Title
EmStar-2: The Next Generation Programming Development Environment for 32-bit Class of Embedded Devices (SYS 9)

Permalink
https://escholarship.org/uc/item/2kf8m371

Authors
Vinayak Naik
Lewis Girod
Martin Lukac
et al.

Publication Date
2006
EmStar-2: The Next Generation of Programming Development Environment for 32-bit Class of Embedded Devices

Vinayak Naik¹, Lewis Girod², Martin Lukac¹, Nithya Ramanathan¹, Ben Greenstein¹, Eddie Kohler¹, and Deborah Estrin¹
¹CENS–UCLA and ²CSAIL–MIT
http://research.cens.ucla.edu

Introduction: EmStar facilitates WSN software development for 32-bit platform

Role of EmStar in WSN

- A development platform for 32-bit class of embedded devices
  - EmStar provides a programming development environment for building robust WSN systems for 32-bit class of embedded devices

Unhandled issues in the current generation of EmStar

- Multiple operating systems for 32-bit platforms
  - All though, Linux is the most widely used operating system, Windows is used by certain devices such as smartphones
- Incorrect memory access resulting in run-time failure
  - Given the scarcity of memory, lack of garbage collection increases the chances run-time crash
- Issues with multiple processes
  - It is complex to write processes, which can deal with restart of other processes; e.g. dealing with restarted time-sync process
  - For large-scale deployment, multiple processes in each node makes system management complex

Proposed Solution: New design principles for EmStar-2

Python as a programming language

<table>
<thead>
<tr>
<th>Garbage collection</th>
<th>Portability across OS</th>
<th>Handling of C structures</th>
<th>Code readability</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>C#</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Java</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Perl</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Python</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

- Asynchronous (non-blocking) process communication
  - A single thread of execution can communicate with multiple processes, e.g. visibility and control tools
  - No issues related to locking, deadlock, or synchronization found in multi-threaded system
  - Use of GLIB tools for generating and handling events

```python
s = socket.socket(socket.AF_UNIX, socket.SOCK_STREAM)
s.connect((FILENAME))
p = select.poll()
p.register(s.fileno(), select.POLLIN | select.POLLHUP)
while 1:
    results = p.poll(1)
    if len(results):
        if results[0][1] == select.POLLIN:
            data = s.recv(8)
        if not len(data):
            print("Remote end closing connection; exiting.")
            break
        print "Received: ", data
        elif (results[0][1] == select.POLLHUP):
            print "Server hanged up; exiting."
            sys.exit(0)
        else:
            print "Problem occurred; exiting."
            sys.exit(0)
```

- Use of exception-handling to reduce run-time crashes
  - In case of erroneous condition, program catches the error gracefully rather than crashing
  - A traceback of function calls is provided to user facilitate debugging

Ability to use devices generated by existing EmStar code

- EmStar-2 code can use all the device files generated by existing EmStar code, hence leveraging on large code-base of existing tools and utilities

Problem Description: Simplify software development and large-scale deployment

- Increasingly involved role for 32-bit platforms in WSN
  - Traditionally, 32-bit platforms serve as micro-server or master in large-scale mote networks
  - Recently, 32-bit platforms are themselves used as sensor nodes, e.g. MASE deployment in Mexico for seismic sensing

- New design principles for seismic sensing deployment in Mexico for nodes, e.g. MASE
  - Asynchronous (non-blocking) process communication
  - Use of GLIB tools for generating and handling events

- Use of exception-handling to reduce run-time crashes
  - In case of erroneous condition, program catches the error gracefully rather than crashing

- Ability to use devices generated by existing EmStar code
  - EmStar-2 code can use all the device files generated by existing EmStar code, hence leveraging on large code-base of existing tools and utilities