VTP: VDIF Transport Protocol

Release 0.9.7

Chris Phillips, Alan Whitney, Mamoru Sekido & Mark Kettenis

18 October 2012

DRAFT

Introduction

The VLBI Data Interchange Format (VDIF) is a frame based format suitable for real-time eVLBI as well as disk recording. The VDIF specification was formally accepted at the 8th International e-VLBI workshop in Madrid June 2009 and most new VLBI acquisitions systems are being designed to conform to the standard.

VDIF just describes contents of the frames. VTP (VDIF Transport Protocol) is the definition of how VDIF frames should be streamed over standard networks such as Ethernet.

VDIF Summary

The full VDIF specification is available at http://www.vlbi.org/vdif. In summary it is a frame-based format with a 32-byte header followed by raw (baseband) sampled data. The header contains the sample time of the first data bit plus rudimentary description of the data (number of bits/sample, number of parallel recording channels etc.). VDIF supports a large number of options, such as complex sampling and the option of multiple parallel data streams (ie sending each data channel over a separate data stream rather than packing multiple channels within a single frame). VDIF does not guarantee in-order frames. Any software/hardware etc. receiving VDIF data should be able to cope with out-of-order or missing frames.

Definitions

DAS	Data Acquisition System (samplers, filters etc. plus some way of sending the data, not necessarily via VTP). Examples would be a dBBC.
Recorder	Something that receives baseband data and writes it to disk (e.g. Mark5c).
Source	Something that sends VDIF/VTP data over the network - could be either a DAS or a recorder.
Sink	Something that receives VDIF/VTP data - could be a recorder or correlator interface.

Transport Options

Initially 2 standard transport options will be defined for VDIF: reliable streams (TCP) and unreliable packet streams (UDP and raw Ethernet). Other transport protocols (e.g. STCP, UDT) could be used if desired. This document will be updated as required when new transport options are utilised.

VTP/TCP

TCP/IP is a reliable network stream and as such as requires no extra data inserted before or after the frames. The VDIF frames should just be sent as part of the TCP data stream unmodified. If explicitly required, VDIF frame size should be sent to the receiver using some parallel communication (i.e. not via the same stream as the VDIF frames). The exact means of passing this extra information is out-of-scope of this document. Other reliable data streams (e.g. UDT over UDP or SCTP) should adopt a similar approach. The TCP receive ports should follow the following

1. The sink acts as a TCP "server" listening for a connection.
2. The default port should usually be 52030, though the source and sink can negotiate a different port(s) to use as necessary. The negotiation process is out of scope for the VTP definition.
3. When receiving multiple streams onto a single sink, the sink can either listen on a single receive port or listen on multiple ports. Normally the ports used should start from 52030.

VTP/UDP

UDP is an unreliable packet streams and as such needs a way of determining packet drops or reordering. While the VDIF headers contain enough information to unambiguously reorder frames and detect missing data, it is important to distinguish packets dropped by the network from those dropped at the sending side (e.g. in burst mode or when the source deliberately drops frames to reduce the data rate).

A VTP/UDP flow has the following requirements

1. Each UDP datagram consists of a 64-bit unsigned integer sequence number followed by a single VDIF data frame. See Figure 1.
2. The sequence number should be in little-endian byte order.
3. It is recommend that the sequence number should start at 0, but this is not mandatory nor should the sink assume the sequence number starts at 0.
4. The sequence number should be incremented for each frame sent "over-the-wire" and not incremented arbitrarily in time (i.e. there is no guarantee of a one-to-one mapping of the VDIF time to the sequence number).
5. There should always be a single VDIF frame per UDP datagram. VDIF frames should not span multiple datagrams nor should a datagram contain multiple VDIF frames.
6. The sink may optionally reorder the received VDIF frames based on the sequence number.

7. When a sink receives VDIF data streams from multiple sources, each stream needs to be sent to a unique destination port on the sink. The sequence number order should be unique for each stream.
8. The stream port number should be negotiated between source and sink. Standard port numbers starting at 52030 is suggested as a default.
9. It is recommended (but not mandatory) than a VTP/UDP data sink generates periodic ACK (acknowledgement) messages. The structure of these ACK messages is defined below.

A VTP/UDP source should be mindful of the following:

1. It is important to properly fill in the Ethernet, IP and UDP headers with correct source and destination ports, IP addresses, and MAC addresses. How these values are determined will be implementation dependent.
2. UDP allows datagrams up to 64kB, which is larger than Ethernet frame size. VTP/UDP allows VDIF frames that are fragmented over multiple Ethernet frames (but within a single UDP datagram). However such usage is discouraged and generally a VTP/UDP source should ensure the UDP datagram fits within the underlying Ethernet MTU.

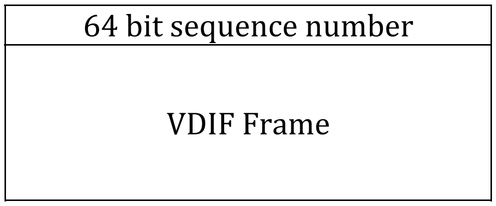

Figure 1: VTP/UDP packet structure

Raw Ethernet

UDP formatted packets are always preferred over raw Ethernet packets, as they are more flexible, allowing for intermediate routers etc. Use of raw Ethernet is to be discouraged. However in some cases raw Ethernet frames may be required or preferred. The basic use of VTP/Ethernet should be the same as for VTP/UDP specified above except:

1. When receiving VTP streams from multiple sources, only a single stream per source is supported and the source MAC address will be used on the sink to separate the streams.

VTP Examples

The following are some example scenarios where VTP would be used.

Local disk recording, over Ethernet

The usual recording mode would be a DAS sampling the signal from the telescope and appropriately channelizing and formatting the data into VDIF frames. The data would be sent from the DAS on a commodity Ethernet interface and received by the Ethernet interface on the disk recorder using the UDP protocol. A number of possible scenarios are possible:

- A single DAS directly connected to a single recorder. The chance of dropped frames is low and packets will always be received in order
- A single DAS connected to single recorder via an Ethernet switch.
- Multiple DAS connected to a single recorder via an Ethernet switch. There is no guarantee of in order arrival of the frames from the various source DAS.
- A single DAS connected, via a switch, to multiple recorders.

If a router is placed between the DAS and recorder, there is a chance of frames from the same source arriving out-of-order or duplicated.

Remote disk recording

Remote disk recording (roughly defined as recording where a recorder is located at a significant distance from a DAS) would usually be setup exactly the same as local disk recording. Generally there should be no need to configure the DAS and recorder any differently from local disk recording, though it should be noted the chance of packet loss on the network will be much higher and often the data will have to be routed (ie using Layer3 connectivity – this would typically mean using UDP or TCP).

Disk playback

VTP is likely to often be used at correlators, in the process of getting data off disk and streamed into the correlator (either software or hardware). Much like disk recording, both local and remote disk playback will be used and there should be no (significant) functional differences between the two. Both UDP and TCP playback must be possible. TCP has a clear advantage, as its build-in congestion control will naturally limit the data rate into the correlator to whatever speed the correlator can handle. If UDP playback were used, some sort of higher-level congestion/throttling control would need to be implemented (which is out-of-scope of this document).

Realtime eVLBI

Realtime eVLBI involves sending data from a DAS at an observatory in realtime to a physically remote data processor. Both TCP and UDP data transport will generally be used on either Layer 2 (Ethernet) or Layer 3 (routed IP) networks. Given the long distances involved, TCP will often be problematic. From the point-of-view of the DAS there is essentially no difference between realtime e-VLBI and remote disk recording.

File transfer

VTP does not cover copying of files across the network, but does not preclude an application from incorporating VTP as the transport mechanism. It is generally assumed though that special purpose file transfer applications would be used (ftp, tsunami-udp etc).

ACK Packets

A feature of modern Ethernet network switches is they dynamically build a switching table based on traffic flow from each port. When data is first sent to a (previously unused) network address, the switch does not know which port it is attached to. It overcomes this by sending the packet to all ports assuming the destination device port will respond, revealing which port it is attached to. The normal ARP process (RFC 826[1]), which is used to map IP addresses to Ethernet addresses, usually ensures the port mapping is setup before any serious data flow occurs. Most switches have a timeout of ~10 minutes on this mapping. For normal bi-directional traffic flow this does not matter as the address cache is constantly updated. However when using protocols such as UDP, which do not include an acknowledgement for received data and with VLBI data which is unidirectional, this timeout is problematic. If nothing is done to ensure some sort of return data then eventually the port mapping is forgotten by the switch and the full data rate is sent to *every* port on the switch (so called flooding). To avoid this it is recommended an acknowledgement packet (ACK) is sent periodically. VTP does not mandate an ACK packet is generated but it is highly recommended. The ACK packet will ensure the switch address table is kept up-to-date and contains simple receive statistics. Such statistics can be used by the source to decide to terminate the stream if packet loss is considered too high. ACK packets are not needed (nor make any sense) for VTP/TCP, only VTP/UDP (and VTP/Ethernet). The following rules define the ACK packet

1. Is sent as a single UDP packet.
2. The destination will usually be to the VTP source but does not have to be.
3. It is recommend an ACK packet is generated once per second.
4. When a sink is receiving VTP streams from multiple sources, an ACK packet must be generated for each stream.
5. The source and sink of the VTP stream will need to negotiate the destination port (and destination address).
6. A default port number of 52020 is suggested.[2]
7. If any of the fields cannot be computed by a sink, then the value can be set to 0xFF..FF to indicate "invalid". This behaviour is discouraged however and sinks should generally fill in all fields.
8. The structure contents of the packet should follow Figure 2 and the description below.

Seconds	Nanoseconds

[1] http://tools.ietf.org/html/rfc826

[2] Ports below 49152 must be registered with Internet Assigned Numbers Authority (IANA) so should not be used

Sequence Number
Received
Reordered

Figure 2: VTP/ACK packet structure.

The definition of the elements is as follows. All integers are little-Endian.

Seconds	Unsigned 32 bit integer	Seconds since Unix Epoch (1 January 1970) when ACK packet generated
Nanoseconds	Unsigned 32 bit integer	Fraction of a second in combination with seconds field above
Sequence Number	Unsigned 64bit integer	Highest sequence number received
Received	Unsigned 64 bit integer	Number of unique frames received (cumulative)
Reordered	Unsigned 64bit integer	Number of frames with Type-P-Reordered=True (see RFC4737[3]) (cumulative).

This could be encoded in the following C struct on a little endian architecture:

```
struct vtp_ack {
        uint32_t sec;
        uint32_t nsec;
        uint64_t seqno;
        uint64_t received;
        uint64_t reordered;
};
```

If a sink is not able to generate ACK packets and it does not generate any "reverse" traffic for other reasons, it is recommended the network infrastructure is configured to hardwire port mappings and take other such measures to avoid flooding the network with undesired packets.

[3] http://tools.ietf.org/html/rfc4737