backward compatible to TBinaryProtocol and TCompactProtocol so the outer layer of THeaderTransport wrap (the one directly accessible from the protocol) will only think that the client does not support THeader, and fallback. So when double wrapping happens, it will appear that everything works, except you cannot get the headers from the protocol (because they are actually in the inner THeaderTransport, not the outer one that's directly accessible from the protocol), makes it very hard to debug. This change adds protection against double wrapping.

github_pulls_comments:

1. @dcelasun @jeking3
2. > I did not add the protection on NewTHeaderTransport level, because I consider that low-level and don't want to make it impossible to explicitly do the double wrapping. Hmm, this doesn't sound right to me. Is there a real world use case for double wrapping?
3. **body:** > > I did not add the protection on NewTHeaderTransport level, because I consider that low-level and don't want to make it impossible to explicitly do the double wrapping. > > Hmm, this doesn't sound right to me. Is there a real world use case for double wrapping? For some testing, probably? My main consideration is that if we prevent double wrapping at `NewTHeaderTransport` level, then there's no way to do it. If you feel that's better I'm happy to do it that way.
   label: code-design
4. **body:** Without a real use case I think it's better to avoid letting users shoot themselves in the foot, especially since accidental double wrapping is hard to debug. So yes, please update the PR.
   label: code-design
5. @dcelasun Done. Commit message also updated.

github_pulls_reviews:

1. > itself will be returned it will be returned as is.
2. Fixed.
3. Really fixed now :)

jira_issues:

1. **summary:** Add THeader for Go
   description: It will be useful to have headers supported in thrift's Go library. I noticed that support for fbthrift's THeader was added to the C++ implementation, and am wondering if it is possible to do the same for Go as well. I am happy to make the change myself, but want to check if there is reason to not do that and/or if there is anything I need to keep in mind while working on it.
2. **summary:** Add THeader for Go
   description: It will be useful to have headers supported in thrift's Go library. I noticed that support for fbthrift's THeader was added to the C++ implementation, and am wondering if it is possible to do the same for Go as well. I am happy to make the change myself, but want to check if there is reason to not do that and/or if there is anything I need to keep in mind while working on it.
3. **summary:** Add THeader for Go

description: It will be useful to have headers supported in thrift's Go library. I noticed that support for fbthrift's THeader was added to the C++ implementation, and am wondering if it is possible to do the same for Go as well. I am happy to make the change myself, but want to check if there is reason to not do that and/or if there is anything I need to keep in mind while working on it.

jira_issues_comments:

1. I'd be happy to see a PR for this, please go ahead. Just try to avoid breaking BC and add tests wherever appropriate. I can have more feedback once I see some code.