```
/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++

Copyright (c) 2001 BeComm Corporation

Filename:

    timesync.c

Group Name:

    todo

Group Overview:

    Uses a broadcast protocol to determine the
    clock offsets of all listening peers.

    Algorithm is based loosely on NTP.

    Also has edges which are used for passing
    sample clocks across the network in a path.

Owner:

    Guy Carpenter (guyc) 16-Aug-2001

----------------------------------------------------------------------
----*/
#define SOS_DEBUG_ZONE "/beads/timesync"
#include <sosstrings.h>
#include <sosmultimedia.h>
#include "timesync.h"

SOS_SOURCE_VERSION("$Id: timesync.c,v 1.14 2001/10/23 16:40:49 guyc Exp
$");

/*
 * Broadcast sync packets every TIMER_INTERVAL milliseconds
 */
#define TIMER_INTERVAL     4000
#define TIMER_INITIAL_DELAY 200
#define MAX_AGE           60000  /* expire after a minute */

/*
```

```
 * Compute offset based on WINDOW_SIZE observations
 */
#define WINDOW_SIZE 8

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++
Named Constants
--------------------------------------------------------------------------
----*/

/*
 * Name of bead
 */
static
const char BEAD_NAME[]   = "timesync";



/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++
Globals
--------------------------------------------------------------------------
----*/

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++
Structs
--------------------------------------------------------------------------
----*/

typedef struct {
    SOS_UINT32       HostId;
    SOS_CLOCK_TICK   XmitTime;
    SOS_CLOCK_TICK   RecvTime;
    SOS_INT32        Offset;
    SOS_BOOLEAN      OffsetValid;
    SOS_INT32        Min[WINDOW_SIZE];
    SOS_INT32        Max[WINDOW_SIZE];
    SOS_UINT32       WindowFill;
    SOS_UINT32       WindowPtr;
} HOSTINFO;

typedef struct _UPDATE_HEADER {
    SOS_UINT32      HostId;
    SOS_CLOCK_TICK  XmitTime;
```

```
} UPDATE_HEADER;

typedef struct {
    SOS_UINT32      HostId;
    SOS_CLOCK_TICK  XmitTime;
    SOS_CLOCK_TICK  RecvTime;
} HOSTSYNC;

typedef struct UPDATE_CONTEXT {
    SOS_POINTERTABLE * HostTable;
    SOS_LOCK *         HostTableLock;
    SOS_UINT32         HostId;
    SOS_PATH *         Path;
    SOS_TIMER_ID       TimerId;
} UPDATE_CONTEXT;

#define   LOCK() SOS_Lock_Acquire(g_Context->HostTableLock)
#define UNLOCK() SOS_Lock_Release(g_Context->HostTableLock)


/*
 * Maintain a single global context, since this is a system-wide
protocol.
 */
static
UPDATE_CONTEXT * g_Context;

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++

------------------------------------------------------------------------
----*/

static
HOSTINFO *
TimeSync_HostInfoFind(
    SOS_UINT32       HostId
)
{
    HOSTINFO *host = NULL;

    SOS_PointerTable_ElementPeek(
        g_Context->HostTable,
        (void*)HostId,
```

```
124     );
125 
126     return host;
127 }
128 static
129 HOSTINFO *
130 TimeSync_HostInfoFindOrAdd(
131     SOS_UINT32       HostId
132 )
133 {
134     HOSTINFO *host = NULL;
135 
136     host = TimeSync_HostInfoFind(HostId);
137 
138     if (!host) {
139         SOS_DEBUGOUT_MAJOR_EVENT(
140             "Adding host %x to table\n",
141             HostId
142         );
143         host = SOS_Mem_Alloc(sizeof(*host));
144         SOS_memset(host, 0, sizeof(*host));
145         host->HostId = HostId;
146         SOS_PointerTable_ElementPut(
147             g_Context->HostTable,
148             (void*)HostId,
149             host
150         );
151     }
152 
153     return host;
154 }
155 
156 static
157 void
158 TimeSync_Expire(
159     UPDATE_CONTEXT *      Context,
160     SOS_UINT32            MaxAge
161 )
162 {
163 
164     SOS_POINTERTABLE_ITERATOR *iterator;
165     HOSTINFO *hostInfo;
166     SOS_CLOCK_TICK now = SOS_Clock_TickGet();
167 
```

```
169
170     LOCK();
171
172     SOS_PointerTableIterator_Create(
173         Context->HostTable,
174         &iterator
175     );
176
177     while (SOS_Success==SOS_PointerTableIterator_Next(
178                iterator,
179                NULL,
180                (void**)&hostInfo
181            )) {
182         SOS_CLOCK_TICK age = now - hostInfo->RecvTime;
183
184         if (age>MaxAge) {
185
186             SOS_DEBUGOUT_MAJOR_EVENT(
187                 "Removing host %x (age %lu) from table\n",
188                 hostInfo->HostId,
189                 age
190             );
191
192             SOS_PointerTable_ElementRemove(
193                 Context->HostTable,
194                 (void*)hostInfo->HostId,
195                 NULL
196             );
197
198             {
199                 /*
200                  * This ugliness works around the fact
201                  * that the table API does not allow us
202                  * to destroy a member during iteration.
203                  * So each time we remove something we
204                  * start again.
205                  */
206                 SOS_PointerTableIterator_Destroy(
207                     iterator
208                 );
209                 SOS_PointerTableIterator_Create(
210                     Context->HostTable,
211                     &iterator
212                 );
```

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.