ARCHITECTURAL REPORT

Noaaport Broadcast System Processor (Nbsp)

Source Code Analysis

Primary Sources: nbspd.conf ($Id v1.40), dafilter.conf.in (211 lines), logrotate.conf (61 lines), hourly-cleanup.conf, filter stdin protocol, Perl filter example code, Tcl configuration override loader, repository file tree

March 2026

Table of Contents

1. Executive Summary

This report analyzes the Nbsp architecture exclusively through its source code: the complete nbspd.conf configuration source (identified by CVS tag $Id: nbspd.conf,v 1.40 2006/02/28), the dafilter.conf.in Tcl filter configuration (211 lines), the logrotate.conf operational configuration (61 lines), the hourly-cleanup.conf scripts, the filter stdin interface protocol with a working Perl implementation, the Tcl configuration override loader, and the repository file tree structure.

The source code reveals a system with a sophisticated internal architecture: a Berkeley DB-backed queue system with soft/hard quota management, memory-mapped I/O files for frame buffering, a multi-protocol server with three distribution modes (NBS1, NBS2, EMWIN), configurable accept/reject regex filtering at four pipeline stages, and a Tcl configuration language with hierarchical override loading from defaults/ and site/ directories.

2. Daemon Configuration Architecture (nbspd.conf)

2.1 Multicast Reception Parameters

The source reveals four default multicast groups with configurable interface binding:

```
set multicastip   "224.0.1.1 224.0.1.2 224.0.1.3 224.0.1.4"
set multicastport "1201 1202 1203 1204"
```

Interface selection supports both name and IP address binding, with kernel routing table fallback:

```
set ifname ""     # e.g., "xl0", "fxp0"
set ifip   ""     # e.g., "192.168.12.100"
```

The UDP receive buffer size (udprcvsize) is platform-dependent: FreeBSD defaults to 65 KB (max 224 KB), Linux defaults to ~86 KB (max 262 KB), and Windows can be as low as 8 KB. The source recommends 256*1024 for Windows. A value ≤ 0 retains the OS default.

2.2 Memory I/O and Frame Buffering

Incoming SBN frames are buffered using memory-mapped I/O files before processing:

```
set memfileblocksize  2048     # ~average frame size
set memfileminsize    131072   # 128KB initial allocation
```

The blocksize is tuned to average frame sizes, while the minimum allocation is sized so most products fit without reallocation. This is a performance-critical path—the source comment notes these parameters directly impact throughput.

2.3 Berkeley DB Queue System

The daemon uses Berkeley DB for persistent queue management with configurable soft and hard quotas:

```
set queuemaxsoft    4096    # warning threshold (elements)
set queuemaxhard    0       # absolute max (0 = disabled)
set qrepquotaperiod 60      # quota report interval (seconds)
```

Each queue element occupies approximately 20 + strlen(full_path) bytes, roughly 80 bytes total. The processor control (pctl) subsystem has its own independent quota system with both element and memory-based limits:

```
set pctlmaxsoft      2048    # element soft limit
set pctlmaxhard      0       # element hard limit
set pctlmaxmemsoft   128     # memory soft limit (MB)
set pctlmaxmemhard   0       # memory hard limit (MB)
set pctlsyncperiod   3600    # DB sync interval (seconds)
```

The DB infrastructure lives in /var/noaaport/nbsp/db/ with a 16 MB memory cache (dbcachemb) and 0644 file permissions. Queue files are named q.db, processor control is pctl.db and pctlmf.db, and the spool tracking database is nbsp.db.

2.4 Four-Stage Accept/Reject Filtering

The source code reveals a sophisticated four-stage regex filtering pipeline, each with independent accept and reject patterns:

Stage	Variables	Controls
1. Reception	`acceptpatt / rejectpatt`	What products are saved to spool from the multicast stream
2. Filter Queue	`filterqacceptpatt / filterqrejectpatt`	What products are dispatched to the filter pipeline
3. Server Queue	`serverqacceptpatt / serverqrejectpatt`	What products are queued for network server distribution
4. Distribution	`nbsacceptpatt/nbsrejectpatt, emwinacceptpatt/emwinrejectpatt`	Per-protocol control of what clients receive

The algorithm is documented in the source: (1) If acceptpatt is NULL, all products are candidates. If it's an empty string, nothing is accepted. If non-null and non-empty, only matches pass. (2) Candidates are then checked against rejectpatt—NULL or empty means all pass, otherwise matches are rejected. The special string "*" matches everything, "" matches nothing. Patterns can be loaded from external files using Tcl exec:

```
set acceptpatt [exec cat "/usr/local/etc/nbsp/noaaport.accept"]
```

3. Server Module Architecture

3.1 Multi-Protocol Server

The servertype variable controls which distribution protocol(s) are active:

```
set servertype 0     # 0=none, 1=NBS1, 2=NBS2, 4=EMWIN
```

Values can be combined (e.g., 5 = NBS1 + EMWIN). NBS1 transmits the full file content (similar to EMWIN), while NBS2 transmits only the file path information (identical to what filters receive on stdin, requiring NFS for file access). EMWIN mode (type 4) uses the QBT “byte blaster” protocol. The server listens on a configurable port (default 1000) with support for unlimited clients (maxnetclients = -1).

3.2 Timing and Timeout Parameters

The source reveals five distinct timeout parameters controlling different I/O paths:

Parameter	Default	Purpose
`rdbroadcastimeout`	120s	Seconds waiting for multicast data before error logging
`wrfifotimeout`	500ms	Timeout writing to a filter FIFO device file
`wrclienttimeout`	1000ms	Timeout writing to a network client socket
`rdqueuetimeout`	1000ms	Timeout reading from filter/server queues
`rdprocqueuetimeout`	1000ms	Timeout reading from the processor queue

3.3 Slave Mode Configuration

The source defines three feed modes and client reconnection handling:

```
set slavemode  0     # 0=master, 1=NBS slave, 2=fpath slave

set mastername ""    # required when slavemode > 0

set masterport 1000

set rdslavetmout      10   # read timeout (seconds)

set reopenslavetmout  2    # reconnect wait (seconds)
```

Mode 1 (NBS slave) receives entire file contents over the network. Mode 2 (fpath slave) receives only file paths, requiring NFS-mounted spool access. The inoaaport wiki documents additional reconnection logic: the server preserves a disconnected client’s queue state for a configurable number of retry cycles (client_reconnect_wait_sleep_secs and client_reconnect_wait_sleep_retry), resuming transmission at the suspension point if the client reconnects within that window.

4. Spool and Retransmission Database

4.1 Spool File Management

Product files are written with configurable permissions and compressed storage:

```
set spooldir      "/var/noaaport/nbsp/spool"

set productmode   0644

set subdirmode    0755
```

The savezacceptpatt parameter controls on-the-fly compression: files matching "_(sdus[2357]|tig)" (radar data, satellite images) are saved in compressed form if transmitted compressed. The source notes that decompression is better deferred to the filter post-processing phase.

4.2 Retransmission Tracking Database

A Berkeley DB tracks received files to detect and skip retransmissions:

```
set nbspdbfname        "nbsp.db"

set nbspdbacceptpatt   # default: monitor everything

set nbspdbrejectpatt
```

The source documents the performance trade-off: DB operations are expensive, so monitoring all files can degrade performance. The recommended approach is to filter monitoring to products where reprocessing cost exceeds DB overhead. For example, set nbspdbacceptpatt {\+(grib|bufr)} monitors only GRIB and BUFR retransmissions. The default inserts everything, preventing duplicate processing when the original was received successfully.

5. Filter Pipeline Source Code

5.1 Filter Input Protocol

The filter stdin protocol, documented in the source with this exact example:

```
88164933 4 1 17 /var/noaaport/nbsp/spool/KMOB/kmob_sdus54-n0rmob.498745
```

Fields: SBN sequence number (88164933), type (4), category (1), code (17), and the full spool file path. The file path encodes the station ID (KMOB) in the directory and the WMO/AWIPS identifiers in the filename.

5.2 Reference Perl Filter Implementation

The source includes a complete working Perl filter example:

```
if(open(OUT, ">/var/noaaport/list.log") == undef){

    syslog("err", "Could not open logfile. $!");

    exit;

}

while($line = <STDIN>){

    chop($line);

    ($seq, $type, $cat, $code, $fpath) = split(/\s+/, $line);

    $dirname = dirname($fpath);

    $fname = basename($fpath);

    print(OUT "$seq $fname\n");

}
```

This reveals the filter contract: read lines from STDIN in a loop, split on whitespace to extract the five fields, then process the referenced file. Filters are long-lived processes—the while loop runs for the daemon's lifetime.

5.3 Filter Activation Mechanisms

Two mechanisms from the source code:

```
# Explicit list (colon-separated executables):

set filterlist "/usr/local/libexec/nbsp/rstfilter:\

                /usr/local/libexec/nbsp/dafilter:\

                /usr/local/libexec/nbsp/gpfilter"


# Auto-discovery directory (hot-deploy via SIGHUP):
```

```
  set devdir "/var/noaaport/nbsp/dev"
```

The emwin filter has special lifecycle handling: the emwinfilteralways flag (default 1) determines whether the emwin/rst filters run even when no clients are connected. Setting it to 0 skips filter invocation when no clients exist—an optimization for server-only deployments.

6. Tcl Configuration Override System

6.1 Hierarchical Override Loader

The source code at the bottom of nbspd.conf reveals a Tcl-based hierarchical configuration override system:

```
set nbspdconfdir       "/usr/local/etc/nbsp"
set nbspdlocalconfdirs [list $nbspdconfdir/defaults \
                             $nbspdconfdir/site]

set _confname "nbspd.conf"
foreach _d $nbspdlocalconfdirs {
    set _localconfname ${_d}/${_confname}
    if {[file exists ${_localconfname}] == 1} {
        source ${_localconfname}
    }
}
unset _d; unset _confname; unset _localconfname
```

This implements a three-layer configuration precedence: (1) compiled-in defaults in the C daemon, (2) defaults/ directory overrides, (3) site/ directory overrides. The site/ layer survives package upgrades because the package manager only touches the base config, not the site/ directory. The Tcl source command executes each override file in the current interpreter context, so later files override earlier ones. The careful unset of temporary variables prevents namespace pollution.

6.2 dafilter.conf.in Source Analysis

The 211-line dafilter.conf.in reveals the filter's Tcl configuration structure using associative arrays:

```
set dafilter(datadir)    [file join $common(datadir) "digatmos"]
set dafilter(invdir)     [file join $common(datainvdir) "digatmos"]
set dafilter(invformat)  $common(datainvformat)
set dafilter(radinvsubdir)  "nexrad"
set dafilter(satinvsubdir)  "sat"
set dafilter(rad_inv_enable) 1
set dafilter(sat_inv_enable) 1
```

The NNTP gateway configuration within the same filter:

```
  set dafilter(nntp_enable)      0

  set dafilter(nntp_server)      "news"

  set dafilter(nntp_from)        "nbspfeed@noaaport.net"

  set dafilter(nntp_groupprefix) "noaaport.data"

  set dafilter(hourlyfilefmt)    "%Y%m%d%H"
```

The filter supports a "lite" configuration mode via an alternate rc file:

```
  set dafilter(localrc)  "dafilter-lite.rc"
```

A common(datadir) variable provides shared path roots across all filters, demonstrating that the filter ecosystem shares a Tcl namespace for cross-cutting configuration.

7. Operational Source Files

7.1 logrotate.conf (61 lines)

The logrotate source defines rotation for 10 distinct file groups:

```
/var/noaaport/nbsp/stats/nbspd.status  { missingok daily rotate 7 }

/var/noaaport/nbsp/stats/nbspd.missing  { missingok daily rotate 7 }

/var/noaaport/nbsp/stats/nbspd.rtx      { missingok daily rotate 7 }

/var/noaaport/nbsp/stats/nbspd.qstate   { missingok daily rotate 7 }

/var/noaaport/nbsp/stats/nbspd.server   { missingok daily rotate 7 }

/var/noaaport/nbsp/stats/nbspd.sthreads { missingok daily rotate 7 }

/var/noaaport/nbsp/stats/nbspd.filter   { missingok daily rotate 7 }

/var/noaaport/nbsp/stats/nbspd.slavestats { missingok daily rotate 7 }

/var/log/noaaport/*log { missingok daily compress rotate 2 }
```

The eight stats files map directly to internal daemon subsystems: reception status, missing frame tracking, retransmission monitoring, queue state, server module, server threads, filter pipeline, and slave replication. Decoder logs are compressed (space optimization) with shorter 2-day retention.

7.2 hourly-cleanup.conf Source

The cleanup script uses a custom DSL for time-based file management:

```
H/6:/var/noaaport/data/inv/gis/sat: -i -mtime +6h

H=01:/var/log/nbsp/www: -mtime +7

H=23:/var/noaaport/nbsp/spool: -mtime +1
```

The format is H[=hour|/interval]:path: find-args. H=23 means “run at 23:00 only,” H/6 means “run every 6 hours.” The -i flag and find arguments are passed directly to the find command. The source comments document the spooldbslots interaction: when non-zero (default), the daemon self-manages the spool, and the H=23 rule serves only as a safety net for orphaned files.

8. Internal State and Monitoring

8.1 Statistics Generation

The daemon writes statistics at a configurable interval:

```
set statsperiod  60   # recompute every 60 seconds

set statusfile   "/var/noaaport/nbsp/nbspd.status"
```

The statefifo provides a real-time named pipe interface to the daemon's internal state:

```
set statefifo     "/var/noaaport/nbsp/nbspd.fifo"

set statefifomode 0644
```

This FIFO enables external monitoring tools to read the daemon's state in real-time without polling the statistics files. The 0644 mode allows any user to read the state.

8.2 Process Management

```
set pidfile  "/var/run/nbspd.pid"

set umask    022
```

Standard Unix daemon conventions: PID file for process management and signal delivery, umask 022 for group-readable file creation.

9. Repository Structure as Code

The repository file tree (from the GitHub source browser) reveals the code organization across 20 top-level directories:

Directory	Language	Code Content
src/	C	Core daemon: nbspd main loop, multicast reception, frame reassembly, spool DB, queue management, server module, memory-mapped I/O
filters/	Tcl	Standard filter implementations with per-filter subdirs (dafilter/tcl/, gpfilter/, rstfilter/, etc.) containing .conf.in templates and .rc rule files
conf/	Tcl	Platform configs (nbspd.conf-linux 42 lines, nbspd.conf-freebsd 43 lines), features.conf, logrotate.conf (61 lines), nbspd.conf master
scripts/	Shell	Operational automation: hourly-cleanup.conf (platform variants), start/stop scripts, log rotation
tclhttpd/	Tcl/HTML	Full TclHttpd 3.5.1 source (src/tclhttpd3.5.1/) with htdocs/ web content and .tml server-side templates
tclgempak/	Tcl/C	GemPak bindings: Tcl wrappers around C GemPak decoder libraries
tclgrads/	Tcl	GrADS bindings with test suite (src/lib/test/) including test-output validation files
tclmetar/	Tcl	METAR parser: aviation weather observation text format decoder
tclupperair/	Tcl	Upper air sounding data processor
tclldm/	Tcl/C	LDM bridge (nbsp2ldm): Tcl/C interface to Unidata product queues
tclssh/	Tcl	SSH library for secure remote file distribution
build/	Shell/M4	Platform packaging: bsdng/ (FreeBSD), deb/ (Debian/Ubuntu), rpm/ (RHEL/CentOS)
dev-notes/	Text	dictionary.txt (6,497 lines/305 KB), stations.txt (8,749 lines/679 KB), WMO product definitions
examples/	Tcl/Cfg	Client integration: ondas/ (GRLevel3 configs with 150+ NEXRAD sites), Digital Atmosphere scripts
tools/	C/Shell	Utility programs for data inspection and debugging
utils/	C/Shell	Helper programs and shared utilities

10. Key Source Code Observations

- **CVS Heritage:** The $Id tag (v1.40, 2006/02/28) reveals the project's CVS version control origins before its migration to Git on Bitbucket (~2011) and later GitHub. The code predates modern build tools.
- **Berkeley DB as Queue Backend:** The use of Berkeley DB for persistent queue management (with named files q.db, pctl.db, pctlmf.db, nbsp.db) provides crash recovery and durability. The dual soft/hard quota system with separate element and memory limits shows careful capacity planning.
- **Memory-Mapped I/O:** The memfileblocksize/memfileminsize parameters reveal a custom memory-mapped file allocator for frame buffering, tuned for the average NOAAPort frame size (~2 KB) with 128 KB pre-allocation.
- **Four-Stage Regex Filtering:** Independent accept/reject patterns at reception, filter queue, server queue, and distribution stages provide fine-grained control over data flow. The NULL vs empty string vs pattern distinction is a precise three-state logic.
- **Tcl as Configuration Language:** The hierarchical override loader (defaults/ then site/) with Tcl source is a configuration management pattern that predates modern tools like Ansible's variable precedence. The filter config namespace (dafilter(), common()) demonstrates Tcl associative arrays as structured configuration.
- **Dual Filter Activation:** The filterlist + devdir mechanism provides both explicit control and hot-deploy flexibility. SIGHUP reloads only the devdir, not the filterlist, separating stable from experimental filters.
- **NBS2 Path-Only Protocol:** Server type 2 transmits only file paths (not contents), requiring NFS—an elegant optimization for same-network deployments that eliminates duplicate data transmission.
- **State FIFO Interface:** The nbspd.fifo named pipe provides a real-time monitoring interface separate from the periodic statsfile writes, enabling zero-latency state queries from external tools.

11. Conclusion

Reading Nbsp through its source code—rather than its documentation or README—reveals an engineering depth that descriptive text cannot convey. The Berkeley DB queue system with dual element/memory quotas, the four-stage regex filtering pipeline, the memory-mapped I/O frame allocator, and the Tcl hierarchical configuration loader are architectural decisions visible only in the code itself.

The nbspd.conf source file alone, at approximately 300 lines of densely commented Tcl, defines over 50 configurable parameters across six subsystems (reception, buffering, queuing, filtering, serving, and slave replication). Each parameter documents its default value, platform-specific behavior, and performance implications in inline comments—making the configuration file itself the most authoritative architectural reference.

The code tells the story of a system designed by an operator: the retransmission database with its configurable monitoring scope (trading DB overhead against duplicate processing cost), the spooldbslots safety-net cleanup, the emwinfilteralways optimization, and the client queue state preservation across reconnections are all features that emerge from running the system in production and solving real problems.