2002 Antelope Workshop

Frank Vernon	IGPP/UCSD
Danny Harvey	BRTT
Jennifer Eakins	IGPP/UCSD
Gary Pavlis	Indiana University
Daniel Quinlan	BRTT

BRTT

Antelope Methods for Seismic Network Processing

Danny Harvey
Boulder Real Time Technologies, Inc.
danny@brtt.com

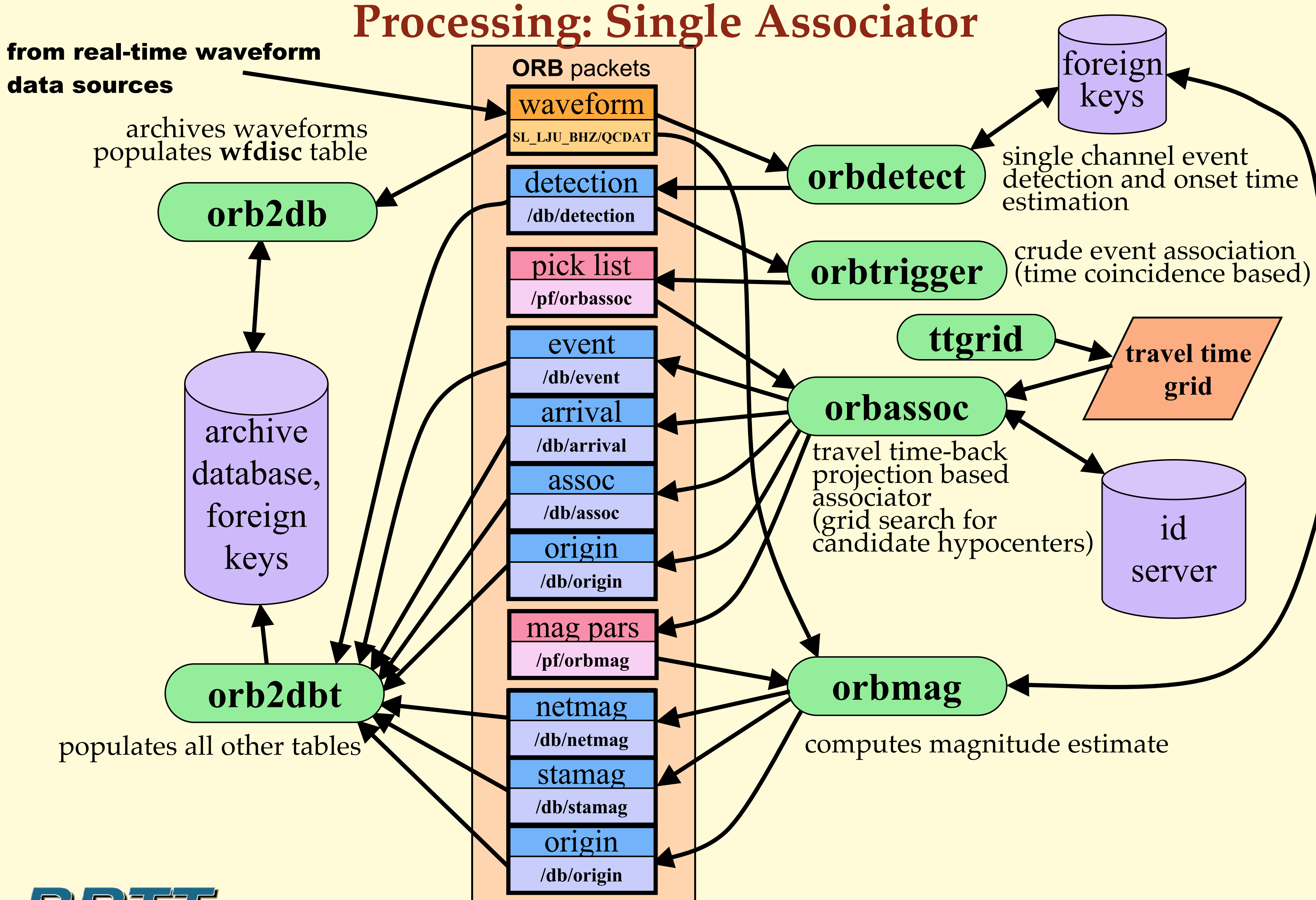
BRTT

2002 Antelope Workshop

"Routine" Automated Seismic Network Processing

	real-time	batch-mode
waveform acquisition	**qt2orb**, **k22orb**, **orb2orb**, **ew2orb**, **cd2orb**, etc.	**orb2db**, **orb2dbt**, **sd2db**, **psd2db**, **dbsteimu**, etc.
single channel detection	**orbdetect**	**dbdetect**
"crude" event association	**orbtrigger**	**dbtrigger**
grid-based event association and location	**orbassoc**	**dbgrassoc**
magnitude estimation	**orbmag**, **orbampmag**	**dbml**, **dbampmag**, etc.
archiving	**orb2db**, **orb2dbt**	

Data Flow for "Routine" Automated Seismic Network Processing: Single Associator



from real-time waveform data sources
ORB packets
waveform
SL_LJU_BHZ/QCDAT
detection
/db/detection
pick list
/pf/orbassoc
event
/db/event
arrival
/db/arrival
assoc
/db/assoc
origin
/db/origin
mag pars
/pf/orbmag
netmag
/db/netmag
stamag
/db/stamag
origin
/db/origin
archives waveforms populates wfdisc table
orb2db
archive database, foreign keys
orb2dbt
populates all other tables
foreign keys
orbdetect
single channel event detection and onset time estimation
orbtrigger
crude event association (time coincidence based)
ttgrid
travel time grid
orbassoc
travel time-back projection based associator (grid search for candidate hypocenters)
id server
orbmag
computes magnitude estimate

BRTT

2002 Antelope Workshop

orbdetect

SYNOPSIS

orbdetect [-v] [-select *expr*] [-tstart *start_time*]
[-twin *minutes*] [-pf *pfname*] [-out *orbout*]
orbname dbname

DESCRIPTION

orbdetect will read waveform data from an input **ORB**, run STA/LTA detectors on one or more channels of the waveform data and write the detection states as Datascope database packets into an output **ORB**. For each channel of data, **orbdetect** will prefilter the data into a specified frequency pass band. Orbdetect can run multiple detections on the same channel of data with different frequency pass bands.

orbdetect – command line arguments

-v Verbose output flag

-select *expr*

An ORB select expression for the input waveform ORB. This argument is optional. If not specified, then no input selection is done (all ORB packets are passed).

-tstart *start_time*

A start time for reading the waveform data. The argument is optional. If not specified, then data will start at the next most recent packet.

-twin *minutes*

A time window for reading waveform data in minutes. This argument is optional. If not specified, then data will be read indefinitely.

-pf *pfname*

Name of program parameter file. The actual parameter file name is *pfname*.pf. If this is not specified, then the default *pfname* is "orbdetect".

-out *orbout*

The name of an output ORB. **orbdetect** will write out the detection states as *detection* table Datascope packets. This argument is optional. If not specified, then the detection states are not output.

orbname

The name of the input ORB containing the waveform data. This argument is required.

dbname

The name of a Datascope database that is used to perform the foreign keys mapping from incoming SEED net-sta-chan-loc codes to output CSS3.0 sta-chan codes (see foreign(3) man page). This argument is required.

orbdetect – parameter file

- Global defaults section
  - Detailed default processing parameters for all channels (averaging type, time windows, gap processing, filter coefficients, detection thresholds, latency parameters, etc.)
- Channel processing list
  - A list of SEED net_sta_chan[_loc] UNIX regular expressions that are matched against incoming waveform packet SEED codes
- Single channel overrides
  - A set of associative arrays with SEED net_sta_chan[_loc] codes as keys and any of the parameters from the global defaults section as entries within the arrays. These are used to "individualize" the parameters for each channel of data.

orbdetect – parameter file example

```
#    Parameter file for orbdetect

#    Following are required and are used as overall defaults

ave_type      rms    # Method for averaging (rms or filter)
sta_twin      1.0    # STA time window
sta_tmin      1.0    # STA minimum time for average
sta_maxtgap   0.0    # STA maximum time gap
lta_twin      10.0   # LTA time window
lta_tmin      5.0    # LTA minimum time for average
lta_maxtgap   4.0    # LTA maximum time gap
nodet_twin    2.0    # no detection if on time is less than this
thresh        5.0    # detection SNR threshold
threshoff     2.5    # detection-off SNR threshold
det_tmin      20.0   # detection minimum on time
det_tmax      600.0  # detection maximum on time
latency       30     # input packet pipe latency (per channel) in packets
filter        none   # default filter
iphase        D      # phase code for onset times
maxfuturetime 600.0 # Maximum number of seconds after system wall clock
otime_noise_tfac  1.0 # ratio of noise tapering time constant to
                      #  sta_twin for onset time estimation
otime_signal_tfac 1.0 # ratio of signal tapering time constant to
                      #  sta_twin for onset time estimation

#    At least one default band must be set set up in the
#    bands table parameter values override default values
#    above for each band

bands &Tbl{ # These are the different frequency bands
     &Arr{       # This is the first frequency band
          sta_twin     5.0
          sta_tmin     5.0
          sta_maxtgap  0.5
          lta_twin     50.0
          lta_tmin     25.0
          lta_maxtgap  4.0
          filter BW 0.5 4 1.2 4  # Butterworth 0.5-1.2 Hz bandpass
          iphase       Dtel
     }
     &Arr{      # This is the second frequency band
          filter BW 3.0 4 0 0 # Butterworth 3 Hz highpass
          iphase       Dloc
     }
}

#    At least one data channel must be specified in the
#    netstachanlocs table

netstachanlocs &Tbl{  # this is the channel processing list
    AZ_.*_BHZ     # matches all stations with net AZ and chan BHZ
    IU_TUC_BHZ_00 # matches only net IU, sta TUC, chan BHZ, loc 00
}

#    Individual channel parameters may be overriden below -
#    following entries are optional


AZ_PFO_BHZ &Arr{  # overrides parameters for channel AZ_PFO_BHZ
    thresh       10.0
    threshoff    5.0
}
```

BRTT

orbdetect – notes

- SEED to CSS3.0 name mapping
  - all ORB waveform packets use SEED naming convention (globally unique)
  - all database tables use CSS3.0 naming convention (sta-chan only, no net and loc)
  - “foreign keys” tables (*snetsta* and *schanloc*) are used to define this mapping
- waveform filtering
  - digital recursion filtering (simple, fast, robust and stable)
  - preload recursion “past” values (removes transients due to 0th order discontinuities)
  - general Butterworth filter (BW *low_freq low_poles high_freq high_poles*)
- waveform packet timing issues
  - time-ordered packet “pipe” used to handle out-of-time-order packets
  - reject packets with future time stamps
- restarting
  - parameter file read only once at program startup
  - no “state” file, starting point controlled manually using **–tstart** command line argument

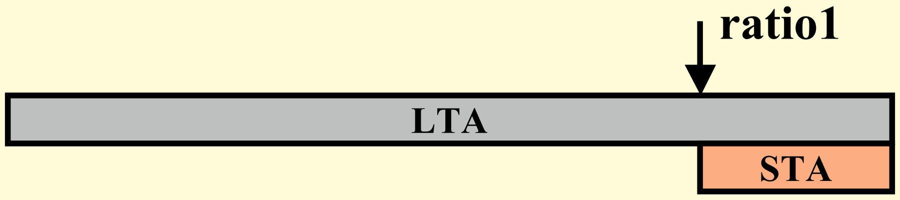
orbdetect – basic data processing sequence

For each incoming waveform ORB packet:

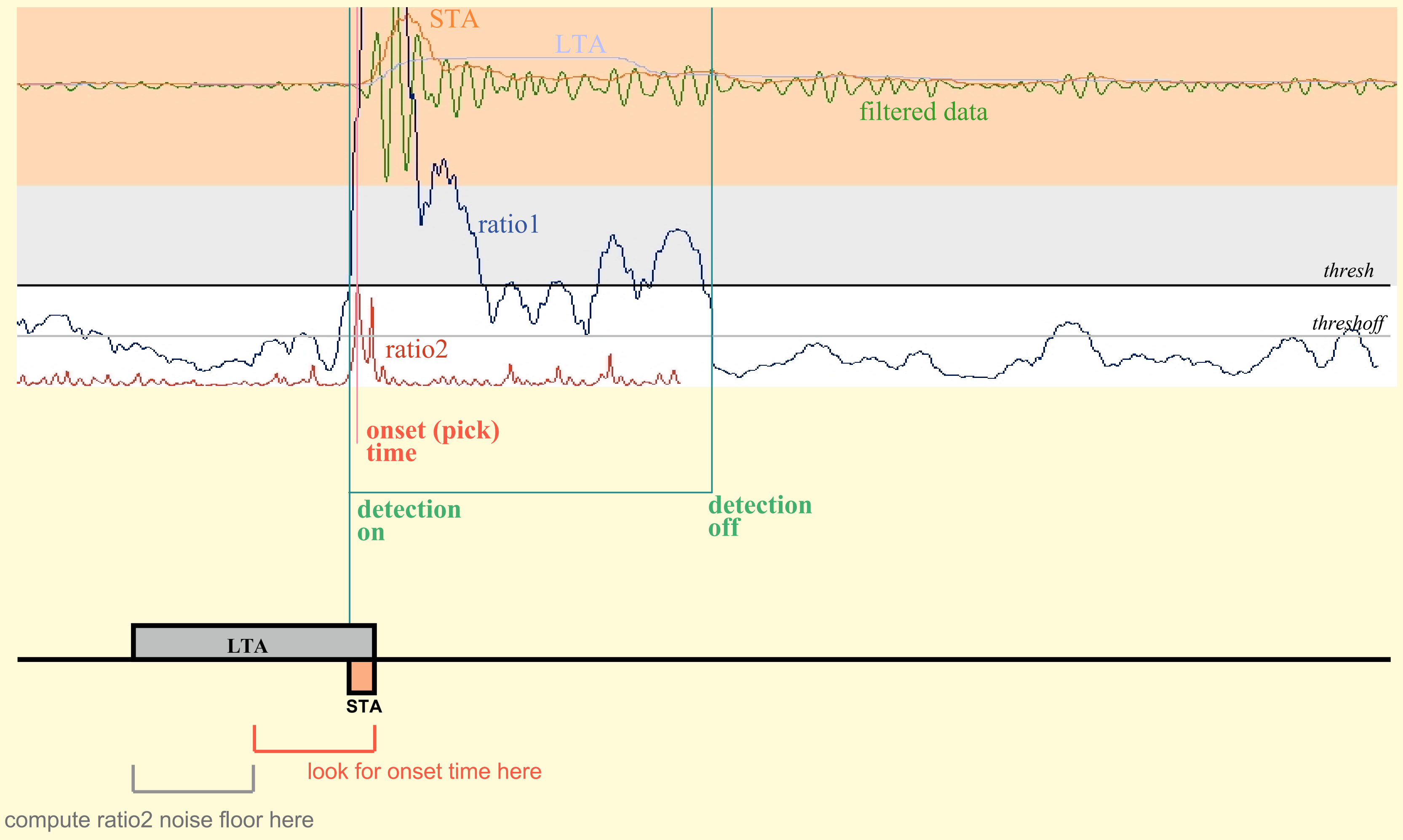
1. Read and unstuff packet
2. For each ORB packet channel:
    1. Match against regular expressions in *netstachanlocs* table
    2. Reject packet-channel if time > **now()** + *maxfuturetime*
    3. Push packet-channel into appropriate packet-channel pipe
    4. Pop available packet-channels from packet-channel pipe. For each:
        1. For each frequency band:
            1. Filter data
            2. Compute time-domain LTA function
            3. Compute time-domain STA function
            4. Compute time-domain ratio1=STA/LTA function
            5. Send ratio1, LTA, STA functions and filtered data function to detection processor and onset time estimator

orbdetect – notes on basic data processing

- All processing is data-packet driven (no timeouts, lots of buffering)
- Channel processing is dynamic
- STA, LTA gap/edge processing:
  - Controlled by *sta_tmin*, *sta_maxtgap*, etc. parameters
  - For each time sample, averages are marked with a special "undefined" value if the gap parameters are exceeded
- ratio1=STA/LTA function processing:
  - STA and LTA averaging time windows overlap (stable at edges and gaps)
  - If either STA or LTA sample values in ratio are marked as "undefined", then the ratio1 sample value is marked as "undefined"



orbdetect – detection processing



BRTT

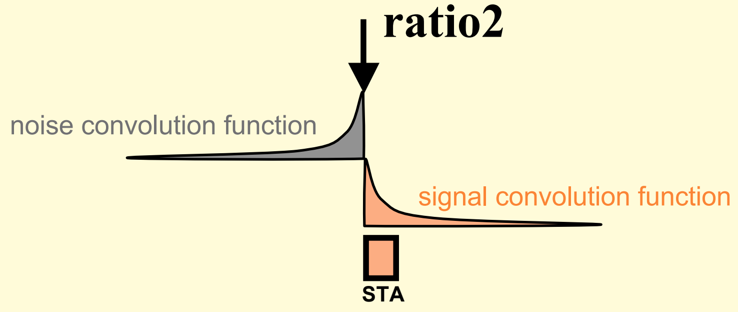
2002 Antelope Workshop

orbdetect – notes on detection processing

- When ratio1(t) > *thresh*, a detection is opened, its start time is set and the value of LTA(t) is saved in LTA_hold
- After a detection has started, ratio1(t) is recomputed as STA(t)/LTA_hold and is continuously checked to see if ratio1(t) < *threshoff*
- When ratio1(t) < *threshoff*, the detection is closed and ratio1(t) is computed again as STA(t)/LTA(t)
- If the detection duration is < *nodet_twin*, then the entire detection, including its associated onset time estimate, is ignored and nothing is output
- The detection off time is subject to the limitations imposed by the *det_tmin* and *det_tmax* parameters

orbdetect – onset (pick) time estimation

- A new ratio2(t) function is computed when there is sufficient data available after a detection has started.
- ratio2(t) is a true signal-to-noise ratio based upon the ratio of time-abutting signal and noise windows.
- Both signal and noise functions are computed by convolving the square of the filtered data with an exponential time function (one pole low-pass filter).
- The noise function is computed using recursive digital filtering in the forward time direction and the signal function is computed using recursive digital filtering in the reverse time direction.
- The time constants for both filters are nominally the same as *sta_twin*. This can be overridden with the *otime_signal_tfac* and *otime_noise_tfac* parameters.
- The noise function is limited by a noise floor value that is computed from the first half of the LTA time window at the detection start time (in order to stabilize the ratio2(t) function).
- The onset time is chosen as the time when ratio2(t) is at its maximum value within a time window from the detection on time – 0.5**lta_twin* + *sta_twin* to the detection in time + *sta_twin*.

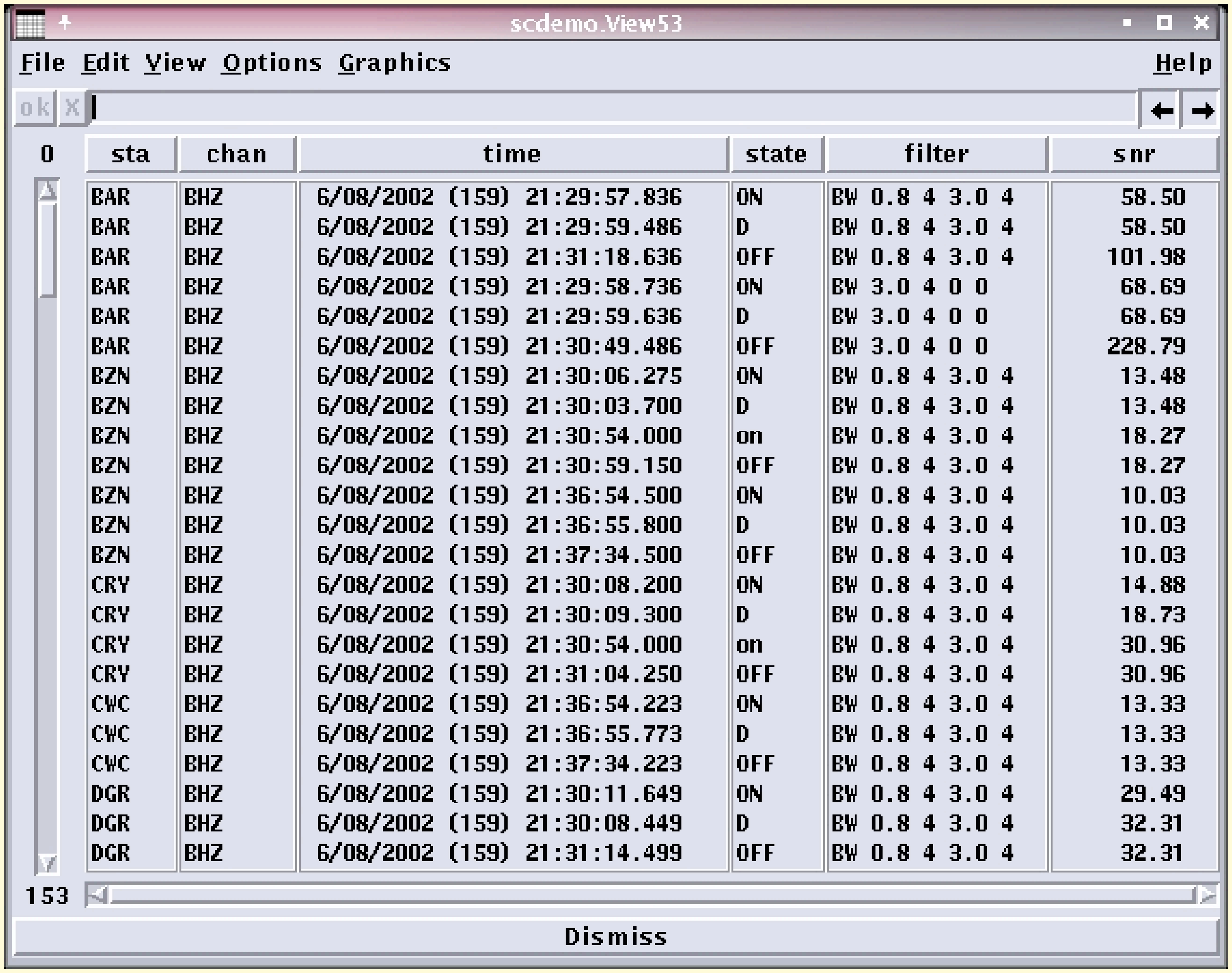


BRTT

orbdetect – output

- Rows in the *detection* table as ORB packets *(/db/detection* srcname) are written to the output ORB in as close to real-time as possible
- The *detection* table is defined in the **rt1.0** schema, an extension to the **css3.0** schema
- For each channel-frequency band, each detection state change (ON and OFF) are written out as separate *detection* rows. In addition, the onset (pick) time estimate is also written as a separate *detection* row. The field *state* within the *detection* table is used to indicate the detection state changes and the onset time estimate
- A signal-to-noise ratio is also stored in each *detection* row. This represents the maximum signal-to-noise that was measured during the detection.

orbdetect – output example

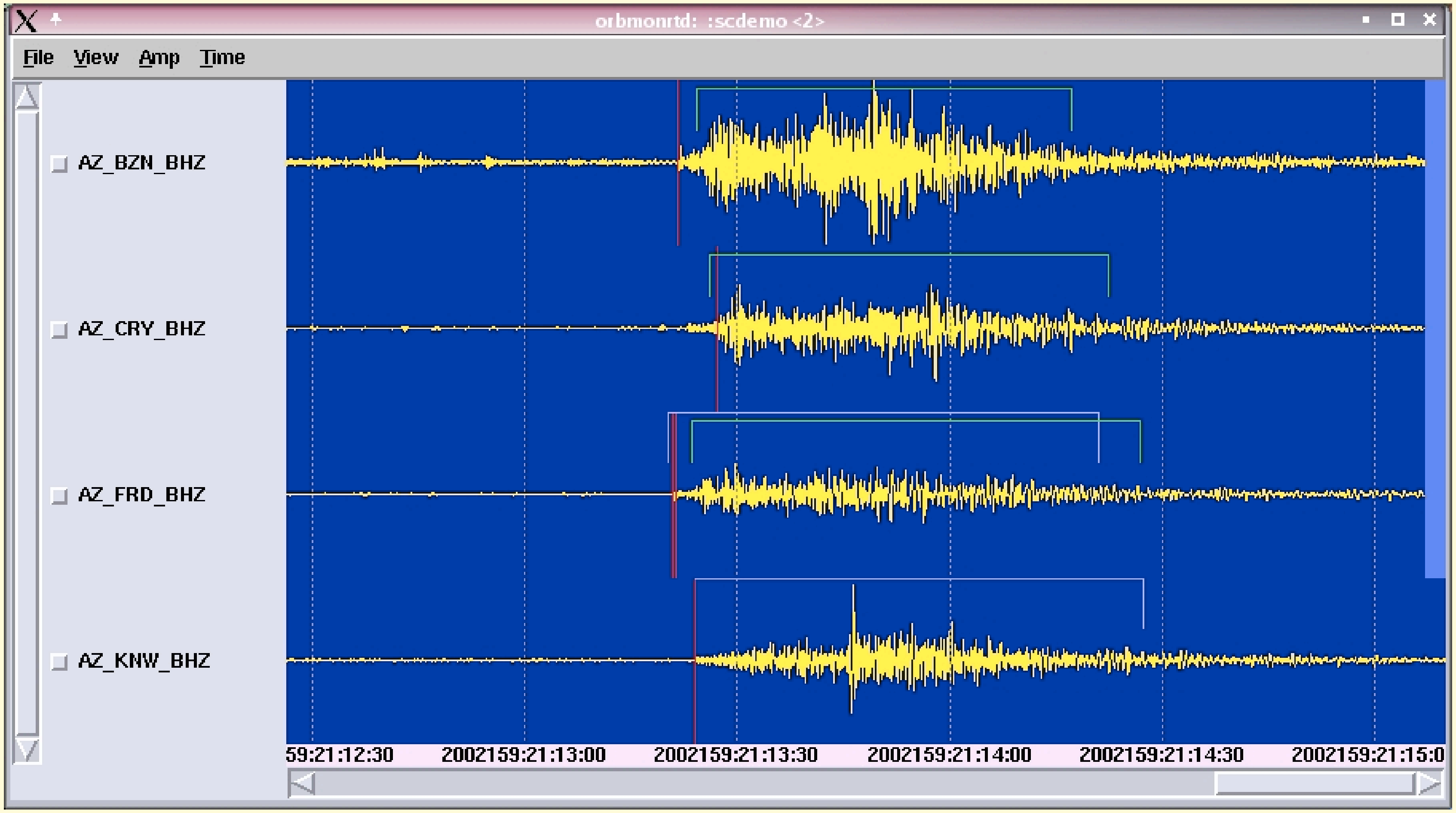


sta	chan	time	state	filter	snr
BAR	BHZ	6/08/2002 (159) 21:29:57.836	ON	BW 0.8 4 3.0 4	58.50
BAR	BHZ	6/08/2002 (159) 21:29:59.486	D	BW 0.8 4 3.0 4	58.50
BAR	BHZ	6/08/2002 (159) 21:31:18.636	OFF	BW 0.8 4 3.0 4	101.98
BAR	BHZ	6/08/2002 (159) 21:29:58.736	ON	BW 3.0 4 0 0	68.69
BAR	BHZ	6/08/2002 (159) 21:29:59.636	D	BW 3.0 4 0 0	68.69
BAR	BHZ	6/08/2002 (159) 21:30:49.486	OFF	BW 3.0 4 0 0	228.79
BZN	BHZ	6/08/2002 (159) 21:30:06.275	ON	BW 0.8 4 3.0 4	13.48
BZN	BHZ	6/08/2002 (159) 21:30:03.700	D	BW 0.8 4 3.0 4	13.48
BZN	BHZ	6/08/2002 (159) 21:30:54.000	on	BW 0.8 4 3.0 4	18.27
BZN	BHZ	6/08/2002 (159) 21:30:59.150	OFF	BW 0.8 4 3.0 4	18.27
BZN	BHZ	6/08/2002 (159) 21:36:54.500	ON	BW 0.8 4 3.0 4	10.03
BZN	BHZ	6/08/2002 (159) 21:36:55.800	D	BW 0.8 4 3.0 4	10.03
BZN	BHZ	6/08/2002 (159) 21:37:34.500	OFF	BW 0.8 4 3.0 4	10.03
CRY	BHZ	6/08/2002 (159) 21:30:08.200	ON	BW 0.8 4 3.0 4	14.88
CRY	BHZ	6/08/2002 (159) 21:30:09.300	D	BW 0.8 4 3.0 4	18.73
CRY	BHZ	6/08/2002 (159) 21:30:54.000	on	BW 0.8 4 3.0 4	30.96
CRY	BHZ	6/08/2002 (159) 21:31:04.250	OFF	BW 0.8 4 3.0 4	30.96
CWC	BHZ	6/08/2002 (159) 21:36:54.223	ON	BW 0.8 4 3.0 4	13.33
CWC	BHZ	6/08/2002 (159) 21:36:55.773	D	BW 0.8 4 3.0 4	13.33
CWC	BHZ	6/08/2002 (159) 21:37:34.223	OFF	BW 0.8 4 3.0 4	13.33
DGR	BHZ	6/08/2002 (159) 21:30:11.649	ON	BW 0.8 4 3.0 4	29.49
DGR	BHZ	6/08/2002 (159) 21:30:08.449	D	BW 0.8 4 3.0 4	32.31
DGR	BHZ	6/08/2002 (159) 21:31:14.499	OFF	BW 0.8 4 3.0 4	32.31

BRTT

2002 Antelope Workshop

orbdetect – detection glyphs in **orbmonrtd**



BRTT

2002 Antelope Workshop

orbtrigger

SYNOPSIS

orbtrigger [-v] [-tstart *start_time*] [-target_orbassoc *torbassoc*]
[-pf *pfname*] [-out *orbout*] *orbname*

DESCRIPTION

orbtrigger will look for seismic events by reading *detection* table rows from an input ORB, and looking for detection concurrency over a specified set of stations. If a candidate event is found, then **orbtrigger** will write a single parameter file object per event, as an ORB packet, targeted for **orbassoc**. Each of these parameter file ORB packets contains a candidate pick list with onset time estimates that are to be used by **orbassoc**.

orbtrigger – command line arguments

-v Verbose output flag

-tstart *start_time*

A start time for processing the *detection* ORB packets. The argument is optional. If not specified, then **orbtrigger** will start with the next most recent packet.

-target_orbassoc *torbassoc*

A target name for the parameter file packet that is targeted for **orbassoc**. The default is "orbassoc".

-pf *pfname*

Name of program parameter file. The actual parameter file name is *pfname*.pf. If this is not specified, then the default *pfname* is “orbtrigger".

-out *orbout*

The name of an output ORB. **orbtrigger** will write out the network trigger as parameter file objects targeted for **orbassoc**. This argument is optional. If not specified, then no ORB output occurs.

orbname

The name of the input ORB containing the *detection* rows. This argument is required.

orbtrigger – parameter file example

```
#
#  Parameter file for orbtrigger

twin        30.0   # This is the concurrency time window in seconds. Detections on at
                   #   least nstations stations must occur within twin seconds in order
                   #   for a network trigger to occur.
nstations   5      # The minimum number of stations thershold.     Detections on at
                   #   least nstations stations must occur within twin seconds in order
                   #   for a network trigger to occur.
latency     20.0   # This is amount of time in seconds that orbtrigger will wait for
                   #   inter-channel data latency.
maxwaittime 0.0    # The amount of time in seconds to wait before outputing the parameter
                   #   file object targeted for orbassoc.
maxdettime  120.0  # maximum time interval in seconds for an input detection to be on
detlatency  300.0  # This is a time latency value in seconds that is used in conjunction
                   #   with "maxdettime" above to determine when an input detection
                   #   should be closed.
stachans &Tbl{     # List of station-channels for processing.
    BZN     .*
    CRY     .*
    ELKS    .*
    FRD     .*
    GLAC    .*
    KNW     .*
    LVA2    .*
}
```

orbtrigger – basic data processing sequence

For each incoming */db/detection* ORB packet:

1. Read and unstuff packet
2. Match sta-chan against regular expressions in *stachans* table
3. Perform detection buffer processing using just this *detection* row
4. Perform network trigger processing using all existing detection buffers

NOTES:

- CSS3.0 naming conventions used throughout (i.e. no SEED net or loc codes)
- Restarting
  - parameter file read only once at program startup
  - no "state" file, starting point controlled manually using **–tstart** command line argument
- All processing is ORB-packet driven (no wall-clock timeouts, lots of buffering)
- Station-channel-filter processing is dynamic

BRTT

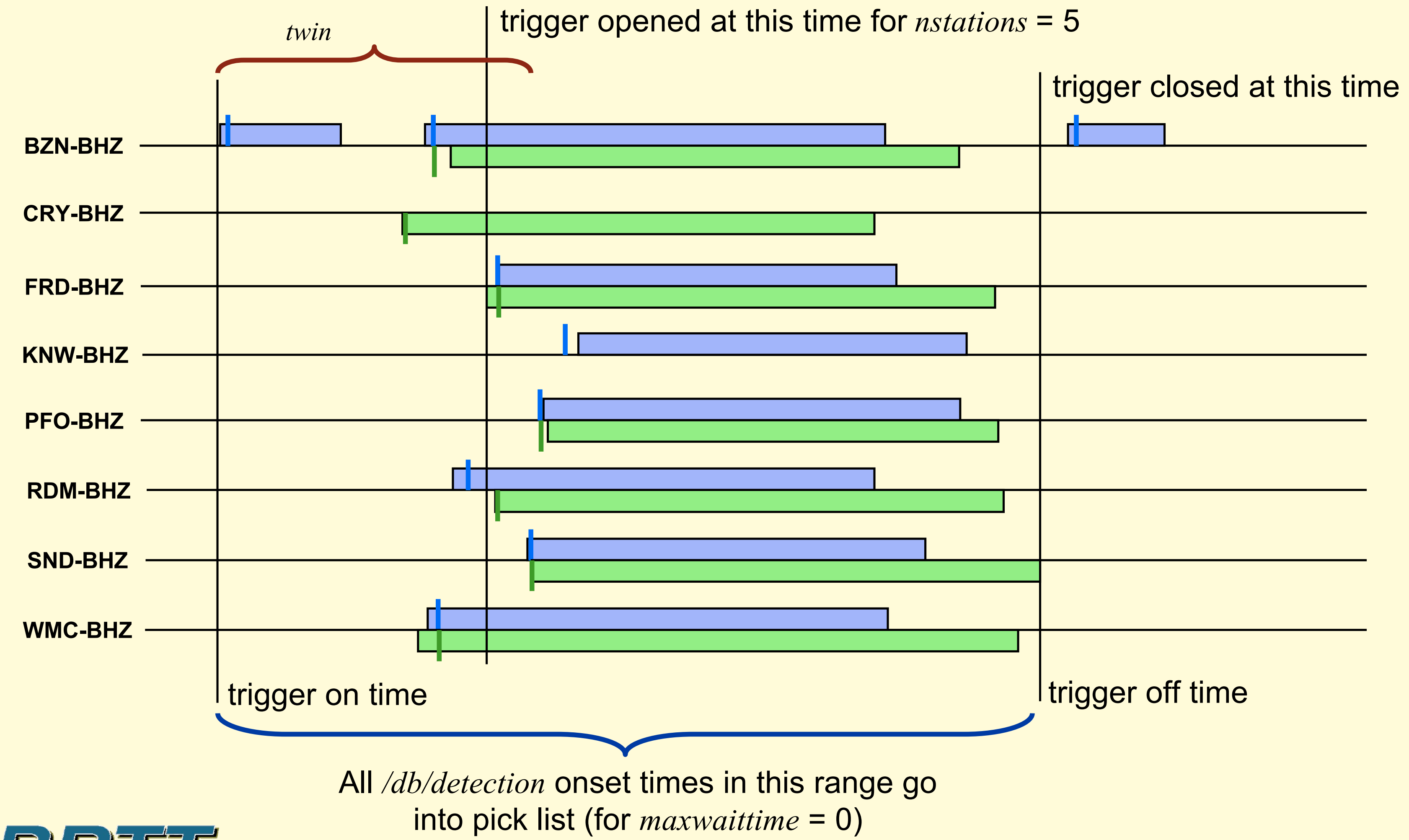
orbtrigger – detection buffer processing

Objective: Assimilate individual incoming *detection* rows into **orbtrigger** detection buffers for each station-channel-filter:

- Each **orbtrigger** detection buffer contains all of the states for a single detection (i.e. normally, three incoming *detection* rows make up a single **orbtrigger** detection buffer – ON, onset time, OFF)
- "Missing" *detection* rows are handled with the *maxdettime* and *detlatency* parameters:
  - Missing *detection* rows can occur whenever **orbdetect** is stopped (or dies) or any other situation where the flow of */pf/detection* ORB packets is interrupted
  - For each new *detection* row, its time is compared against all currently open **orbtrigger** detection buffers:
    - Detection buffers with times that are more than *maxdettime* + *detlatency* seconds before the new *detection* row time are marked as being "expired" and are closed out allowing trigger processing to continue with these buffers

orbtrigger – network trigger processing

Objective: Scan all **orbtrigger** detection buffers to create/modify a network trigger and output an event pick list when appropriate:



twin
trigger opened at this time for *nstations* = 5
trigger closed at this time
BZN-BHZ
CRY-BHZ
FRD-BHZ
KNW-BHZ
PFO-BHZ
RDM-BHZ
SND-BHZ
WMC-BHZ
trigger on time
trigger off time
All */db/detection* onset times in this range go into pick list (for *maxwaittime* = 0)

BRTT

2002 Antelope Workshop

orbtrigger – notes on network trigger processing

- Only one network trigger at a time
- A trigger is opened when the detection ON times from *nstations* stations occur within a time duration of *twin*, the trigger ON time is set to the earliest detection ON time that initiated the trigger
- After a trigger is opened, it continues to accumulate detection buffers until there are no detection buffers with overlapping times, at which time the trigger is closed and its OFF time is set to the latest detection buffer OFF time
- While a trigger is open, **orbtrigger** accumulates all of the detection onset time estimates into an internal pick list buffer
- The internal pick list buffer is written to the output ORB as a parameter file packet when:
  - The trigger is closed, if *maxwaittime* = 0, or
  - A new detection time > trigger_ON_time + *maxwaittime*
  - Note that once the pick list is output, it will not be output again until a new trigger is opened

orbtrigger – output */pf/orbassoc* pick list

```
arrivals        &Tbl{
        BAR BHZ Dl 1023647045.63560 68.69000
        BAR BHZ Dt 1023647045.48560 58.50000
        BZN BHZ Dt 1023647049.70000 13.48000
        CRY BHZ Dt 1023647055.30000 18.73000
        DGR BHZ Dl 1023647059.99910 18.19000
        DGR BHZ Dt 1023647054.44910 32.31000
        FRD BHZ Dl 1023647049.02500 10.88000
        FRD BHZ Dt 1023647049.30000 13.22000
        GLA BHZ Dl 1023647037.76060 75.61000
        GLA BHZ Dt 1023647037.81060 119.94000
        JCS BHZ Dl 1023647045.73380 241.03000
        JCS BHZ Dt 1023647047.33380 50.17000
        KNW BHZ Dl 1023647052.12500 8.88000
        LVA2 BHZ Dl 1023647047.42500 30.58000
        LVA2 BHZ Dt 1023647047.40000 24.31000
        MONP BHZ Dl 1023646931.10000 5.12000
        MONP BHZ Dl 1023647043.42500 77.79000
        MONP BHZ Dt 1023647042.70000 102.69000
        PLM BHZ Dl 1023647050.51180 20.99000
        PLM BHZ Dt 1023647054.16180 35.24000
        RDM BHZ Dt 1023647058.02500 16.68000
        SND BHZ Dl 1023647049.95000 16.80000
        SND BHZ Dt 1023647050.10000 6.99000
        SOL BHZ Dt 1023647055.12500 13.14000
        TRO SHZ Dl 1023647048.10000 8.45000
        TRO SHZ Dt 1023647048.00000 12.60000
        WMC BHZ Dt 1023647050.67500 12.71000
}
```

orbassoc

SYNOPSIS

orbassoc [-start {*pktid*|*time*|OLDEST}] [-number *number*] [-nowait]
[-select *expr*] [-target_orb2dbt *torb2dbt*]
[-target_orbmag *torbmag*] [-pf *pfname*]
[-auth *author*] [-dbnextid *dbnextid*]
orbin orbout ttgridf1 [*ttgridf2* [...]]

DESCRIPTION

orbassoc is an ORB client program that continuously looks for candidate pick lists for event association contained in ORB parameter file packets, as produced by **orbtrigger**. For each of these pick lists, **orbassoc** searches over one or more spatial grids for a candidate hypocenter that produces theoretical travel time to each station that most closely matches the observations. Both P and S travel times can be used. If a suitable match is found to the observations, then the associated arrivals, including time and phase information, are written to an output ORB as either Datascope database packets (*arrival* rows), along with the associations (*assoc* rows) and the associated location (*event* and *origin* rows), or all of the database info is encapsulated into a single special parameter file packet destined for further processing by **orb2dbt**.

orbassoc – command line arguments

-tstart {*pktid*|*time*|OLDEST}

A start time for processing the pick list ORB packets from **orbtrigger**. The argument is optional. If not specified, then **orbassoc** will start with the next most recent packet.

-number *number*

How many input parameter file packets to process. This argument is optional. If this is not specified, then there will be no limit on the number of parameter object packets processed.

-nowait

If this is specified, then **orbassoc** will exit when the ORB read pointer gets to the most recent parameter file packet. If not specified, then **orbassoc** will wait for new parameter object packets to arrive indefinitely.

-select *expr*

An ORB select expression that is applied to all ORB reads. The name should be consistent with the **orbassoc** target names specified in **orbtrigger**. The default is "/pf/orbassoc".

-target_orb2dbt *torb2dbt*

A target name for the output parameter file packet that is targeted for **orb2dbt**. The default is “orb2dbt".

-target_orbmag *torbmag*

A target name for the output parameter file packet that is targeted for **orbmag**. The default is “orbmag".

-pf *pfname*

Name of program parameter file. The actual parameter file name is *pfname*.pf. If this is not specified, then the default *pfname* is “orbassoc".

-auth *author*

An author name to be filled into the database *auth* fields. If this argument is not specified, then the default *author* is "orbassoc".

-dbnextid *dbnextid*

Datascope database name of database that contains the *lastid* table used for assigning new *evid*, *orid* and *arid* values for the new database packets. This argument is optional. If specified, then the database rows are written to the output ORB and the **orbmag** parameter file packet is also written. If not specified, then the results are encapsulated into a single parameter file and the **orbmag** parameter file packet is not output.

orbin

The name of the input ORB containing the pick list parameter file packets from **orbtrigger**. This argument is required.

orbout

The name of an output ORB. This argument is required. All new database packets and/or parameter file packets are written here.

ttgridf1 ttgridf2 ...

Names of travel time grid files, as produced by the program **ttgrid**. At leat one file name must be specified.

orbassoc – modes of operation

- In “legacy” mode **orbassoc** writes out *event*, *origin*, *assoc* and *arrival* database ORB packets as well as a parameter file packet targeted for **orbmag.** In this mode:
  - **orb2dbt** simply passes on the database rows to the archive database without modifications (its normal mode of operation for all database tables)
  - **orbassoc** is responsible for procuring new and unique database ids: *evid*, *orid* and *arid*
  - The –dbnextid **orbassoc** command line argument must be specified. The *dbnextid* database must contain a *lastid* table which is used to assign new database ids. In many cases this database will be managed by a database id-server.
  - **orbassoc** will also write out a special ORB parameter file packet with all of the info necessary for **orbmag** to compute magnitudes.
  - Only a single associator (**orbassoc**) may be run. This is because multiple simultaneous associators can often put out the same exact *arrival*s and *origin* estimates of the same event which would not be properly merged by **orb2dbt**.
- In “multiple associator” mode **orbassoc** encapsulates all of the *event*, *origin*, *assoc* and *arrival* database info for each event into a single ORB parameter file packet that is targeted for **orb2dbt**. In this mode:
  - **orb2dbt** performs a full event association with existing events in the archive database using the event info contained in the */pf/orb2dbt* parameter file packets from **orbassoc**
  - **orb2dbt** is responsible for procuring new and unique database ids
  - The –dbnextid **orbassoc** command line argument must not be specified, since **orbassoc** has no requirement to assign unique database ids (that is now done by **orb2dbt**). In fact the existence or absence of this command line argument is the switch used by **orbassoc** to determine which operational mode it is to use.
  - **orbassoc** will not write out a special ORB parameter file packet with all of the info necessary for **orbmag** to compute magnitudes. This is done now by **orb2dbt**.
  - Multiple simultaneous associators (**orbassoc**) may be safely run. This is because the conflicts that arise with multiple associators will be properly resolved by **orb2dbt**.

orbassoc – ttgrid files

- All phase travel times are read by **orbassoc** from one or more travel time grid files, also known as *ttgrid* files.
- Each *ttgrid* file contains one or more travel time grid objects.
- A single travel time grid object is composed of sets of travel times in binary form corresponding to a particular set of receiver locations and a grid of source locations for one or more seismic phases.
- **orbassoc** does no on-the-fly travel time computations. All travel times come from *ttgrid* files. This results in efficient operation and decoupling of the complexities of computing travel times from those of doing the associations.
- Currently the only software available in the ***Antelope*** release for computing *ttgrid* files is the program **ttgrid**. The **ttgrid** program currently supports the computation of 2D grids, 3D grids and slowness-based teleseismic grids.
- The *ttgrid* file concept provides for completely general specification of travel times from grids of sources to sets of receivers. In theory, these files could contain travel times with complex receiver-source corrections and employ full 3-dimensional wave propagation models.

orbassoc – parameter file example

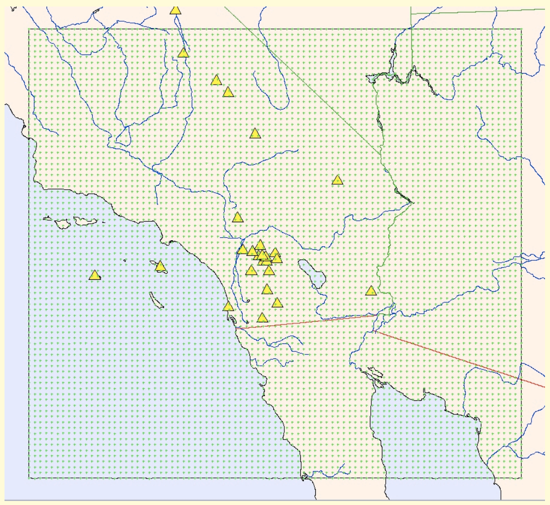
```
#
#  Parameter file for orbassoc

grid_params &Arr{
     local &Arr{
             nsta_thresh     6       # Minimum allowable number of stations
             nxd             11      # Number of east-west grid nodes for depth scans
             nyd             11      # Number of north-south grid nodes for depth scans
             cluster_twin    2.5     # Clustering time window
             try_S           no      # yes = Try observations as both P and S
                                     # no  = Observations are P only
             drop_if_on_edge yes     # Drop if solution is on the edge of the grid
     }
     regional &Arr{
             nsta_thresh     6       # Minimum allowable number of stations
             nxd             11      # Number of east-west grid nodes for depth scans
             nyd             11      # Number of north-south grid nodes for depth scans
             cluster_twin    2.0     # Clustering time window
             try_S           no      # yes = Try observations as both P and S
                                     # no  = Observations are P only
             drop_if_on_edge yes     # Drop if solution is on the edge of the grid
     }
     tele &Arr{
             nsta_thresh     6       # Minimum allowable number of stations
             cluster_twin    2.0     # Clustering time window
             try_S           no      # yes = Try observations as both P and S
                                     # no  = Observations are P only
             drop_if_on_edge no      # Drop if solution is on the edge of the grid
     }
}
```

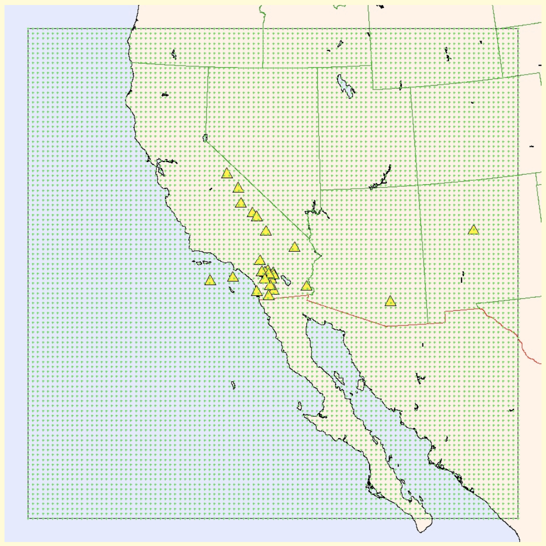
orbassoc – parameter file notes

- The **orbassoc** parameter file must contain an associative array with the name *grid_params*.
- The *grid_params* array must contain a set of associative arrays each with the name of a grid contained in one of the *ttgrid* files. Grid names are assigned to each grid when the *ttgrid* files are created (normally by the program **ttgrid**).
- A single instance of **orbassoc** can process multiple travel time grids. When more than one grid is being processed, **orbassoc** will chose the single best solution from all of the grids, base upon the grid node with the most number of associated stations and the smallest residual.
- This processing method allows the user to specify both local and teleseismic source grids. **orbassoc** usually can correctly discriminate between local and teleseismic events using this technique.

orbassoc – source grids

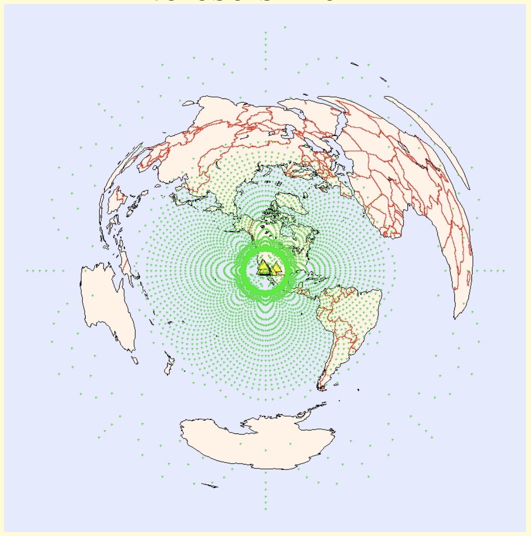


local



regional

teleseismic



BRTT

2002 Antelope Workshop

orbassoc – basic association processing

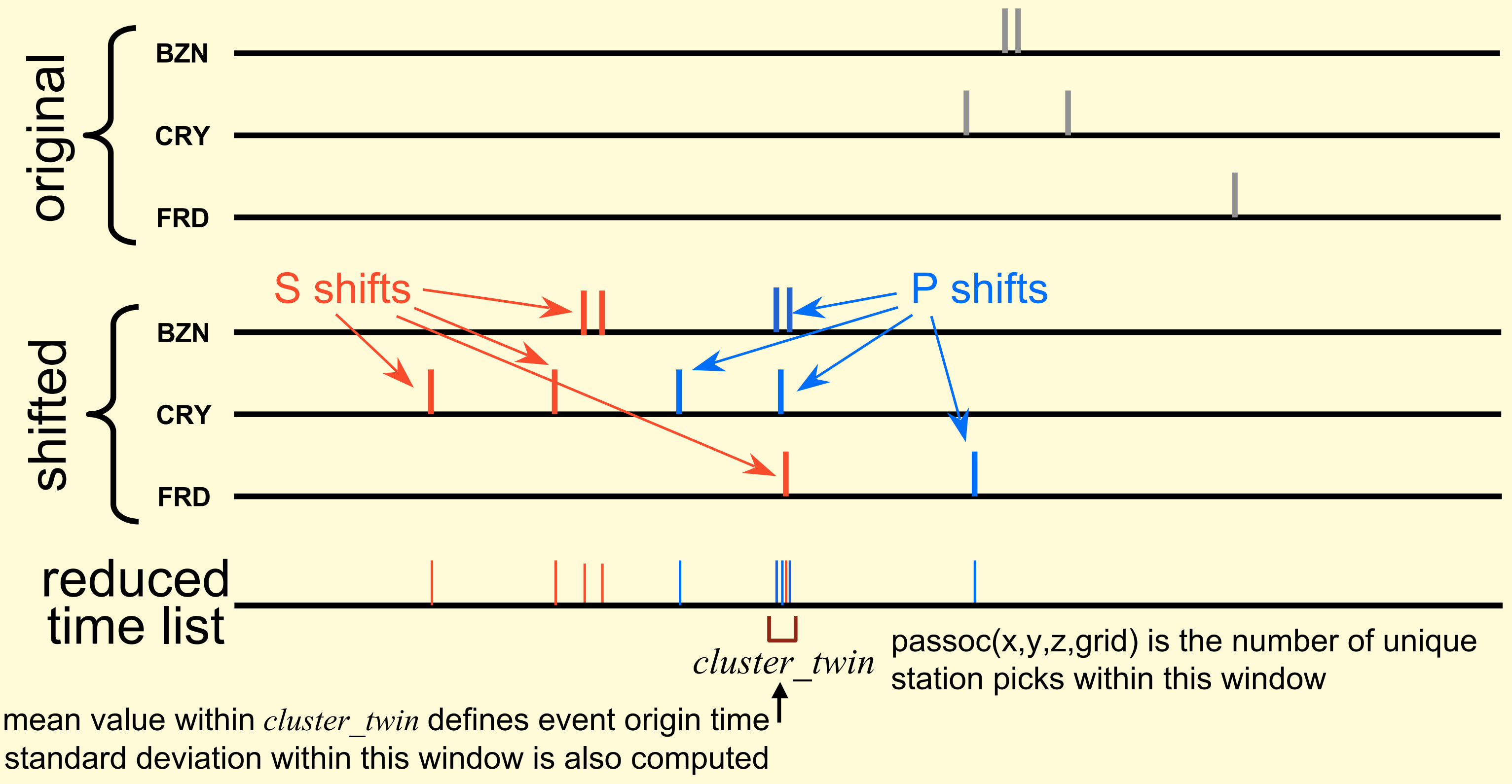
For each incoming */pf/orbassoc* ORB packet:

1. Read and unstuff packet to get the pick list.
2. For each travel time grid object:
    1. For each source location grid node:
        1. Compute passoc(grid,x,y,z), a positive definite performance function that will be used to determine the best source location.
    2. Search for the source location that produces a maximum value of passoc(x,y,z,grid).
3. Search for the grid and source location that produces a maximum value of passoc(x,y,z,grid).
4. Output the corresponding location, arrivals and associations.

orbassoc – passoc processing

For each grid-source location node:

1. All of the times in the pick list are reduced by the phase travel times to an equivalent origin time.
2. These reduced pick times for each travel time phase are put into a reduced time list for subsequent time-clustering analysis:

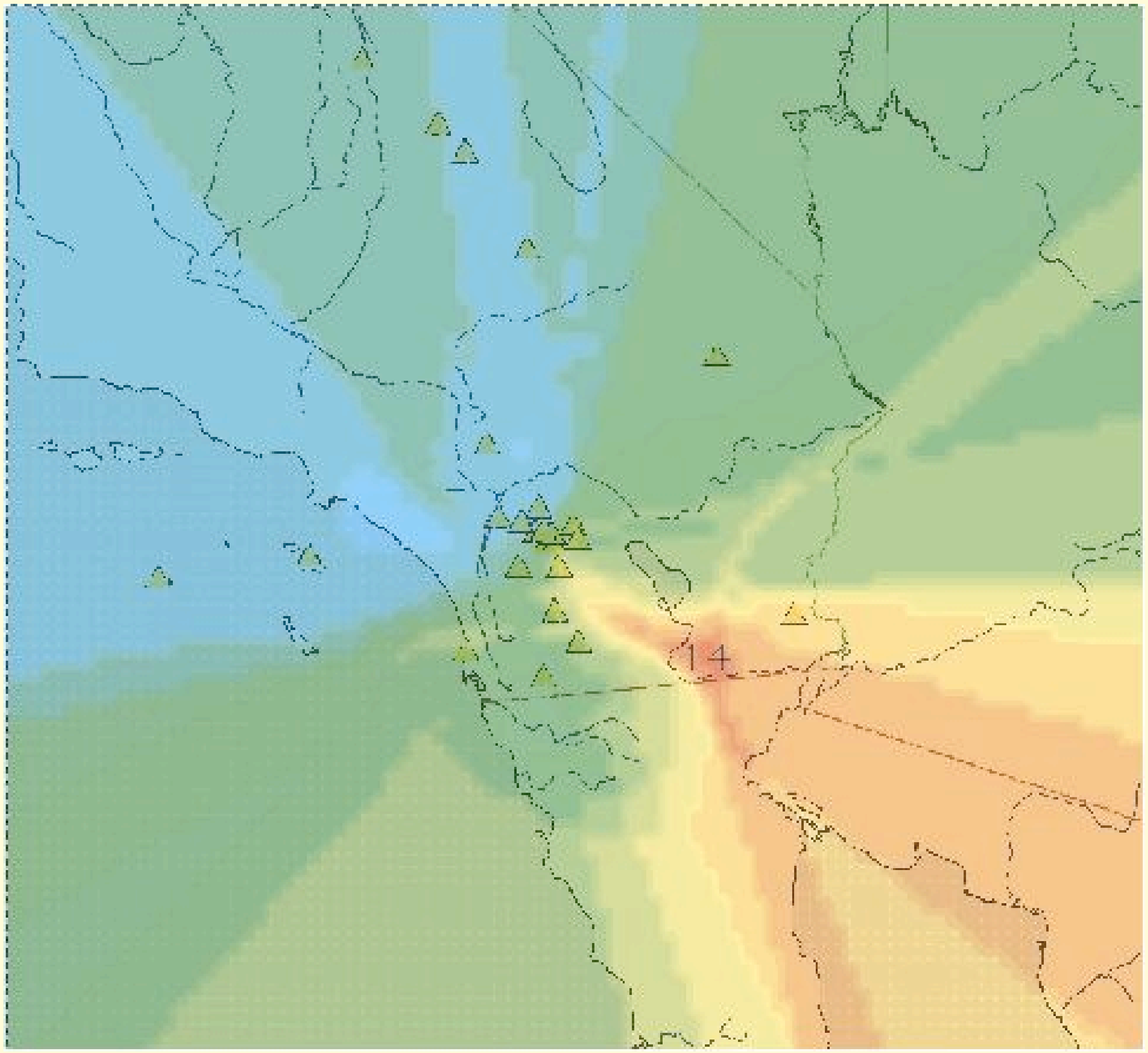


passoc(x,y,z,grid) is the number of unique station picks within this window

mean value within *cluster_twin* defines event origin time
a standard deviation within this window is also computed

BRTT

orbassoc – example passoc(x,y,local)



BRTT

orbassoc – notes

- In the time cluster analysis, an indexing scheme is used to insure that only one pick per station is used
- For all x,y,z,grid where passoc(x,y,z,grid) is the same, the x,y,z,grid that produces the smallest residual standard deviation is chosen as the solution.
- Generally, the grid-source node search is a global search. However, it is possible to constrain the search in depth for 3D grids using the *nxd* and *nyd* parameters.
- For each grid, if the x,y,z solution is on the edge of the grid and the *drop_if_on_edge* parameter is set to yes, then the solution for that grid is disallowed.
- For each grid, if the number of stations in *cluster_twin* is < *nsta_thresh* parameter, then the solution for that grid is disallowed.

Final thoughts

- Look at the ttgrid(1) man page to see how to make *ttgrid* files. Look at the ttgrid_show(1) and displayttgrid(1) man pages to see how to inspect *ttgrid* files. If there is interest in the user community, BRTT could develop some more general methods for creating *ttgrid* files that would allow users to employ their own travel time computation methods.
- Look at the orbmag(1) man page to see how to compute magnitudes in real-time. Also look at Nikolaus Horn's **orbampmag** program (orbampmag(1) man page), which provides a more generalized real-time magnitude calculator.
- Look into setting up multiple associators by reading the orb2dbt(1) man page.
- All of the basic processing algorithms are the same for **dbdetect**, **dbtrigger** and **dbgrassoc**. These can be used effectively to "tune" parameters for the real-time system.