Back to the Future: Fault-tolerant Live Update with Time-traveling State Transfer

Cristiano Giuffrida Călin Iorgulescu Anton Kuijsten
Andrew S. Tanenbaum

Vrije Universiteit Amsterdam

27th USENIX Large Installation System Administration Conference

Washington, D.C., USA
November 3-8, 2013

The Update-without-downtime Problem

Solutions

"If you think database patching is onerous,
then try patching a SCADA system
that's running a power plant."

Kelly Jackson Higgins on the SCADA patch problem, 2013

Solution 1: "*Spare time for downtime*"

Solutions

"In one of the biggest computer errors in banking history,
Chemical Bank mistakenly deducted about $15 million
from more than 100,000 customers' accounts."

Saul Hansell, New York Times, 1994

Solution 2: "*Roll your upgrades*"

Solutions

"Our research shows that 75% of successful attacks occur against previously known vulnerabilities for which a remediation was already available."

Neil MacDonald, Gartner Research, 2012

Solution 3: "*Don't patch, don't tell*"

Solutions

"All problems in computer science can be solved by another level of indirection—but that will usually create another problem."

Butler Lampson, quoting David Wheeler

Our solution: "*Live update*"

Live Update in the Real World

ORACLE | Ksplice

Servers protected with Ksplice Uptrack:
100,000+ at more than **700 companies**

Updates applied on production systems:
More than **2 million** and counting

How it works

Your Linux vendor releases an update.

Ksplice converts the upate into a rebootless update.

You install the update seamlessly, without rebooting.

Source: `http://www.ksplice.com`

Are We There Yet?

```
--- a/drivers/md/dm-crypt.c
+++ b/drivers/md/dm-crypt.c
@@ -690,6 +690,8 @@ bad3:
 bad2:
     crypto_free_tfm(tfm);
 bad1:
+    /* Must zero key material before freeing */
+    memset(cc, 0, sizeof(*cc) + cc->key_size * sizeof(u8));
     kfree(cc);
     return -EINVAL;
 }
@@ -706,6 +708,9 @@ static void crypt_dtr(...)
         cc->iv_gen_ops->dtr(cc);
     crypto_free_tfm(cc->tfm);
     dm_put_device(ti, cc->dev);
+
+    /* Must zero key material before freeing */
+    memset(cc, 0, sizeof(*cc) + cc->key_size * sizeof(u8));
     kfree(cc);
 }
```

Linux kernel security patch for CVE-2006-0095

Are We There Yet?

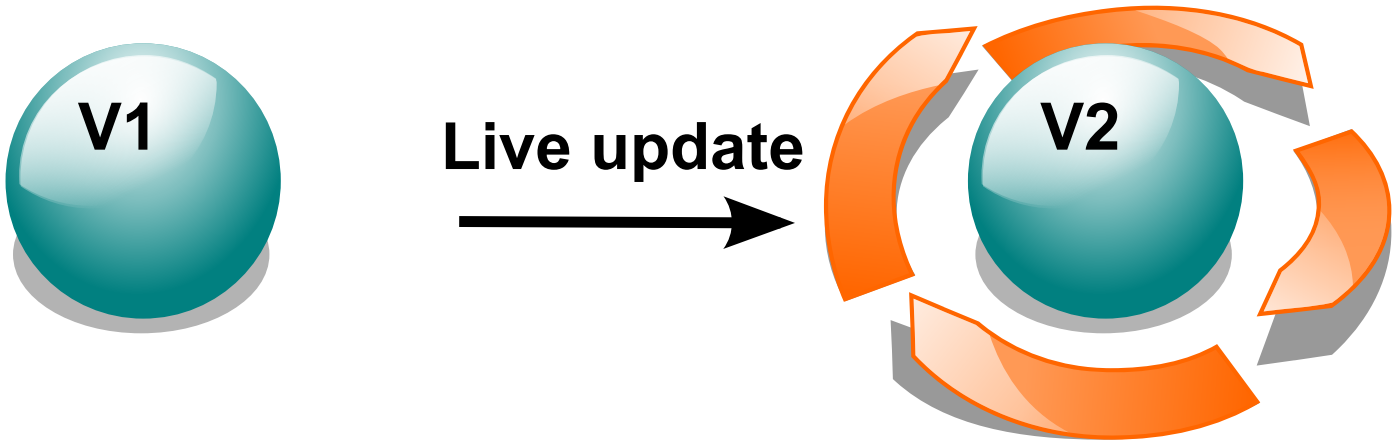
```
--- a/example.c
+++ b/example.c
@@ -1,13 +1,12 @@
 struct s {
     int count;
-    char str[3];
-    short id;
+    int id;
+    char str[2];
     union u u;
-    void *ptr;
     int addr;
-    short *inner_ptr;
+    int *inner_ptr;
 } var;

 void example_init(char *str) {
-    snprintf(var.str, 3, "%s", str);
+    snprintf(var.str, 2, "%s", str);
 }
```

Sample patch with data structure changes

WWW: What We Want

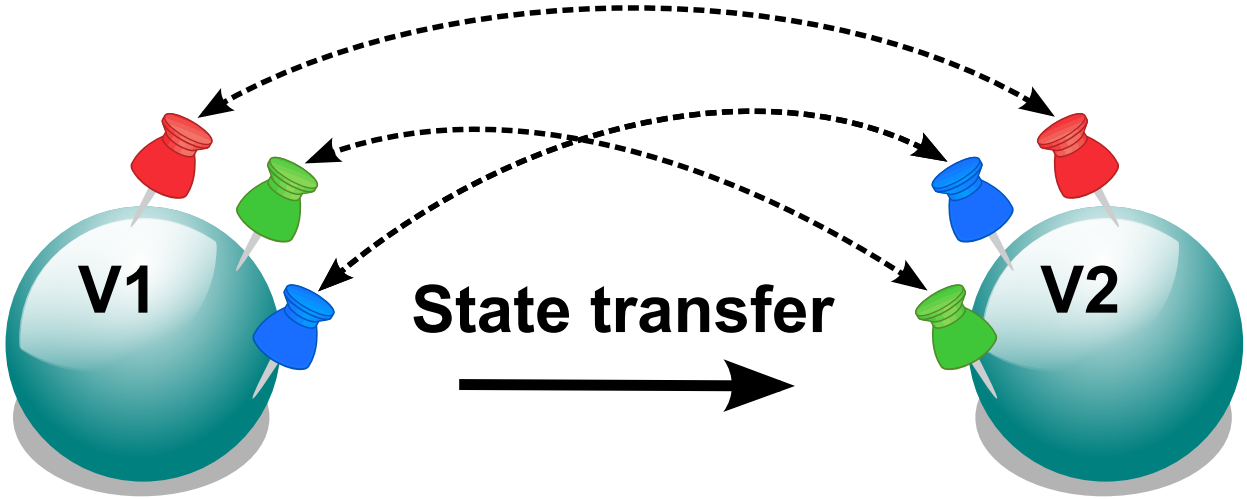
- Support for arbitrarily complex software updates.
- Support for generic C programs found "*in the wild*".
- Automatic state transfer and state validation.
- Automatic detection of run-time and memory errors.
- Automatic error recovery from failed updates (*hot rollback*).

Time-traveling State Transfer Overview



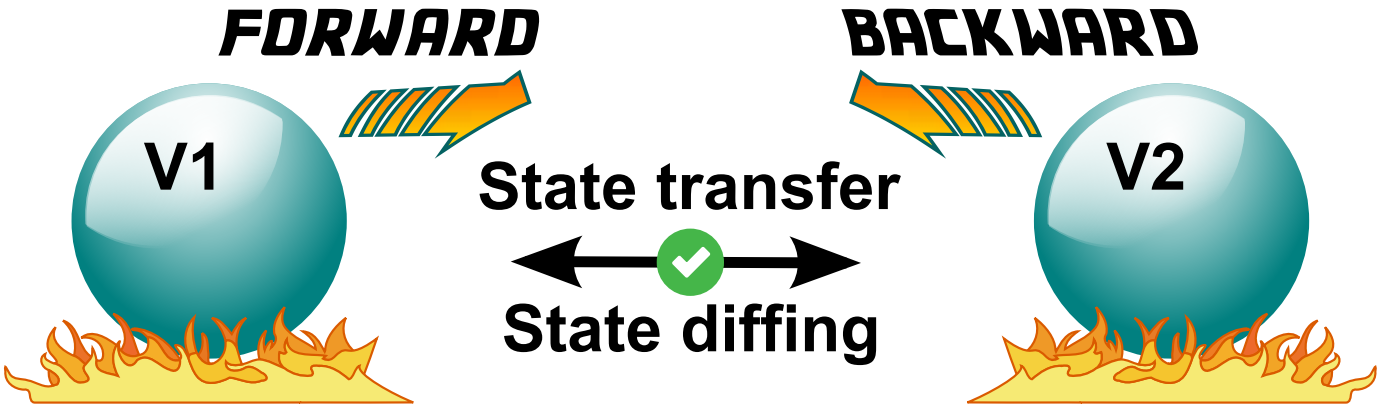
Process-level updates

Time-traveling State Transfer Overview



Automatic state transfer

Time-traveling State Transfer Overview



Automatic state validation

Sysadmin Interface

```
% readlink -f `which my-program`
/path/to/my-program-3.5.24

% ttst-ctl update /path/to/my-program-3.6.10 \
      `pidof my-program`

ttst: Live update requested for my-program.
ttst: Loading /path/to/my-program-3.6.10...
ttst: Applying changes...
ttst: Validating changes...
ttst: Cleaning up old version...
ttst: Live update done.
```

Version update

Sysadmin Interface

```
% readlink -f `which my-program`
/path/to/my-program-3.5.24

% ttst-ctl update /path/to/my-program-3.6.10 \
      `pidof my-program`

ttst: Live update requested for my-program.
ttst: Loading /path/to/my-program-3.6.10...
ttst: Applying changes...
ttst: Validating changes...
ttst: Cleaning up old version...
ttst: Live update done.
```

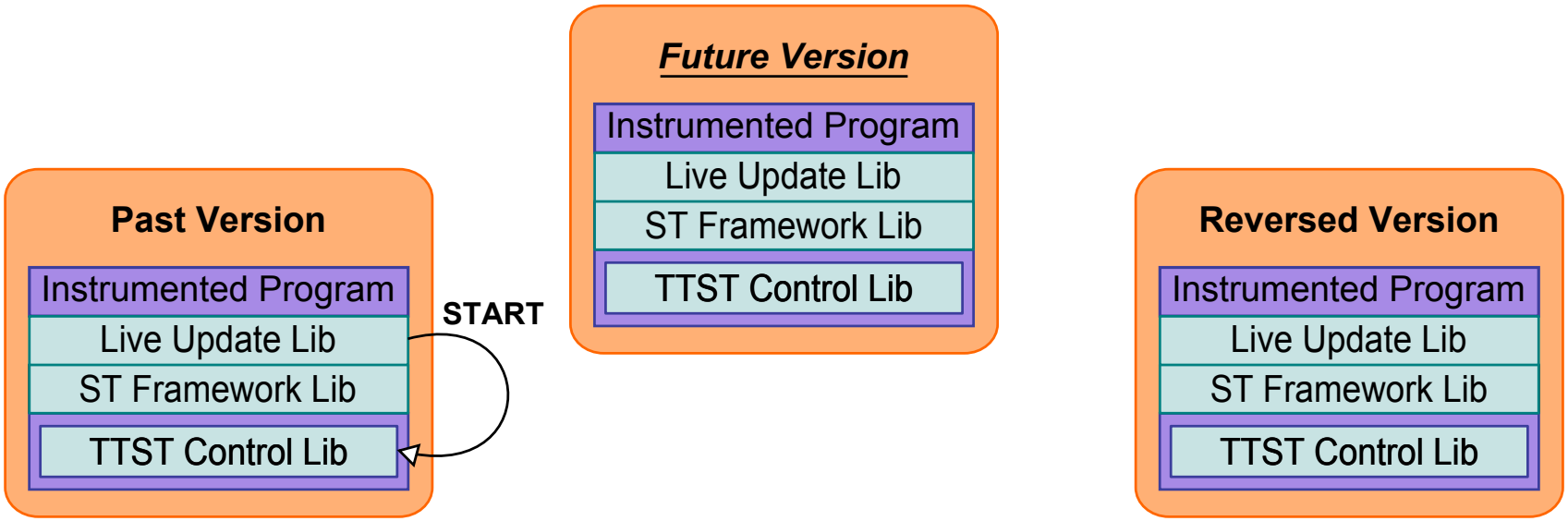
ttst-ctl tool

Sysadmin Interface

```
% readlink -f `which my-program`
/path/to/my-program-3.5.24

% ttst-ctl update /path/to/my-program-3.6.10 \
      `pidof my-program`

ttst: Live update requested for my-program.
ttst: Loading /path/to/my-program-3.6.10...
ttst: Applying changes...
ttst: Validating changes...
ttst: Cleaning up old version...
ttst: Live update done.
```

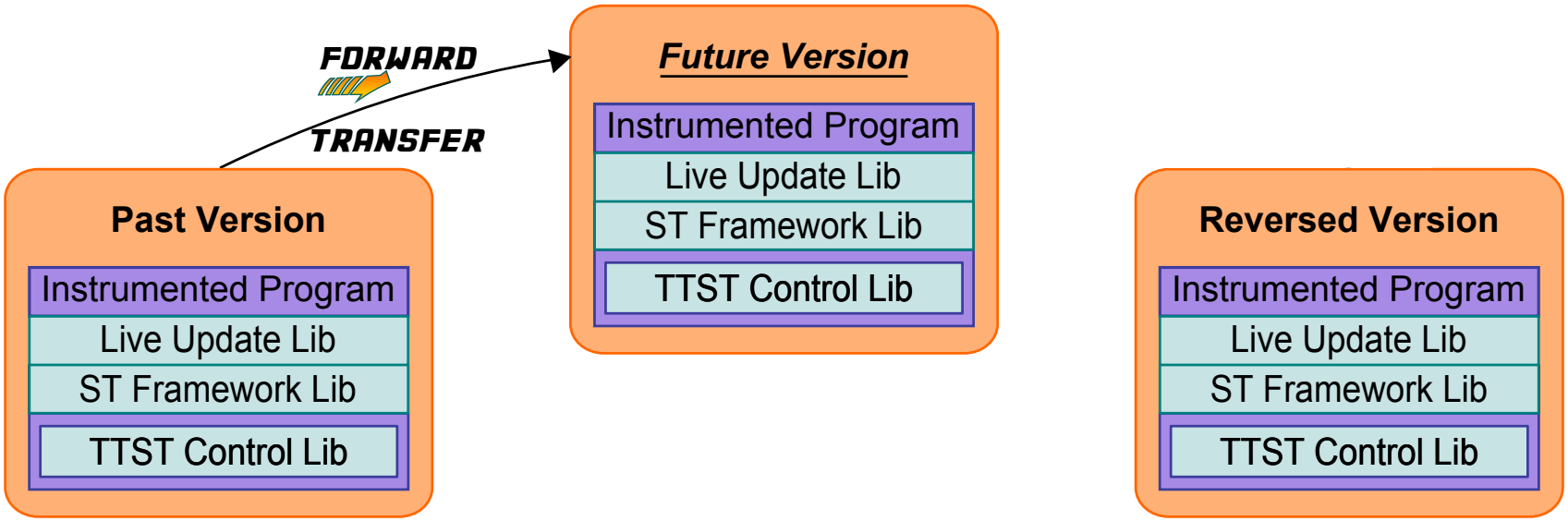
Live update applied automatically

TTST-enabled Live Update



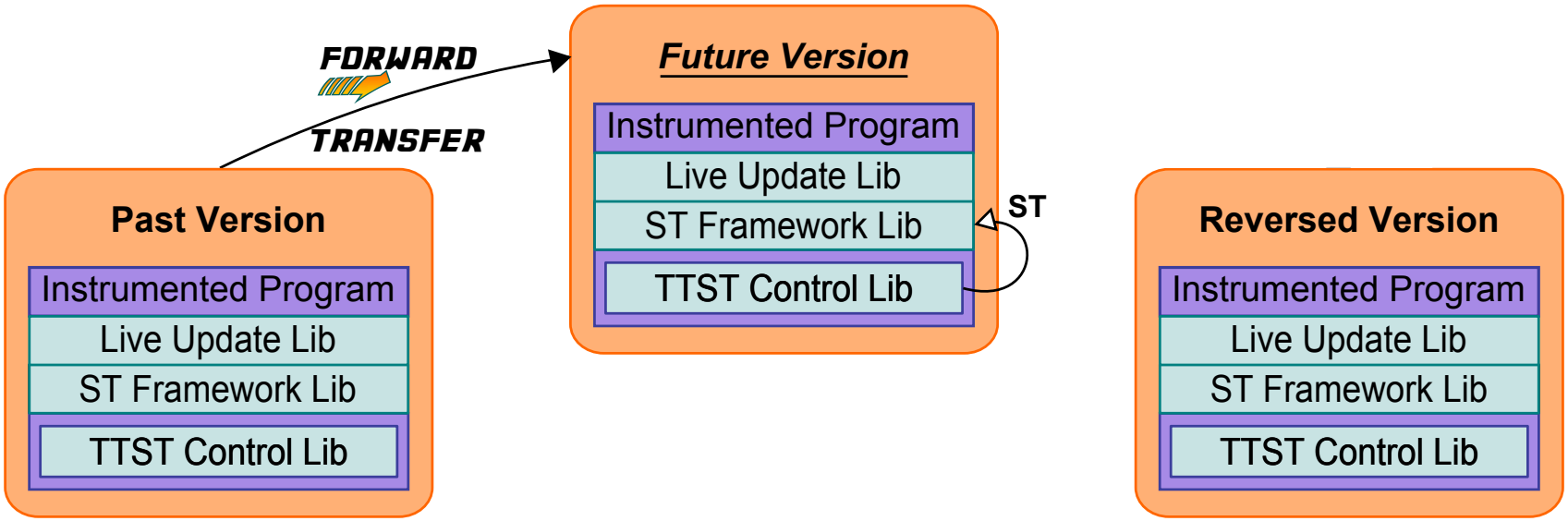
Live update initialization

TTST-enabled Live Update



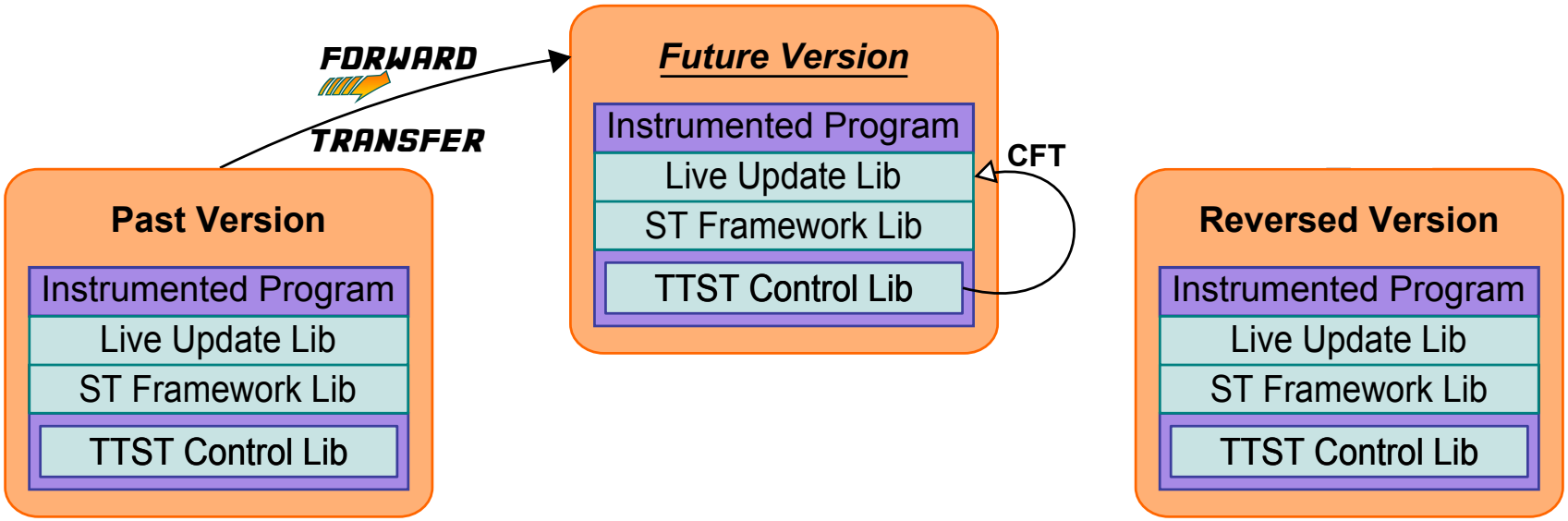
Forward state transfer

TTST-enabled Live Update



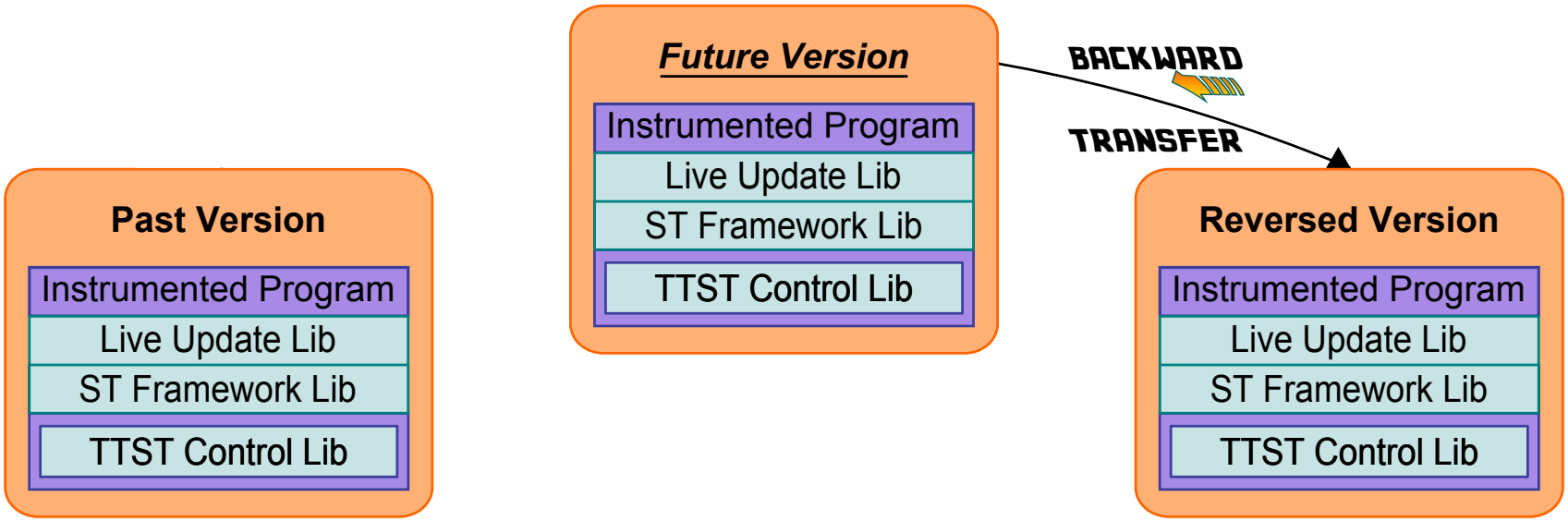
Forward state transfer

TTST-enabled Live Update



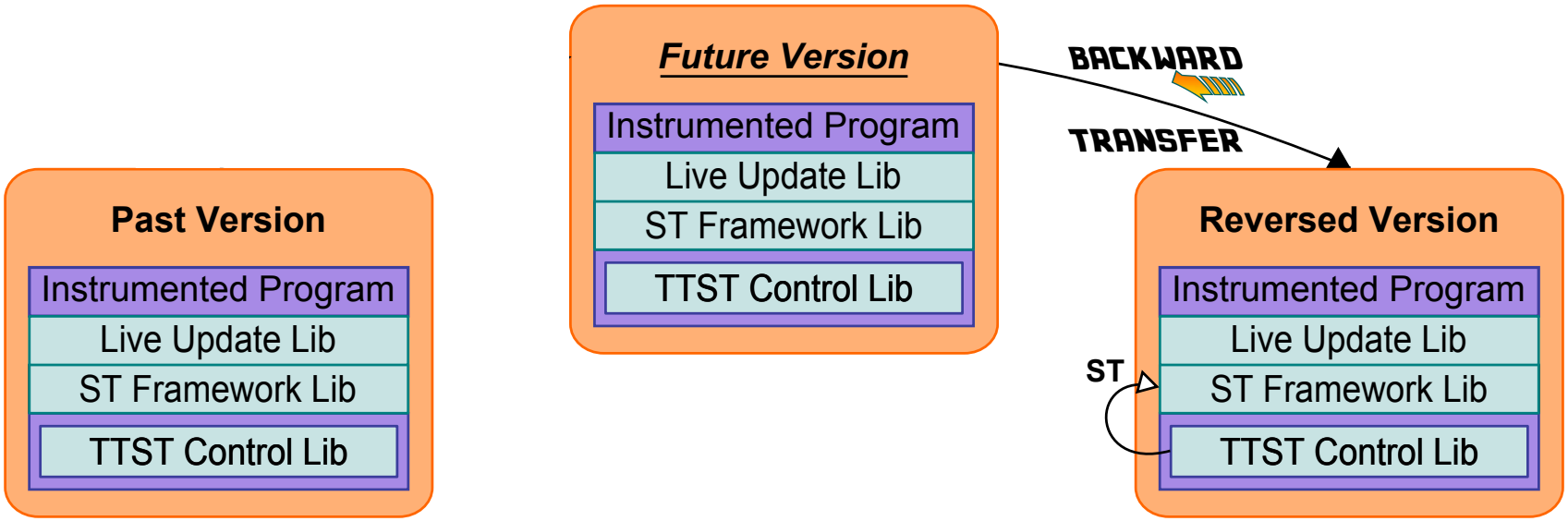
Forward state transfer

TTST-enabled Live Update



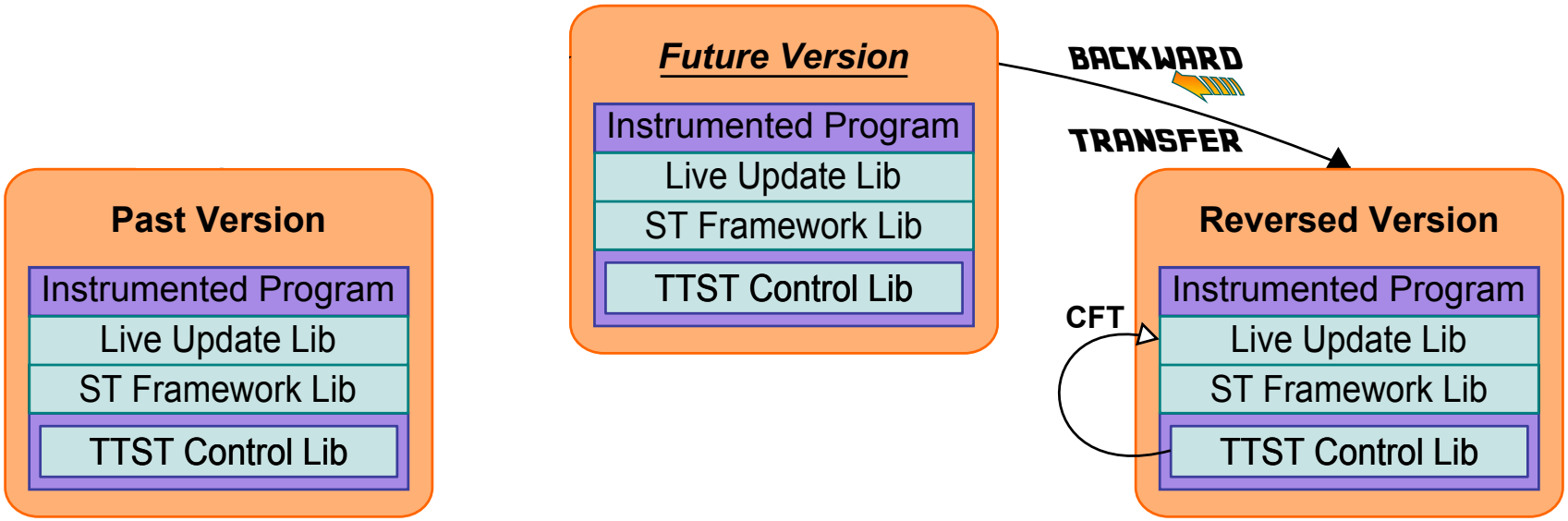
Backward state transfer

TTST-enabled Live Update



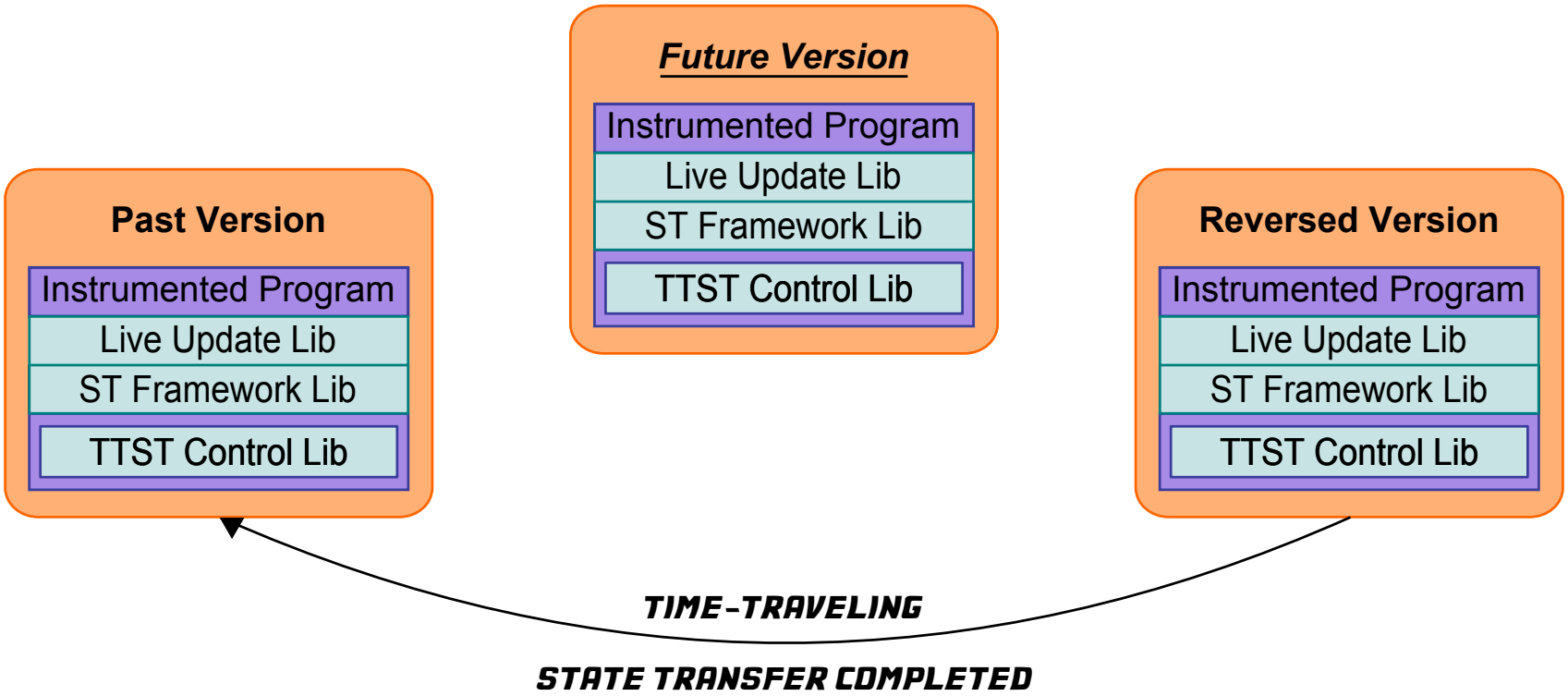
Backward state transfer

TTST-enabled Live Update



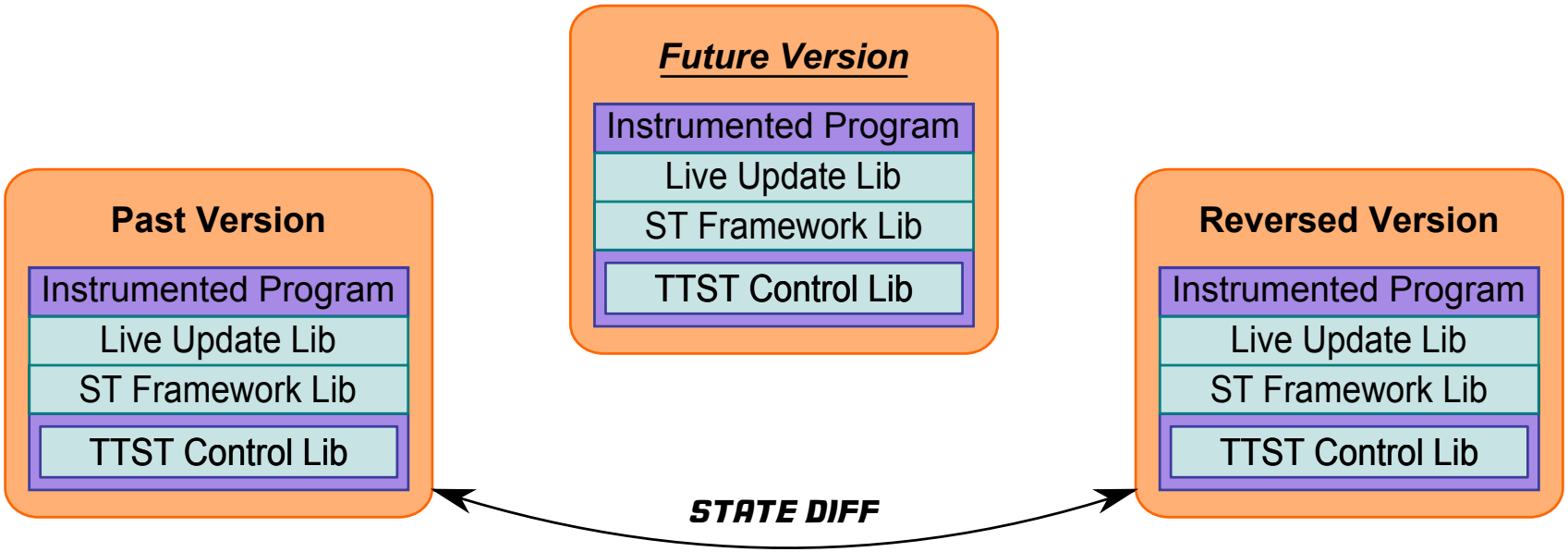
Backward state transfer

TTST-enabled Live Update



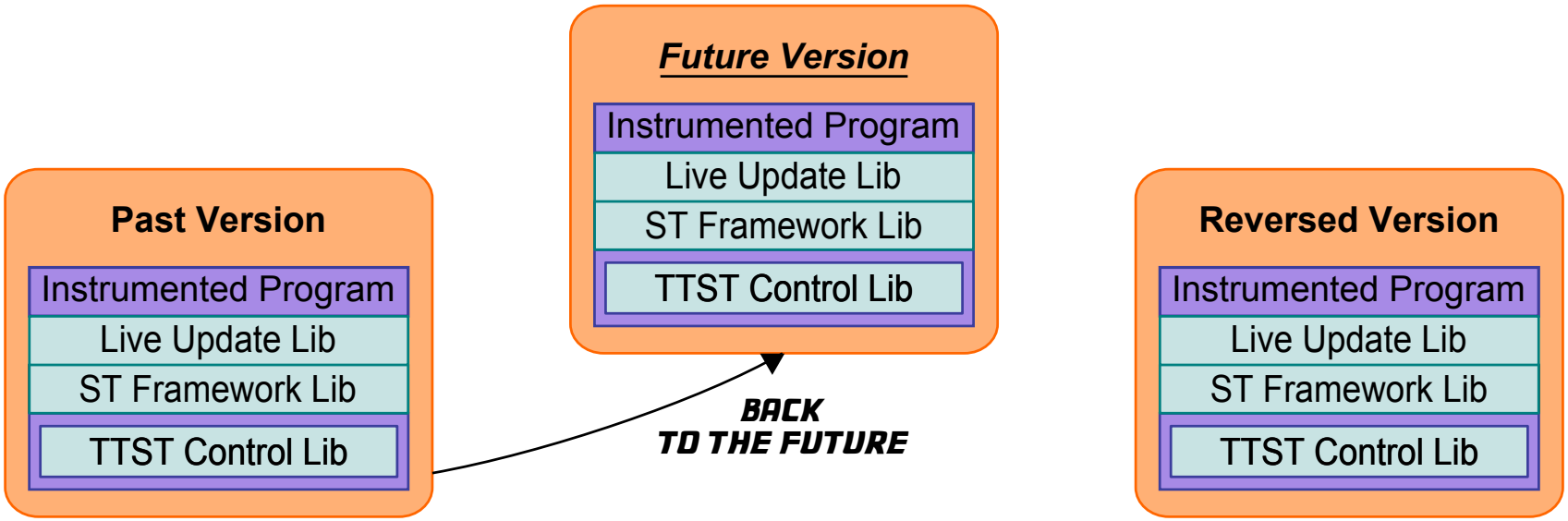
Automatic state validation

TTST-enabled Live Update



Automatic state validation

TTST-enabled Live Update



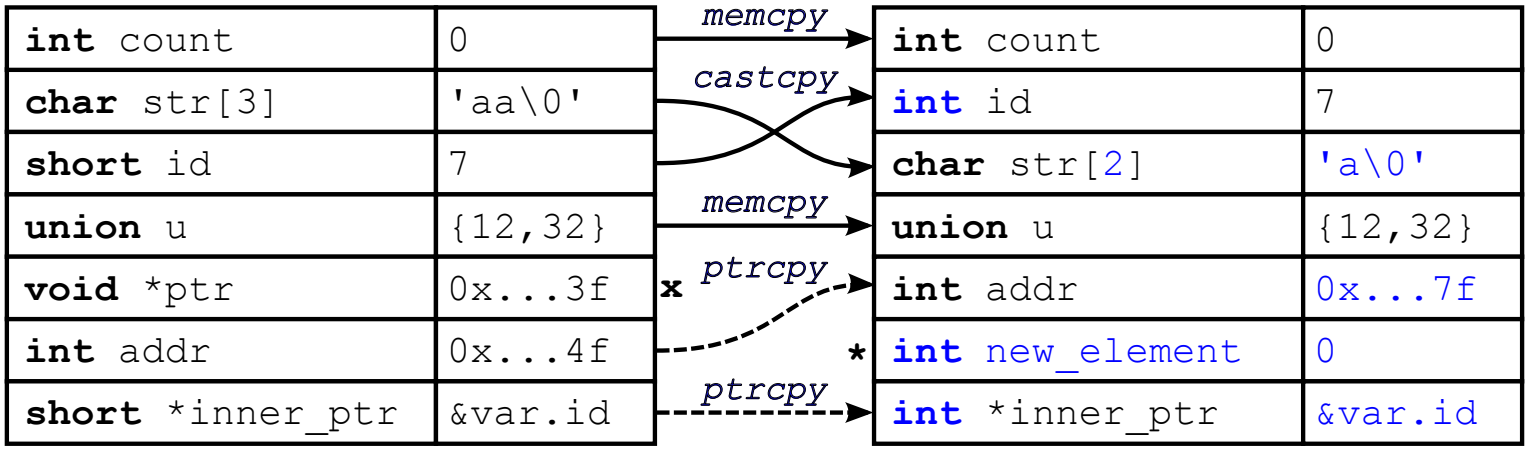
Live update completed

State Transfer Instrumentation

State Transfer Example

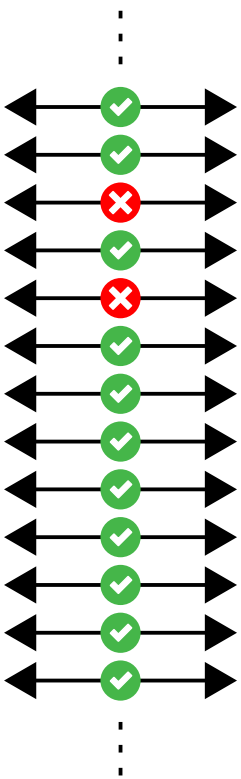
```
struct s { //old version
    int count;
-   char str[3];
-   short id;
    union IXFER(my_u) u;
-   void *ptr;
    PXFER(int) addr;
-   short *inner_ptr;
} var;
```

```
struct s { //new version
    int count;
+   int id;
+   char str[2];
    union IXFER(my_u) u;
    PXFER(int) addr;
+   int new_element;
+   int *inner_ptr;
} var;
```



State Diffing

```
$ cat /proc/$MY_PAST_PID/maps                    $ cat /proc/$MY_REV_PID/maps

08048000-08124000 ... /my-prog                   08048000-08124000 ... /my-prog
08124000-08125000 ... /my-prog                   08124000-08125000 ... /my-prog
08125000-0812a000 ... /my-prog                   08125000-0812a000 ... /my-prog
0812a000-0812f000 ...                            0812a000-0812f000 ...
09f79000-0a583000 ... [heap]                     09f79000-0a583000 ... [heap]
...                                              ...
41000000-41020000 ... mylib.so                   41000000-41020000 ... mylib.so
41020000-41021000 ... mylib.so                   41020000-41021000 ... mylib.so
41021000-41022000 ... mylib.so                   41021000-41022000 ... mylib.so
...                                              ...
b776a000-b776c000 ...                            b776a000-b776c000 ...
b776c000-b776d000 ... [vdso]                     b776c000-b776d000 ... [vdso]
bf868000-bf889000 ... [stack]                    bf868000-bf889000 ... [stack]
```



Sample diff result: 2 memory pages differ

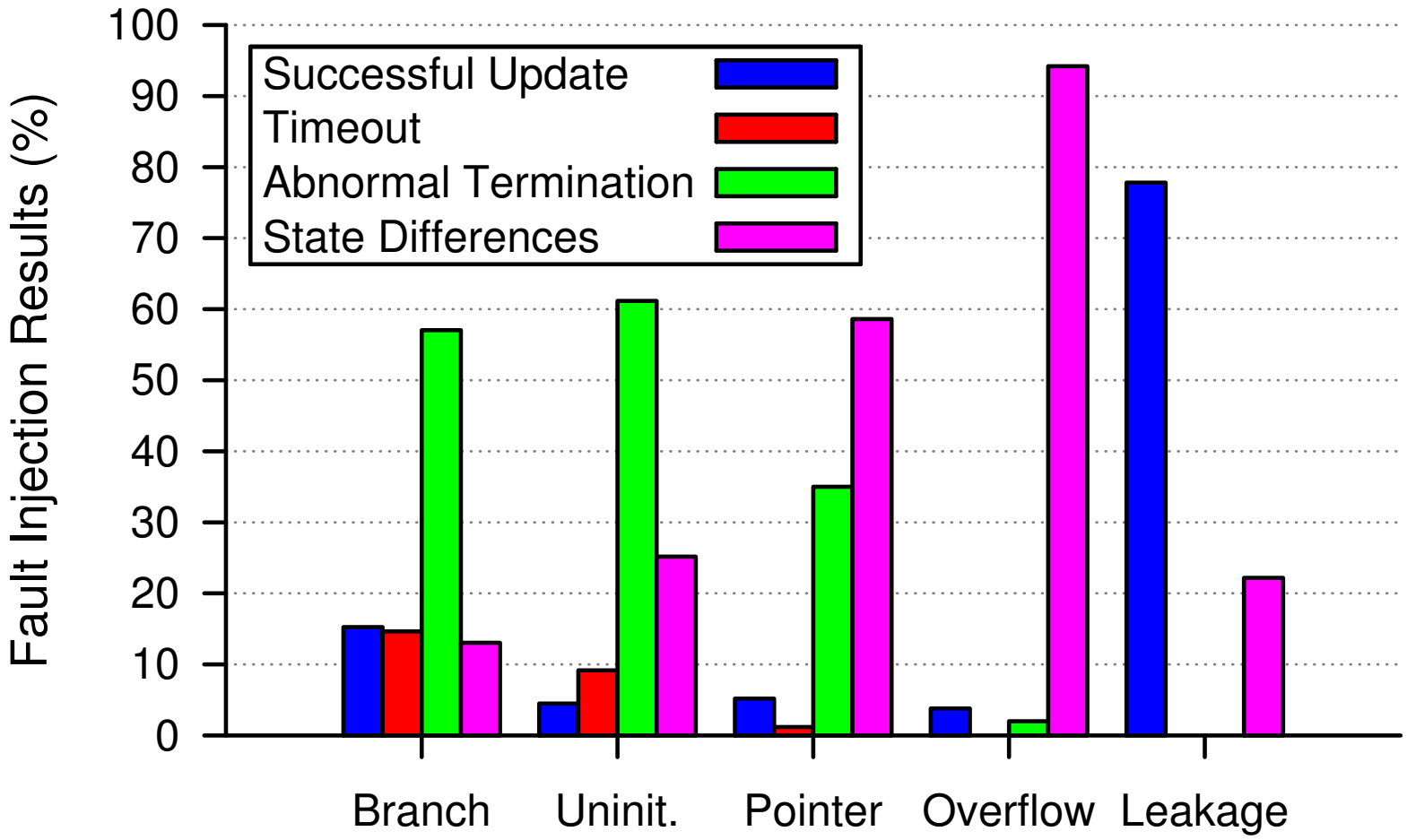
Time-traveling State Transfer Complexity

	Total LOC	Trusted LOC
LLVM plugins		
State transfer instrumentation	9,330	1,119
Support libraries		
Live update library	382	235
TTST control library	3,209	412
State transfer framework	13,311	0
Sysadmin interface		
`ttst-ctl` tool	381	0
Total		
Time-traveling state transfer	26,613	1,766

Experience

- Applied 40 updates (41 KLOC) on **`httpd`**, **`nginx`**, **`vsftpd`**, **`sshd`**.
- Forward state transfer required 896 LOC overall.
- Full-coverage backward state transfer required 299 extra LOC.
- TTST yields negligible run-time overhead.
- TTST yields modest virtual memory overhead ($2.6 - 4.3x$).
- TTST yields modest live update time ($0.1 - 1.2s$).

Fault Injection



Summary

- TTST: A new fault-tolerant live update technique.
- Supports several classes of updates with minimal manual effort.
- Automates state transfer and state validation.
- Detects and recovers from arbitrary run-time and memory errors.
- Relies on a minimal amount of trusted code.

Demo

Back to the Future: Fault-tolerant Live Update with Time-traveling State Transfer

Cristiano Giuffrida, Călin Iorgulescu, Anton Kuijsten, Andy Tanenbaum
{giuffrida,calin.iorgulescu,kuijsten,ast}@cs.vu.nl

Vrije Universiteit Amsterdam