



TSDuck

An extensible toolbox for MPEG transport streams

Version 3.29

Topics

- TSDuck overview
- Transport stream processor
- PSI/SI table manipulation using XML or JSON
- Extending TSDuck
- TSDuck as an MPEG/DVB C++ library
- Using TSDuck from Java and Python



TSDuck overview

- Process ISO/IEC 13818-1 transport streams
- Set of low-level utilities
 - extensible through plugins
- « Batch & Bash » oriented
 - command-line only, no fancy GUI
 - one utility or plugin = one elementary function
 - can be combined in any order
- Written in C++
 - reusable and extensible code
 - Java and Python bindings
- Available on Linux, Windows and macOS



What TSDuck is / is not

4

TSDuck is a general-purpose toolbox
for digital TV engineers

→ lab, demo, test, integration, debug, end-to-end testing ←

TSDuck is not an off-the-shelf ready-to-use specialized
application for production and operations



Sample usages (1/2)

- TS acquisition (IP, HTTP, HLS, SRT, RIST, DVB, ATSC, ISDB, ASI)
- TS analysis
- Transmodulation
- Analysis, edition, injection of PSI/SI
 - using and editing PSI/SI in XML or JSON format
- Service manipulation
 - extract, remove, rename, etc.
- SCTE 35 splicing injection, extraction and monitoring
- MPE injection and extraction (Multi-Protocol Encapsulation)



Sample usages (2/2)

- Route, merge, fork TS between applications
- Test bed for CAS or STB
 - injection of test cases
 - DVB or ATIS Scrambling with DVB SimulCrypt support
- Extraction of specific streams
 - T2-MI (DVB-T2 Modulator Interface)
 - PLP's (Physical Layer Pipe)
 - Teletext subtitles
- Any combination of the above and more...



Availability

- Web site
 - <https://tsduck.io/>
- Open-source code
 - <https://github.com/tsduck/tsduck>
- BSD license
 - liberal, no GPL-like contamination
 - can be used in all applications



- Installation

- pre-built binary installers for Windows and Linux
 - Fedora, CentOS, Ubuntu, Debian, Raspbian
 - user's contribution: Arch Linux (AUR)
- through Homebrew on macOS
 - user's contribution: MacPorts

- Documentation

- user's guide
 - PDF, 500+ pages of references and examples
- programmer's guide
 - doxygen-generated, online on tsduck.io
 - for C++, Java and Python developers



tsp

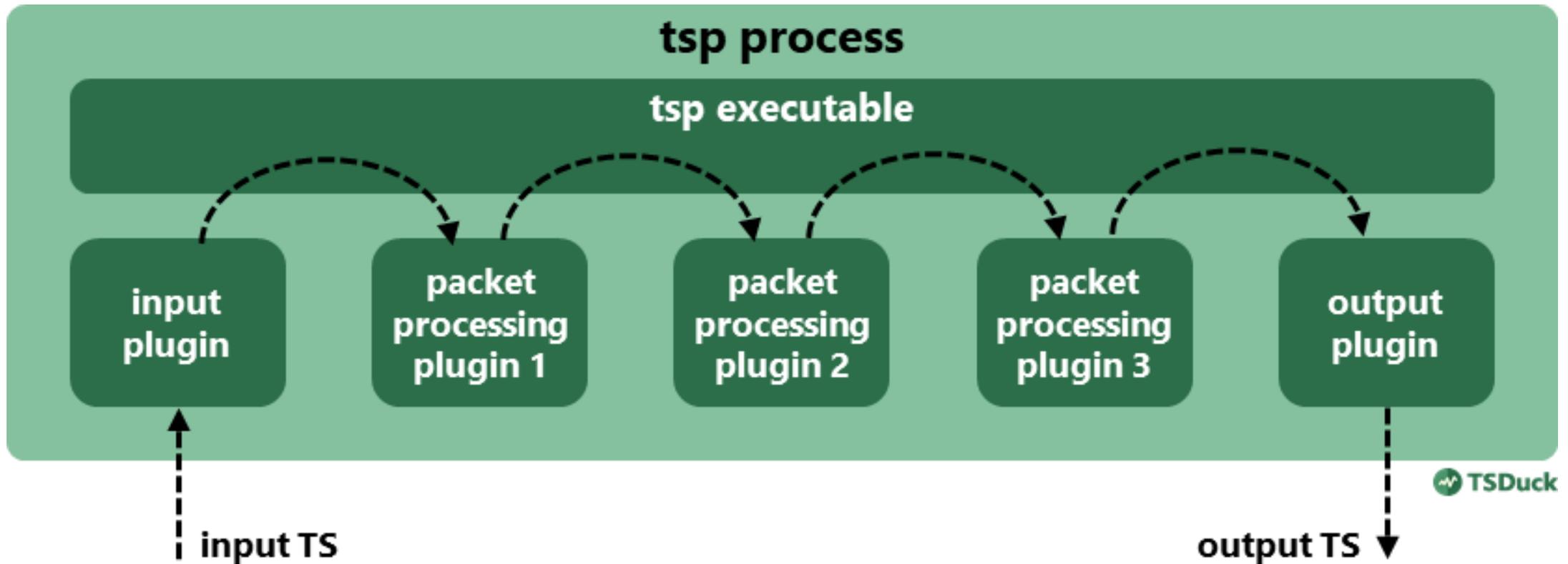
the transport stream processor



- Transport stream processing framework
 - Combination of elementary processing using plugins
 - One input plugin
 - receive a TS from various sources
 - Any number of packet processing plugins
 - perform transformations on TS packets
 - may remove packets
 - may NOT add packets
 - One output plugin
 - send the resulting TS to various destinations



tsp plugin chain



tsp plugins

- Each tsp plugin is a shareable library
 - .so (Linux), .dylib (macOS), .dll (Windows)
 - plus some embedded plugins, not seen as shareable libraries
- File naming
 - plugin named *foo* in file *tsplugin_foo.so* (or .dylib or .dll)
- General command line syntax

```
tsp [tsp-options]  
  -I input-name [input-options]  
  -P processor-name [processor-options]  
  ...  
  -O output-name [output-options]
```



tsp basic syntax

- TS acquisition

```
tsp -I dvb --uhf 21  
-P until --seconds 20  
-O file capture.ts
```

capture DVB stream from UHF channel 21

pass packets during 20 seconds, then stop

save packets to file capture.ts

- Display the PMT of a selected service

```
tsp -I dvb --uhf 35  
-P zap france2  
-P sifilter --pmt  
-P tables --max 1  
-O drop
```

extract service « France 2 », rebuild SPTS

extract PID containing the PMT

display one table, then stop

drop output packets



Simple examples

- Transmodulation of a service over IP multicast

```
tsp -I dvb --uhf 35  
-P zap france2 --audio fra  
-O ip 224.10.11.12:1000
```

extract service « France 2 »,
keeping only one audio track

broadcast resulting SPTS to
multicast IP address:port

- On-the-fly replacement of a PSI / SI table

```
tsp -I dvb --uhf 24  
-P inject nit.xml --pid 16 --replace --stuffing  
-O dektec --uhf 24 --convolution 2/3 --guard 1/3
```

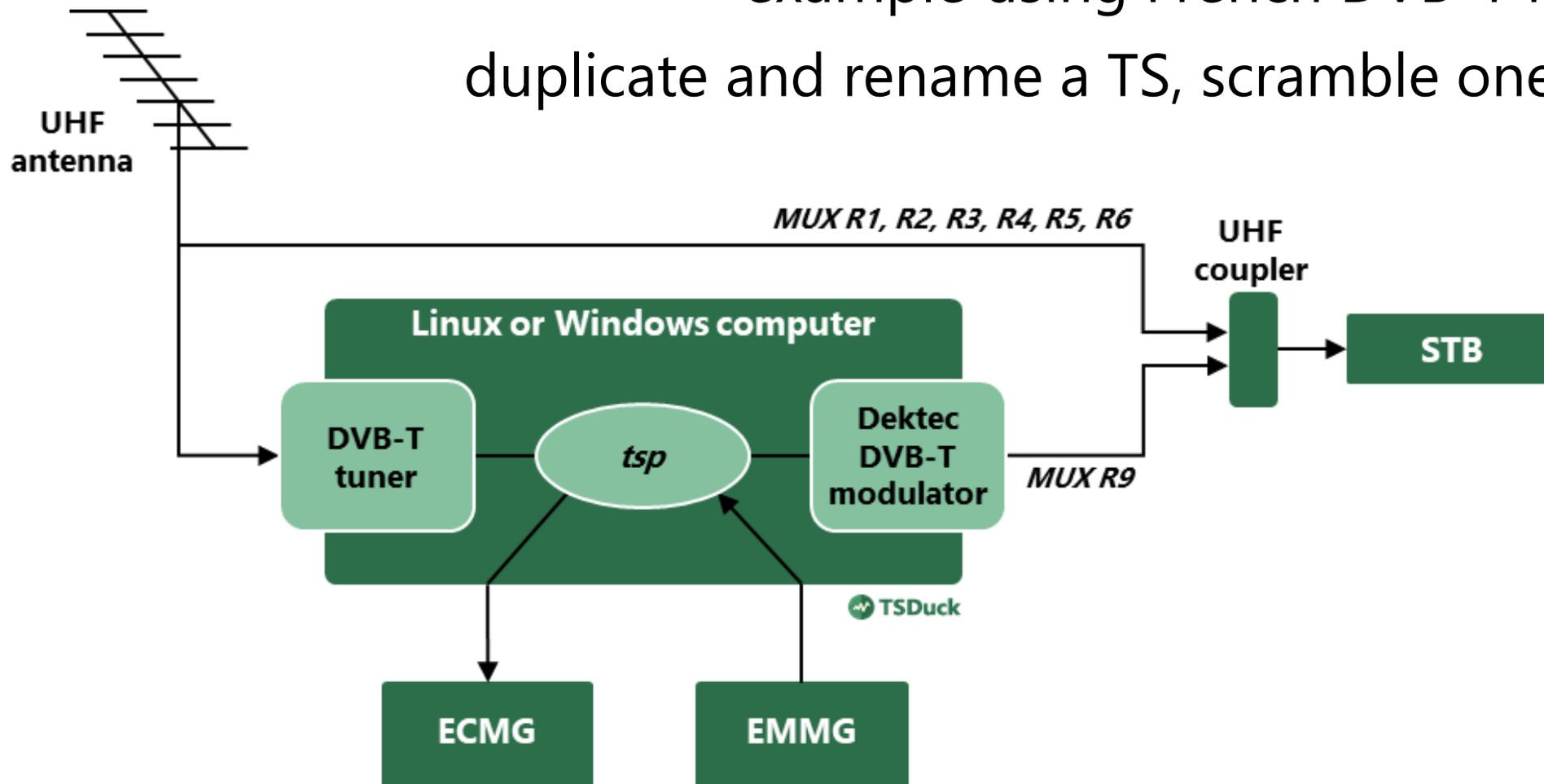
replace content of PID 16 with
table from an XML file (a NIT)

send modified TS to a Dektec
modulator on same frequency



Sample CAS test bed (1/2)

example using French DVB-T network:
duplicate and rename a TS, scramble one service



Sample CAS test bed (2/2)

16

```
tsp -I dvb -u $UHF_INPUT
-P tsrename -t 9 -a
-P svrename direct8 -i 0x0901 -l 41 -n "Direct 8 Test"
-P svrename bfmtv -i 0x0903 -l 42 -n "BFM TV Test"
-P svrename 'i>tele' -i 0x0904 -l 43 -n "i>TELE Test"
-P svrename virgin17 -i 0x0905 -l 44 -n "Virgin 17 Test"
-P svrename gulli -i 0x0906 -l 45 -n "Gulli Test"
-P svrename france4 -i 0x0907 -l 46 -n "France 4 Test"
-P svrename 0x02FF -i 0x09FF
-P scrambler GulliTest -e $ECMG -s $SUPER_CAS_ID
    -p $PMT_CADESC_PRIVATE -a $AC
    -b $ECM_BITRATE --pid $ECM_PID
-P cat -c -a $CAS_ID/$EMM_PID/$CAT_CADESC_PRIVATE
-P datainject -r -s $MUX_SERVER_PORT
    -b $EMM_MAX_BITRATE -p $EMM_PID
-O dektec -u $UHF_OUTPUT --convolution 2/3 --guard 1/32
```

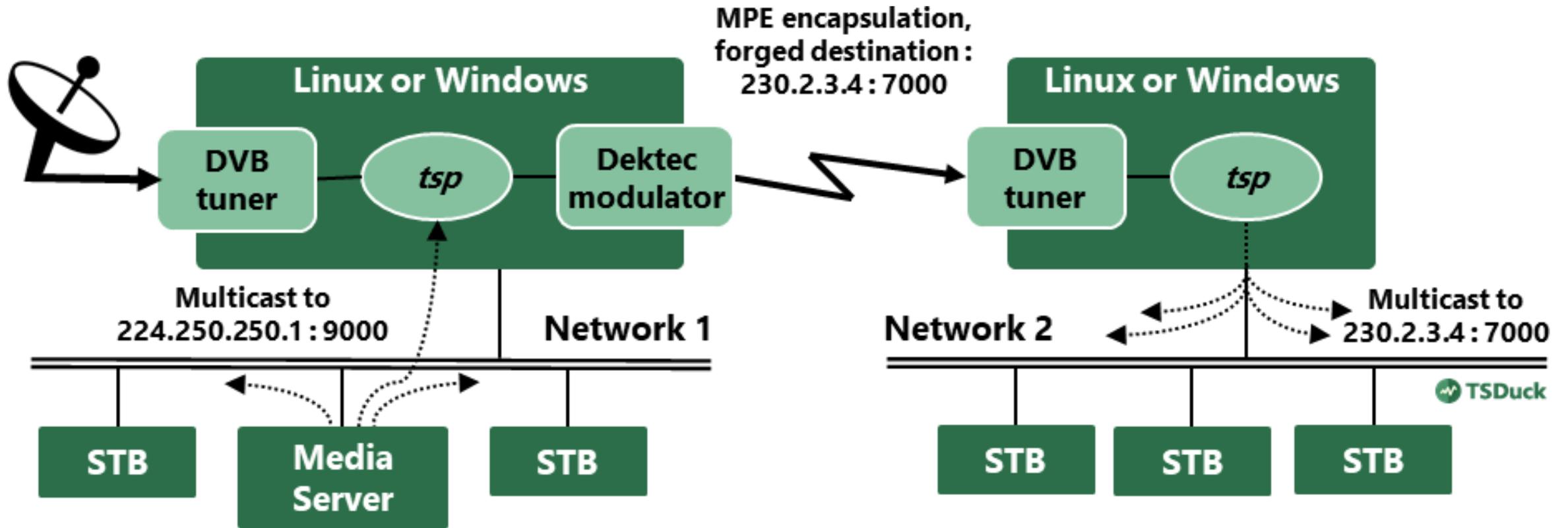
rename the TS,
rename all services in the TS
(service name, service id, LCN)

scramble one service,
connect to a real ECM Generator,
update PMT, insert ECM's

accept connection from
a real EMM Generator,
update the CAT, insert EMM's

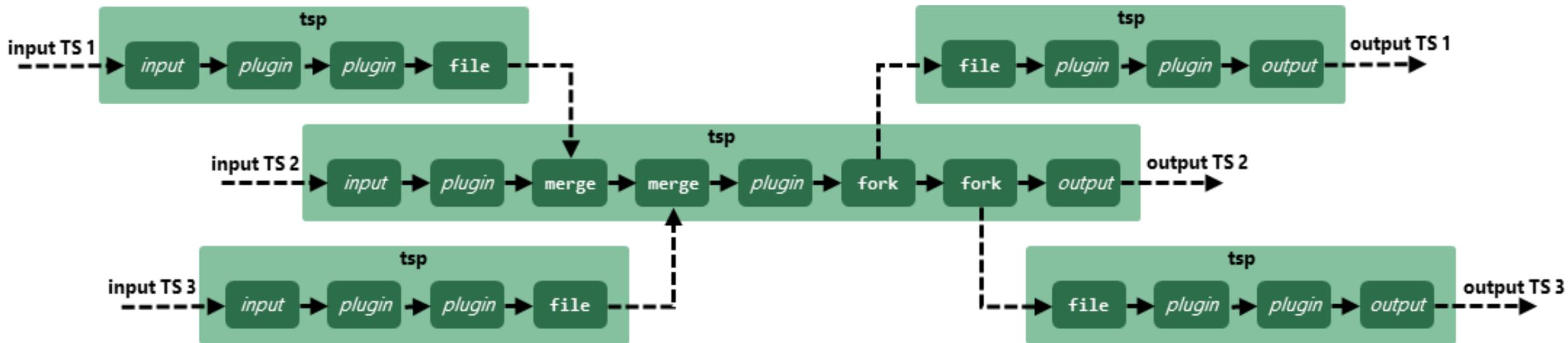


Sample MPE injection and extraction

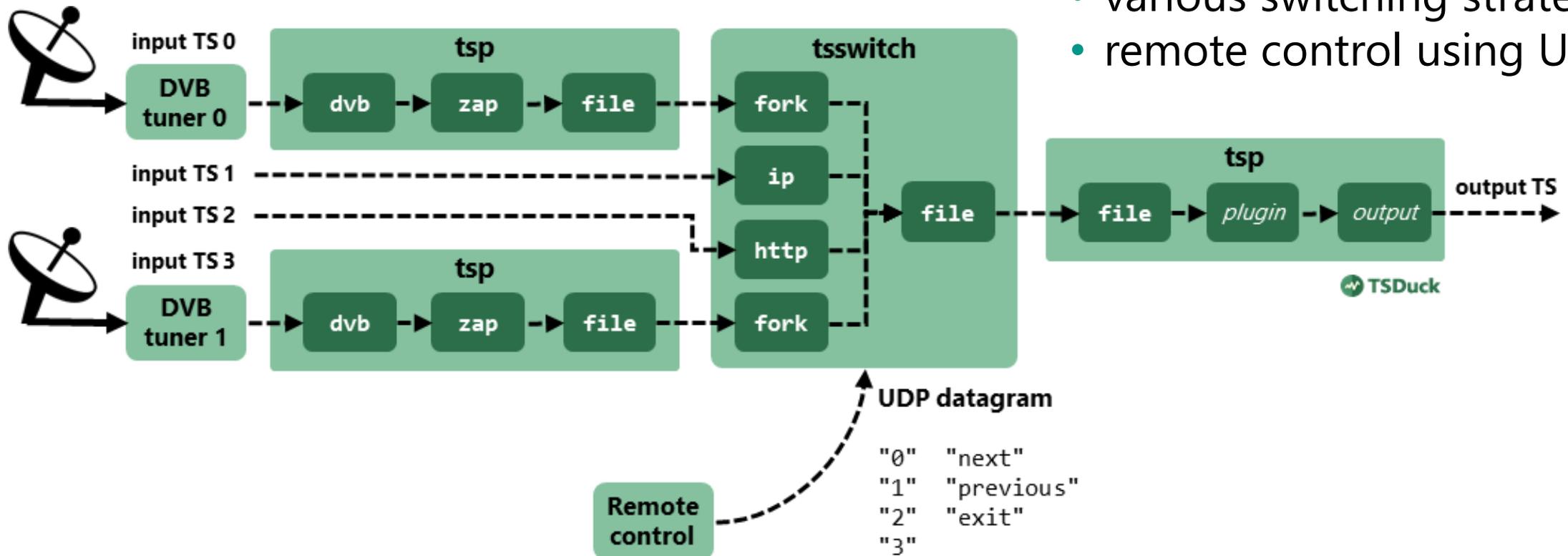


Multiple tsp instances

- fork : duplicate the TS to another application
- merge : merge with a TS coming from another application
merge service references (PAT, CAT, etc.)



Switching between multiple inputs



- tsswitch command
- same plugins as tsp
- various switching strategies
- remote control using UDP



Input and output plugins

- Input plugins
 - Files: TS, M2TS, PCAP
 - Network: multicast IP, HLS, HTTP, SRT, RIST
 - Hardware: tuners (DVB, ATSC, ISDB), Dektec (ASI, demodulators)
 - Application: fork, memory, craft, null
- Output plugins
 - Files: TS, M2TS
 - Network: multicast IP, HLS, SRT, RIST
 - Hardware: Dektec (ASI, modulators), HiDes (modulators)
 - Application: fork, memory, drop, media player



Packet processing plugins

- TS transformations
 - PID or packet filtering, PSI/SI transformation or injection, service extraction or modification, etc.
- TS regulation
 - time regulation, time shifting, scheduled recording, etc.
- TS analysis and monitoring
 - TS analysis, PSI/SI extraction, PID, bitrate monitoring, ECM or EMM monitoring, etc.
- TS scrambling & descrambling
 - DVB SimulCrypt support for ECM / EMM injection
- Data injection or extraction
 - SCTE 35, T2-MI, MPE, Teletext
- Any other processing you wish to develop...
 - 75 packet processing plugins available (version 3.29)



PSI/SI tables manipulation

binary, XML or JSON



MPEG tables and sections

- Extraction from TS, injection into TS
- Data formats
 - binary
 - raw sections
 - XML
 - fully documented in user's guide
 - easy to manually edit or process in applications
 - JSON
 - through automated XML-to-JSON conversion
 - easy to process in applications, especially in Python
- All formats are uniformly used
 - TS extraction, modification, injection
 - file manipulation



Sample XML file

```
<?xml version="1.0" encoding="UTF-8"?>
<tsduck>
  <PAT version="8" transport_stream_id="0x0012" network_PID="0x0010">
    <service service_id="0x0001" program_map_PID="0x1234"/>
    <service service_id="0x0002" program_map_PID="0x0678"/>
  </PAT>
  <PMT version="4" service_id="0x0456" PCR_PID="0x1234">
    <CA_descriptor CA_system_id="0x0777" CA_PID="0x0251"/>
    <component elementary_PID="0x0567" stream_type="0x12">
      <ISO_639_language_descriptor>
        <language code="fre" audio_type="0x45"/>
        <language code="deu" audio_type="0x78"/>
      </ISO_639_language_descriptor>
    </component>
  </PMT>
</tsduck>
```



Multiple ways to update tables in a TS

- Specialized plugins with predefined options
 - BAT, CAT, NIT, PAT, PMT, SDT
- Manual XML handling
 - Extract the table as an XML file
 - Edit the file
 - Reinject the file
- Automated XML modification
 - Using « XML patch files »
 - Flexible XML templates to update tables on the fly
 - Similar to XSLT in principle, but much simpler



Extending TSDuck

C++ transport stream programming



Extending TSDuck

- TSDuck is extensible
 - Source code provided

```
git clone https://github.com/tsduck/tsduck.git
```
 - Common API for Linux, Windows and macOS
 - DVB tuners, Dektec and HiDes devices are not supported on macOS
 - Programmer's guide
 - doxygen-generated, see <https://tsduck.io/>
- You can modify it yourself !



Why extending TSDuck?

- Identify your needs
- Try to find a solution using existing TSDuck plugins
 - review utilities and plugins
- Try to extend an existing utility or plugin
 - add new options
 - add features, don't modify existing behavior
 - remain upward compatible
- Develop your own plugin
 - it is quite simple, really
- Send your code back using a pull request
 - so that everyone can benefit from it



- Don't write a plugin from scratch
 - use an existing one as code base
 - choose one which is technically similar
 - input? output? PSI/SI transformation? packet filtering?
- Implement simple & elementary features
 - preserve TSDuck philosophy
 - develop several elementary plugins if necessary
 - not a single big plugin implementing several features
- Read the « TSDuck coding guidelines » document
 - a recommended reading, although not required



Proprietary extensions

- Reason: manipulation of confidential data
 - private PSI/SI signalization
 - conditional access system implementation
- Completely independent source code and binary packages
 - additional plugins
 - additional commands
 - dynamically merging inside TSDuck proprietary implementations of tables, descriptors, CAS data structures, naming, etc.
- Compatible with TSDuck BSD license
- Sample code in source code tree
 - `sample/sample-extension`



Using the TSDuck library

to develop third-party applications
in C++, Java or Python



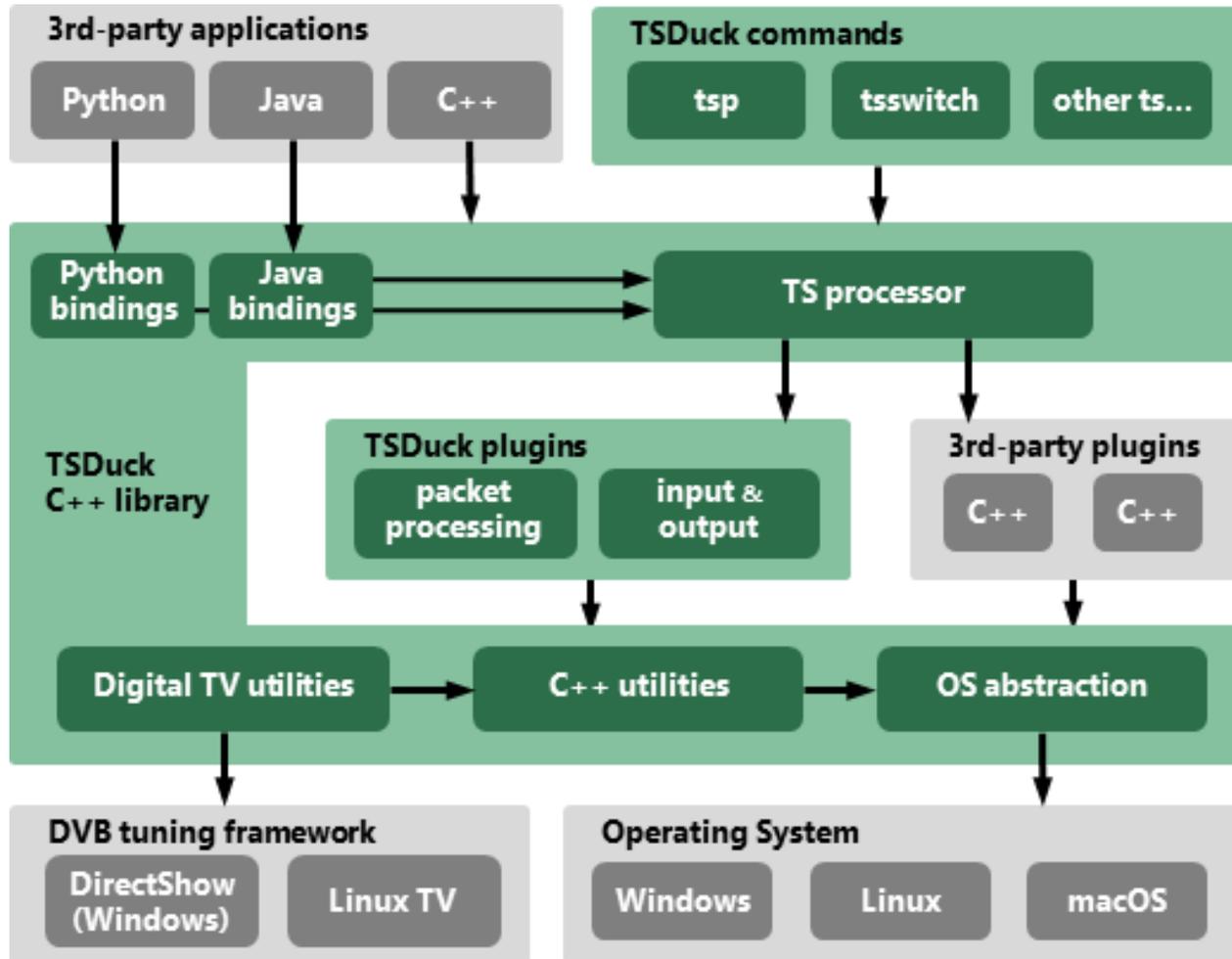
TSDuck library

- All TSDuck common code is in one large library
 - libtsduck.so / libtsduck.dylib / tsduck.dll
- Contains generic and reusable C++ code
 - basic operating system independent features
system, multi-treading, synchronization, networking, cryptography, etc.
 - MPEG/DVB features
TS packets, PSI/SI tables, sections and descriptors, demultiplexing, packetization, encapsulation, DVB tuners, etc.
- Can be used in your application
 - even if not part of TSDuck



Software architecture

33



- C++ applications
all TSDuck features
- Java or Python applications
high-level features only
interactions using JSON or XML
memory buffers for input/output
- Custom plugins
 - C++ only



Programmer's guide

- Online, see <https://tsduck.io/doxy/>
 - automatically updated every night
- Tutorials
 - building TSDuck and applications using its library
 - developing TSDuck plugins and extensions
 - C++ library tutorial
 - Java and Python bindings
- Reference
 - all C++, Java and Python classes





Thank you

Any question?