TIPC

Contents

About

TIPC is Transparent Inter-process Communication. You can find more info on Sourceforge

Community Contact

The best way to contact TIPC maintainers is through the mailing list. The archive may also contain an answer to your question.

Development

Ericsson is active in the maintenance and feature development of TIPC. For details, see the internal TIPCBacklog#Roadmap

Upstream Maintainer and Author of TIPC:

- Jon Maloy - jon.maloy@ericsson.com

SLES / Suse Maintainers for SLES11, SLES12SP0, SLES12SP2:

- Canh Duc Luu - canh.d.luu@dektech.com.au
- Hoang Huu Le - hoang.h.le@dektech.com.au
- Tung Quang Nguyen - tung.q.nguyen@dektech.com.au
- Parthasarathy Bhuvaragan - parthasarathy.bhuvaragan@ericsson.com
- Mohan -

Known users of the TIPC protocol

USER	SOCKET_TYPE	I/O MODEL	OTHER	Contacts
vDicos	stream	Non-blocking	epoll EDGE_TRIGGERRED. Deployed only on Virtual systems, but RDA tests them on Native systems too.	Tamás Végh <tamas.vegh@ericsson.com> Balázs Tuska <balazs.tuska@ericsson.com>
DBS/DBN	seqpacket	Non-blocking	primarily one way communication, Server pushes large amount of data as soon as clients connect.	Gergely Kiss <gergely.kiss@ericsson.com> Sándor Rédei <sandor.redei@ericsson.com>
CoreMW/OpenSaf	dgram/rdm	-	primarily unicast, in restart cases many senders send to a single receiver causing receive buffer overflows.	Hans Nordebäck <hans.nordeback@ericsson.com> Anders Widell <anders.widell@ericsson.com>
eVIP	dgram/rdm	-	primarily multicast/broadcast, nametable must be consistent on all nodes.	Leif Andersen <leif.andersen@tieto.com> Lars Ekman G <lars.g.ekman@ericsson.com>
APZ	-	-	proprietary implementation of the TIPC stack inside the APZ VM.	Per Sundström XP <per.xp.sundstrom@ericsson.com>
TSP	-	-	proprietary implementation of the TIPC stack	Hiroshi Doyu <hiroshi.doyu@ericsson.com>
IPOS	-	-	Used as IPC between Control cards and Line cards.	IPOS-OWNERS-LINUX <PDLWNERSLI@pdl.internal.ericsson.com>

Process for getting patches into the distributions

getting patches into SUSE

Getting SUSE kernel source from uname

SUSE kernel source

TIPC Infrastructure/ First time Setup

Administrative stuff

1. Create an account in Gerrit Central add your ssh keys
2. Create an account / Request access for Jenkins

Once you have created an account in Gerrit central, and send the request below to current tipc maintainers.

```
Hello Admins,
Please add me to the Group linux-tipc
https://gerrit.ericsson.se/#/admin/groups/2090,members
```

Currently TIPC has the following repositories in gerrit.ericsson.se/projects/ and linux-tipc group has access to all of it.

- Build and Test Repo ("known as lab repo"):

```
linux/lab
```

- Delivery distribution repositories, containing only the tipc patches:

```
linux/tipc-sles12
linux/tipc-sles12sp2
linux/tipc-sles11
linux/tipc-rhel
```

lab repository information

Cloning

Clone from your local mirror:

```
$ git clone --recursive ssh://gerritmirror.rnd.ki.sw.ericsson.se:29418/linux/lab.git
```

Building

For x86

```
$ make <-j x>
```

Build requirements

There's a lot of package dependencies to build all components in the environment. Make sure you have the following installed:

```
autoconf
bison
libdaemon-dev
libdb5.1-dev
libjansson-dev
libnl-3
libnl-cli-3-dev
libnl-genl-3-dev
libnl-route-3-dev
expect
flex
```

Installing

Install virsh & kvm

```
$ sudo apt-get install libvirt-bin qemu-kvm
```

Add virsh sudoers rules

```
$ sudo visudo

YOUR-USER ALL=NOPASSWD: /usr/bin/virsh
```

Make sure you have the tcl package tdom

```
$ sudo apt-get install tdom
```

Choose one of the virtual clusters

```
$ ./test/lat2/lat.pl -L
```

Install the cluster

```
$ ./test/lat2/lat.pl -e CLUSTER test/suites/virsh/virsh_install.yaml
```

Testing

Physical LAB Infrastructure

```
 Host       Console                    port    Slot     Info
-------------------------------------------------------------
SCX    digi-srv.lab.linux.ericsson.se 7020     0       Left side switch
tipsy  digi-srv.lab.linux.ericsson.se 7022     1       Mgmt, NFS server for /home/, build host, tftp/boot server for Test blades from /srv/t
lab1   digi-srv.lab.linux.ericsson.se 7023     3       Test blade
lab2   digi-srv.lab.linux.ericsson.se 7024     5       Test blade
lab3   digi-srv.lab.linux.ericsson.se 7025     7       Test blade
lab4   digi-srv.lab.linux.ericsson.se 7026     9       Test blade
lab5   digi-srv.lab.linux.ericsson.se 7027     11      Test blade
lab6   digi-srv.lab.linux.ericsson.se 7028     13      Test blade (re-install sles12sp2 and use as VM host, user/pass: root/rootroot)
grant  digi-srv.lab.linux.ericsson.se 7029     15      Jenkins
poe    digi-srv.lab.linux.ericsson.se 7030     17      VM host (Canh / Tung / Hoang)
daly   digi-srv.lab.linux.ericsson.se 7031     19      VM host (Partha / Jon)
SCX    digi-srv.lab.linux.ericsson.se 7021     25      Right side switch

https://digi-srv.lab.linux.ericsson.se (admin/admin)
```

Virtual hosts

The VM host (daly and tipsy) can host the following test configurations:

```
   Setup            cluster_size   Info
-------------------------------------------------------------------------
virtual-small         2            192.168.123.101/2
virtual-medium        4            192.168.124.101/4    can run simultaneously with small on the same host
virtual-large         8            192.168.125.101/8    overcommitting CPU
virtual-mega          16           192.168.126.101/16   overcommitting CPU
```

Basic Node configuration parameters

The nodes in the cluster has the following configurations.
3 network interfaces named:

```
ctrl  - For O&M access, not for tipc traffic.
data0 - First traffic interface
data1 - Second traffic interface
```

These interfaces use the virtio driver of linux. This can be changed to say e1000, by editing the env.yaml file at interface->name->model->type. Each interface of a node is connected to the rest of the nodes using Linux bridge.

```
sudo brctl show
bridge name     bridge id               STP enabled     interfaces
tipc-large-0            8000.52540086c951       no              tipc-lare-0-nic
tipc-large-1            8000.5254000db832       no              tipc-lare-1-nic
tipc-large-c            8000.525400ee559d       no              tipc-lare-c-nic
tipc-medium-0           8000.525400dde8e7       no              tipc-medm-0-nic
                                                                vnet1
                                                                vnet10
                                                                vnet4
                                                                vnet7
tipc-medium-1           8000.525400a4705a       no              tipc-medm-1-nic
                                                                vnet11
                                                                vnet2
                                                                vnet5
                                                                vnet8
tipc-medium-c           8000.525400297741       no              tipc-medm-c-nic
                                                                vnet0
                                                                vnet3
                                                                vnet6
                                                                vnet9
tipc-mega-0             8000.5254004391c3       no              tipc-mega-0-nic
tipc-mega-1             8000.5254002f8a3d       no              tipc-mega-1-nic
tipc-mega-c             8000.525400a1e3f9       no              tipc-mega-c-nic
tipc-small-0            8000.5254000611b6       no              tipc-smal-0-nic
                                                                vnet13
                                                                vnet16
tipc-small-1            8000.5254007e09e6       no              tipc-smal-1-nic
                                                                vnet14
                                                                vnet17
tipc-small-c            8000.525400246c43       no              tipc-smal-c-nic
                                                                vnet12
                                                                vnet15
```

Only ssh is enabled on these nodes for root user without password.

```
The ip address for ssh is derived from the env.yaml file based on nodes->ip.
The nodes are created with 4 CPU's. This changed be modified by specifying -o virsh_vcpu=<cnt> option to virsh_install.yaml.
```

GDB & Crash tool

The kernel images are built with kdump support.

```
If the kernel crashes, they copy the crash kernel to /tmp/virsh_export/tipc-<small/medium/large/mega>.
You can use crash tool to inspect these dumps. Its very effective way to troubleshoot issues.
```

The vm environment also supports gdb, which is specified using -o virsh_gdb=1 to virsh_install.yaml.

```
The gdb port numbers reserved per node instance will be printed on the console.
use gdb on vmlinux and connect to the gdbserver port.
```

This is very efficient way to understand code flow or inspect code instead of printk.

Depending on the cluster size and the type of bug, sometimes generated dumps results in low memory situation on the hypervisor.
So, you need to remove them manually to free up the memory.

```
sudo rm -rf $(sudo find /tmp/virsh_export -name vmcore)
```

Running Test

Run all available tests any running cluster

```
 $ ./test/lat2/lat.pl -e CLUSTER test/suites/all.yaml
```

Making a change in the main repo (lab)

```
 1. Make your change
 2. Commit your change with a sign-of and gerrit change-id
 3. Push to refs/for/master
```

Updating Linux

The linux directory in the lab repository is intentionally left empty. The user needs to add the desired remote repository and clone the content into it.

Updating Linux mirrors

Add the following remote repositories and create your own branch to track them.

```
$ git remote add suse https://github.com/openSUSE/kernel
$ git remote add mainline git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git
$ git remote add net git://git.kernel.org/pub/scm/linux/kernel/git/davem/net.git
$ git remote add net-next git://git.kernel.org/pub/scm/linux/kernel/git/davem/net-next.git
```

Next, fetch all the changes from these repositories. The first update takes a while as the kernel is BIG.

```
$ git remote update
$ git fetch --tags
```

For SLES

suse remote is used to build the kernel image for our distribution testing. Suse has branches for each release and its corresponding service packs. This makes it easy to check the suse tree for a specific bug fix or commits.

```
$ git branch -a
  remotes/suse/SLE11-SP3
  remotes/suse/SLE11-SP4
  remotes/suse/SLE12
  remotes/suse/SLE12-SP1
  remotes/suse/SLE12-SP1-ARM
  remotes/suse/SLE12-SP2
  remotes/suse/SLE12-SP3
  remotes/suse/SLE15
  remotes/suse/master
  remotes/suse/openSUSE-13.2
  remotes/suse/openSUSE-42.1
  remotes/suse/openSUSE-42.2
  remotes/suse/openSUSE-42.3
  remotes/suse/stable
```

How to create kernel commits

Tons of information is available online, read them. Here are my 5-cents:

Commit message:

```
1: Current solution.
2: Problem statement (why the current solution is not good)
3: Your new solution, and why it is better than any alternative solution, if any.
```

Checklist before sending:

```
- Local variable declarations shall be ordered from longest to shortest line (also known as "reverse christmas tree format").
- check for unnecessary typecast and braces.
- Checking with sparse, look for lock and prototype warnings
  make C=2 M=net/tipc
- Checking with smatch, static code analyser
  ~/Projects/smatch/smatch_scripts/kchecker --spammy drivers/modified_file.c
  make CHECK="~/Projects/smatch/smatch -p=kernel" C=1 M=net/tipc clean
  make CHECK="~/Projects/smatch/smatch -p=kernel" C=1 M=net/tipc
- Run testcases
- Run checkpatch --strict
 for A in {1..10}; do git diff HEAD~$((A-1)) HEAD~$A |  ./scripts/checkpatch.pl -; done
```

For iproute2

The fixes in tipc ultility is sent to iproute2 package to netdev mailing list. tipc-config is deprecated, hence is no longer maintained.

```
make -C tipc clean && CFLAGS=-Wunused-variable make -C tipc
for A in {1..10}; do git diff HEAD~$((A-1)) HEAD~$A |  ../../linux/scripts/checkpatch.pl -; done
```

Sending a change for community review

Patches are sent to tipc-discussions mailing list and discussed until they are approved or accepted. Next, we need to send these to netdev (networking mailing list) to be accepted. Bugs are sent to net ("prefixed by net as the subject") mailing list. The development updates are prefixed by net-next. The patches are always versioned, making it easier to follow.

Creating patches:

```
git format-patch -n --subject-prefix="PATCH v1" --cover-letter -M origin/master -o outgoing/
```

Editing Cover-letter and mailing

```
vi outgoing/0000-*
git send-email outgoing/* --annotate --to tipc-discussion@lists.sourceforge.net --to jon.maloy@ericsson.com --to maloy@donjonn.com --to ying.
```

I found a mail server which doesn't mess with the incoming / outgoing mails. Check for smtpserver in gitconfig section.

Backporting Changes form Upstream / Mailbox

Refresh kernel Source repository

Perform a refresh of your kernel source repository, to get the latest changes.

```
   git pull
   git remote update
```

Create a branch

Checkout a new branch <new> to start the porting work. The old branch is called <old>

Collect the delta

Choose the target files to update and walk through the history between the current version and target version if they share a common ancestor.

```
git log --oneline HEAD..<target> -20 net/tipc/subscr.c net/tipc/server.c
2017-03-28 Ying Xue  7efea60dcffc tipc: adjust the policy of holding subscription kref
2017-03-28 Ying Xue  139bb36f754a tipc: advance the time of deleting subscription from subscriber->subscrp_list
2017-03-21 Ying Xue  557d054c01da tipc: fix nametbl deadlock at tipc_nametbl_unsubscribe
2017-01-24 Parthasarathy Bhuvaragan  35e22e49a5d6 tipc: fix cleanup at module unload
2017-01-24 Parthasarathy Bhuvaragan  4c887aa65d38 tipc: ignore requests when the connection state is not CONNECTED
2017-01-24 Parthasarathy Bhuvaragan  9dc3abdd1f7e tipc: fix nametbl_lock soft lockup at module exit
2017-01-24 Parthasarathy Bhuvaragan  fc0adfc8fd18 tipc: fix connection refcount error
2017-01-24 Parthasarathy Bhuvaragan  d094c4d5f5c7 tipc: add subscription refcount to avoid invalid delete
2016-06-23 Amitoj Kaur Chawla  810bf1103363 tipc: Use kmemdup instead of kmalloc and memcpy
2016-05-17 Eric Dumazet  b91083a45e4c tipc: block BH in TCP callbacks
2016-04-27 Dan Carpenter  b43586576e54 tipc: remove an unnecessary NULL check
2016-04-12 Parthasarathy Bhuvaragan  333f796235a5 tipc: fix a race condition leading to subscriber refcnt bug
```

Use helper script

You can use the helper script in the distro repository, which looks the the patch-id (shasum of diff patch), instead of the commit-id and walks between branches and reports the following:

```
$ ~/tipc-sles12sp2/scripts/analyze_git_commits.sh HEAD sles12sp2_madhu_rest_no_link

Cherry-picked patches cleanly applied
----------------------------------------
----------------------------------------

Cherry-picked patches applied with fuzz
----------------------------------------
de6e68467c9e tipc: fix nametbl deadlock at tipc_nametbl_unsubscribe
6564720d70cf tipc: fix connection refcount error
6ed79b77a508 tipc: ignore requests when the connection state is not CONNECTED
8997bc92cbac tipc: adjust the policy of holding subscription kref
2a872640da5b tipc: add subscription refcount to avoid invalid delete
d436a138d233 tipc: advance the time of deleting subscription from subscriber->subscrp_list
586d4a0df437 tipc: Use kmemdup instead of kmalloc and memcpy
426e14e48f64 tipc: fix nametbl_lock soft lockup at module exit
f32fc4e5331f tipc: fix cleanup at module unload
a5ea2a400746 tipc: block BH in TCP callbacks
----------------------------------------

Unique Patch Sets
----------------------------------------
ee32dd9f7552 tipc: fix a race condition of releasing subscriber object
7cc60cea7a29 tipc: remove subscription references only for pending timers
----------------------------------------

Total:12 Missing:2 Clean:0 Fuzzy:10 Result:OK
```

Updating KABI first time

Patching tipc header files in KABI user space is usually a no-no during back porting. However, this workaround was needed to bump new features as we update our kernels very seldom. In order to add newer kernel features (like link monitoring) on the older kernel, I did a hack as described the following commit.

```
commit f6f11e2af565fe5156c2cb0938a0a38dccbbb8af
Author: Parthasarathy Bhuvaragan <parthasarathy.bhuvaragan@ericsson.com>
Date:   Thu Oct 20 10:41:54 2016 +0200

    tipc: compat workaround for new netlink attributes

    In this commit, we copy the tipc_netlink.h from the include directory
    of user and place in tipc source directory.
    We adapt all the references in source to point to this new
    tipc_netlink.h.

    From now on, we will adapt all the upstream changes for tipc_netlink.h
    in the tipc_netlink.h placed in the source directory.

diff --git a/net/tipc/core.h b/net/tipc/core.h
-#include <linux/tipc_netlink.h>
+#include "tipc_netlink.h"
```

Future KABI updates

This way, if we need to update the tipc_netlink.h in future, we do:

```
git format-patch -1 <commit#>
git apply -p 4 --directory "net/tipc" <commit#>
```

Cherry-picking commits

Manually check if the commit to be back ported existed already in your tree. Else "git cherry-pick" will inform that this is an empty commit and you abort the cherry-pick. Use "-x" flag while cherry-picking, this will add the upstream reference. git cherry-pick -x b91083a45e4c

If the commit is not in upstream: - a git diff, then use git apply and commit the change. - a patch in mailbox, then use git apply

Verify the change summary

Now do a git diff --stat from the HEAD to previous branch and check the commits and the diff.

```
$ git diff --stat HEAD/<new> <old>
 net/tipc/name_table.c |   2 --
 net/tipc/server.c     |  59 +++++++++++++++++++++--------------------------
 net/tipc/subscr.c     | 127 +++++++++++++++++++++++++++++++++++++++++------------------------------------------------
 net/tipc/subscr.h     |   4 ---
 4 files changed, 92 insertions(+), 100 deletions(-)
```

Verify the change content

Check the commit deltas.

```
$ git log --oneline HEAD...<old>
7cc60cea7a29 tipc: remove subscription references only for pending timers
ee32dd9f7552 tipc: fix a race condition of releasing subscriber object
8997bc92cbac tipc: adjust the policy of holding subscription kref
d436a138d233 tipc: advance the time of deleting subscription from subscriber->subscrp_list
de6e68467c9e tipc: fix nametbl deadlock at tipc_nametbl_unsubscribe
f32fc4e5331f tipc: fix cleanup at module unload
6ed79b77a508 tipc: ignore requests when the connection state is not CONNECTED
426e14e48f64 tipc: fix nametbl_lock soft lockup at module exit
6564720d70cf tipc: fix connection refcount error
2a872640da5b tipc: add subscription refcount to avoid invalid delete
586d4a0df437 tipc: Use kmemdup instead of kmalloc and memcpy
a5ea2a400746 tipc: block BH in TCP callbacks
```

Rebase the disto patch tree

Once ready, move to distro patch repository and pull the latest changes:

```
cd ~/tipc-sles12sp2
git pull
```

Prepare patches for distro

Move back to the kernel tree and do:

```
../scripts/convert_commit_to_patch_file.sh <old> ~/tipc-sles12sp2
```

If the conversion was ok, you should see the patches in the distro and the output looks like:

```
Copied 12 patches from kernel-tree to /home/qparbhu/tipc-sles12sp2/src
```

=====Commit & Review

1. Move to the distro patch repository and add the patches, update changes and commit.
2. Push to gerrit, which should trigger a jenkins job and inform the reviewers.
3. If jenkins job failed, go through the logs and fix the issue. Post a new patch.
4. Once the patch set is verified, wait for review.
5. Once reviewed merge the change.

Best Practises

kernel pre-commit-hook

Place this hook in lab repo. This ensures that the every commit will have to be satisfy the rules specified here.

```
cat > .git/modules/linux/hooks/pre-commit
#!/bin/bash

set -o errexit

git diff --cached | scripts/checkpatch.pl --no-signoff -

echo "Checking with sparse, look for lock and prototype warnings"
make C=2 M=net/tipc

echo "Checking with smatch, static code analyser"
make CHECK="~/Projects/smatch/smatch -p=kernel" C=1 M=net/tipc
```

gitconfig

```
[user]
        email = XX
        name = XX
[core]
        pager = less -FRXS
        editor = vim
        abbrev = 12
[color]
        branch = auto
        diff = auto
        interactive = auto
        status = auto
[url "ssh://git.code.sf.net"]
    pushInsteadOf = git://git.code.sf.net
[alias]
```

```
        stree = log --since='one week ago' --graph --pretty=oneline --abbrev-commit --decorate --color
        tree = log --graph --abbrev-commit --decorate=full --color --oneline
        ftree = log --graph --pretty=oneline --abbrev-commit --all --decorate --color
        lines = log --date=short --pretty=\"%C(cyan)%ad %C(green)%aN  %C(yellow)%h %C(reset)%s\"
        flog = log --decorate=full --full-diff
        signoff-rebase = "!GIT_SEQUENCE_EDITOR='sed -i -re s/^pick/e/' sh -c 'git rebase -i $1 && while git rebase --continue; do git commit
        pdiff="!f() { git diff $1~1 $1 $2 $3; }; f"
        apply-include="!f() { git show $1 -- include | git apply -p4 -v --directory=net/tipc/; }; f"
        refresh = submodule update --recursive
        ldiff= diff --no-index
        lidiff= diff --no-index -w --ignore-blank-lines
[push]
        default = matching
[pretty]
        fixes = Fixes: %h (\"%s\")
[sendemail]
        chainreplyto = false
        smtpserver = ESESSHC011.ss.sw.ericsson.se
        smtpencryption = tls
        smtpuser = $USER
        smtpserverport = 25
        smtpsslcertpath = /etc/ssl/certs/ca-bundle.trust.crt
        suppresscc = self
        aliasesfile = /home/$USER/.git_aliases
        aliasfiletype = mutt
```

Vim Plugins

I use the following:
fugitive for Git integration
cscope.maps.vim - cscope
git:file.vim - kernel style syntax

utility scripts

gen_cscope.sh

We filter out not-so-commonly used files and create a database as the default filter is way too big.

```
 cat > ~/bin/gen_cscope.sh
#!/bin/bash -e

find $PWD                                                                \
        -path "$PWD/arch/*" ! \( -path "$PWD/arch/x86*" \) -prune -o       \
        -path "$PWD/kernel*"                                                \
        -path "$PWD/net*"                                                          \
        -path "$PWD/virt*"                                                         \
        -path "$PWD/ipc*"                                                          \
        -path "$PWD/mm*"                                                           \
        -path "$PWD/init*"                                                         \
        -path "$PWD/tools*"                                                        \
        -path "$PWD/ipc*"                                                          \
        -path "$PWD/include*"                                               \
        -path "$PWD/lib*"                                  \
        -path "$PWD/firmware*" -prune -o                   \
        -path "$PWD/sound*" -prune -o                      \
        -path "$PWD/.git*" -prune -o                       \
        -path "$PWD/crypto*" -prune -o                     \
        -path "$PWD/Documentation*" -prune -o              \
        -path "$PWD/security*" -prune -o                   \
        -path "$PWD/samples*" -prune -o                    \
        -path "$PWD/scripts*" -prune -o                    \
        -path "$PWD/drivers*" -prune -o                    \
        -path "$PWD/fs*" -prune -o                         \
        -path "$PWD/usr*" -prune -o                        \
        -path "$PWD/block*" -prune -o                      \
        -name "*.[chxsS]" -print > $PWD/cscope.files

cscope -b -q -k

printf "\nCscope Rebuilt database for %d files \n" $( cat $PWD/cscope.files | wc -l)
```

start_tipc.sh

```
 cat > ~/bin/start_tipc.sh
#!/bin/bash

CLUSTER="medium"
SLOT=""
REPO=$PWD
TEST_CASE=""
ONLY_BUILD=""
ONLY_TEST=""
INSTALL=""

help()
{
        echo "$0 [options]"
        echo "options: "
        echo "  -e <small/medium/large/mega>"
        echo "  -s <node_id> -t <test_case>"
        echo "  -i \"install image\""
        echo "  -b \"only_build\""
        echo "  -r \"only_test\""
}

build()
{
        BUILD='make -j 20 --output-sync=target install'

        if ! $BUILD ; then
                echo "Error Build Failed"
                exit 1
        fi
        printf "\nBUILD: PASS\n"
}
```

```
test()
{
        ENV='env LANGUAGE=en_US LC_ALL=en_US.UTF-8'
        SCRIPT="./lat2/lat.pl -e virtual-$CLUSTER -o virsh_vcpu=4 -c config/bearer/data1_eth.yaml"
        LAT_FLAG='--fatal'

        if [ $INSTALL ]; then
                TEST_CASE="suites/virsh/virsh_install.yaml cases/setup/initialize_tipc.tcl $TEST_CASE"
        fi
        ssh daly "cd $REPO && $ENV $SCRIPT $TEST_CASE $LAT_FLAG"
        if [ $? -ne 0 ]; then
                echo "ERROR running Test"
                exit 1
        fi
        printf "\nTEST : PASS\n"
}

while getopts ":he:s:t:bri" opt; do
        case $opt in
                h)
                        help
                        exit 0
                        ;;
                s)
                        SLOT=$OPTARG
                        ;;
                e)
                        CLUSTER=$OPTARG
                        ;;
                i)
                        INSTALL=1
                        ;;
                r)
                        ONLY_TEST=1
                        ;;
                b)
                        ONLY_BUILD=1
                        ;;
                t)
                        TEST_CASE=$OPTARG
                        ;;
                :)
                        echo "Option -$OPTARG requires an argument." >&2
                        exit 1
                        ;;
                \?)
                        echo "Invalid option: -$OPTARG" >&2
                        help
                        exit 1
                        ;;
        esac
done

if [ $ONLY_BUILD ]; then
        build
        exit 0
fi

if [ $ONLY_TEST ]; then
        test
        exit 0
fi
```

run_all_commits.sh

This script can be used along with git-bisect to find a faulty commit. Instead of the range, specify a single commit.

```
cat > ~/bin/run_all_commit.sh
#!/bin/bash

scriptname="$(basename $(readlink -f $0))"

if [ $# -ne 2 ]; then
        echo "Usage: $scriptname branch num_of_commits" >&2
        exit 1
fi

LINUX_BRANCH="$1"
END_OFFSET="$2"
REPO=~/upstream/lab

BUILD='make -j 20 --output-sync=target install'

ENV='env LANGUAGE=en_US LC_ALL=en_US.UTF-8'
#TEST_SCRIPT='./lat2/lat.pl -e virtual-small suites/virsh/virsh_install.yaml suites/all.yaml --fatal'
TEST_SCRIPT='./lat2/lat.pl -e virtual-small suites/virsh/virsh_install.yaml cases/setup/initialize_tipc.tcl --fatal'

BUILD_FAILED=0
TEST_FAILED=0
BUILD_PASSED=0
TEST_PASSED=0

cd $REPO

for (( A=$END_OFFSET; A>=0; A-- ))
do
        echo "RTS Iteration: $LINUX_BRANCH~$A @ $(date)";
        git -C linux checkout $LINUX_BRANCH~$A
        ID=$(git -C linux log --oneline -n1)

        echo "RTS Build Triggerred : $ID"
        $BUILD >& /dev/null
        if [ $? -ne 0 ]; then
                BUILD_FAILED=$(($BUILD_FAILED+1))
                echo "RTS ERROR During Build :$ID"
                continue
        fi
        BUILD_PASSED=$(($BUILD_PASSED+1))
```

```
        echo "RTS Build PASS: $ID"

        ssh daly "cd $REPO/test && $ENV $TEST_SCRIPT"
        if [ $? -ne 0 ]; then
                TEST_FAILED=$(($TEST_FAILED+1))
                echo "RTS ERROR running Test : $ID"
                continue
        fi

        TEST_PASSED=$(($TEST_PASSED+1))
        echo "RTS Test PASS: $ID"
done

printf "RTS Summary\n-----------------\n"
printf "Build pass=%d  fail=%d \n" $BUILD_PASSED $BUILD_FAILED
printf "Test  pass=%d  fail=%d \n\n" $TEST_PASSED $TEST_FAILED
```

gdbinit

```
cat > ~/.gdbinit
add-auto-load-safe-path /home/XXXX/<repo>/linux/scripts/gdb/vmlinux-gdb.py
```

crashrc

```
cat > ~/.crashrc
set scroll off
# mod -S
# bt -lFFF
```

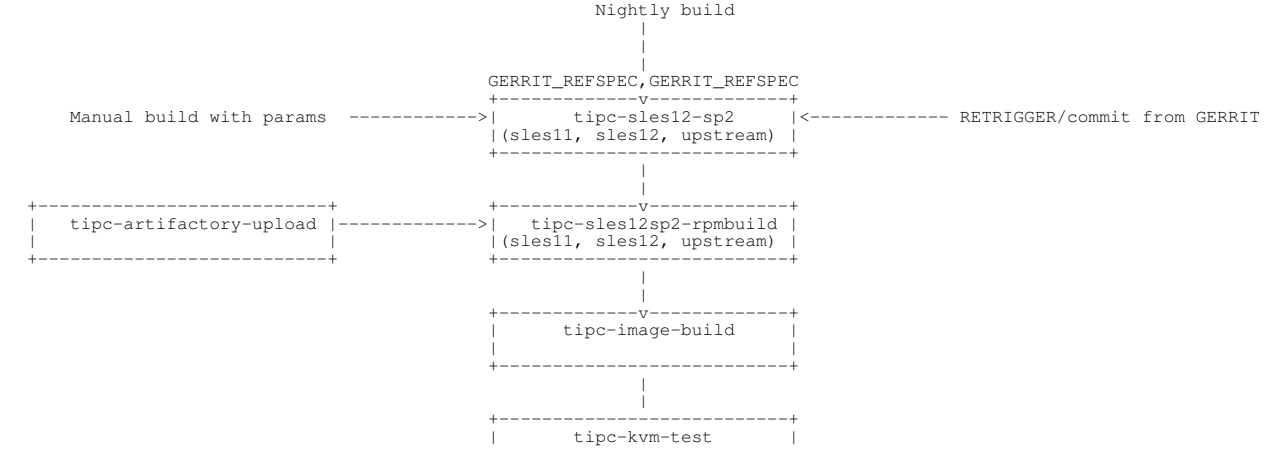
bash_aliases

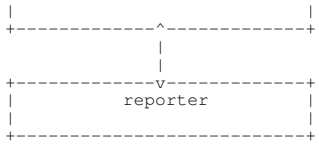
A small list of possibilities with our lab environment. These are some of the best practise which I think improves efficiency.

```
# Virsh
alias virsh="sudo virsh"
virsh_destroy() {
        for((a=1; a<=$1; a++))
        do
                virsh destroy "tipc-$2-node$a"
        done
}

alias des_mega="virsh_destroy 16 mega"
alias des_large="virsh_destroy 8 large"
alias des_medium="virsh_destroy 4 medium"
alias des_small="virsh_destroy 2 small"
alias vcs1="virsh console tipc-small-node1"
alias vcs2="virsh console tipc-small-node2"
alias vcl1="virsh console tipc-large-node1"
alias vcl2="virsh console tipc-large-node2"

# Tipc build commands
bimage() {
    make clean && make -j $(nproc) --output-sync=target install
}
export -f bimage
alias binstall="bimage && cd test/ && ~/bin/start_tipc.sh -e medium -ri && cd -"
alias binstmega="bimage && cd test/ && ~/bin/start_tipc.sh -e mega -ri && cd -"
alias birtl="bimage && cd test/ && ~/bin/start_tipc.sh -e large -ri -t suites/all.yaml && cd -"
alias birtm="bimage && cd test/ && ~/bin/start_tipc.sh -e medium -ri -t suites/all.yaml && cd -"

alias bkernel='make -j $(nproc) --output-sync=target kernel-install'
alias csbuild='find $PWD -name "*.[ch]" -print > ./cscope.files ; cscope -bq'
alias cskbuild="rm -f cscope.* && ~/bin/gen_cscope.sh"
function _load_crash() {
        if [ $# -ne 2 ]; then
                echo "$0 <small/medium/large> <slot_number>"
                return
        fi

        sudo chmod -R 777 "/tmp/virsh_export/tipc-$1/node$2/"
        sudo chown $USER: "/tmp/virsh_export/tipc-$1/node$2/vmcore"
        ~/git/crash/crash ./vmlinux "/tmp/virsh_export/tipc-$1/node$2/vmcore"
}
alias crash_setup=_load_crash
```

Jenkin

tipc jenkin workflow





TR handler

CBA/LDE

Before think tipc problem, just make sure don't have network disturbance, we can check the log of:

1. DRBD on SC (LDE)
2. Bonding (LDE)
3. NFS (LDE)
4. BFD (evip)
5. Driver