aselsan

Node.JS Appliances on Embedded Linux Devices

Mehmet Fatih Karagöz & Cevahir Turgut

Outline

- Introduction to Node.js
- Cross-compiling Node.js and Node Package Manager(NPM)
- Development environment
- Scripting samples in embedded devices
- Development story of a surveillance application
- Demo
- Questions

Outline

- Introduction to Node.js
- Cross-compiling Node.js and NPM
- Development environment
- Scripting samples in embedded devices
- Development story of a surveillance application
- Demo
- Questions

Introduction to Node.js

- What is Node.js?
  - Node.js is a platform built on Chrome's JavaScript runtime (V8) for easily building fast, scalable network applications.

- Event-Driven
- Non-Blocking I/O
- Lightweight
- Efficient HW Usage

Introduction to Node.js

Advantages

- Open source
- HW Efficiency
- Learning Curve
- Development Time
- Javascript / No Compile
- NPM Package Manager

Disadvantages

- Small Developer Pool
- Bad at CPU Bound Jobs
- Asynchronous Debugging

Introduction to Node.js

- What is learning curve?
  - Node.JS uses Javascript
  - Most web developers are familiar with the language
  - You can get started with building very basic application in less than one hour (that includes installation time!)

Introduction to Node.js

- NPM (Node Packaged Modules)

- NodeJS package management system
- Install modules very easily even on embedded
  - "npm install express“
- Installs dependant modules too
- Global install option
  - "npm install -g express"

Introduction to Node.js

- Popular Modules
  - Express
  - Request
  - Async
  - Grunt
  - socket.io
  - Mocha
  - Underscore
  - Mongoose
  - Redis
  - Connect

Outline

- Introduction to Node.js
- Cross-compiling Node.js and NPM
- Development environment
- Scripting samples in embedded devices
- Development story of a surveillance application
- Demo
- Questions

Cross-compiling Node.js and NPM

- Where is Node.js?

- Download
  - http://nodejs.org/download
- Git
  - git clone git://github.com/joyent/node.git
  - cd node
  - git checkout v0.10

- Node.js is released under the MIT license.

Cross-compiling Node.js and NPM

- Configuration Options
  - Usage: configure [options]
  - Options:
    - --without-npm Don't install the bundled npm package manager
    - --without-ssl Build without SSL
    - --without-snapshot Build without snapshotting V8 libraries. You might want to set this for cross-compiling.
    - --dest-cpu=DEST_CPU CPU architecture to build for. Valid values are: arm, ia32, x64
    - --dest-os=DEST_OS Operating system to build for. Valid values are: win, mac, solaris, freebsd, openbsd, linux, android

Cross-compiling Node.js and NPM

- How to make and install
  - export AR=arm-linux-gnueabihf-ar
  - export CC=arm-linux-gnueabihf-gcc
  - export CXX=arm-linux-gnueabihf-g++
  - export LINK=arm-linux-gnueabihf-g++

  - ./configure --without-snapshot --dest-cpu=arm --dest-os=linux
  - make
  - make install DESTDIR=~/node-armhf/

Outline

- Introduction to Node.js
- Cross-compiling Node.js and NPM
- Development environment
- Scripting samples in embedded devices
- Development story of a surveillance application
- Demo
- Questions

Development Environment

- Suitable Editors
  - Vim
  - Gedit
  - Webstorm
  - Eclipse / Nodeclipse Plugin
  - Cloud9 (Cloud based editor)

Development Environment

- "node debug myscript.js"
- Debugging Options
  - cont, c - Continue execution
  - next, n - Step next
  - step, s - Step in
  - out, o - Step out
  - pause - Pause running code (like pause button in Developer Tools)

  - setBreakpoint(), sb() - Set breakpoint on current line
  - setBreakpoint(line), sb(line) - Set breakpoint on specific line
  - setBreakpoint('fn()'), sb(...) - Set breakpoint on a first statement in functions body
  - setBreakpoint('script.js', 1), sb(...) - Set breakpoint on first line of script.js
  - clearBreakpoint, cb(...) - Clear breakpoint

  - backtrace, bt - Print backtrace of current execution frame
  - list(5) - List scripts source code with 5 line context (5 lines before and after)
  - watch(expr) - Add expression to watch list
  - unwatch(expr) - Remove expression from watch list
  - watchers - List all watchers and their values (automatically listed on each breakpoint)
  - repl - Open debugger's repl for evaluation in debugging script's context

Outline

- Introduction to Node.js
- Cross-compiling Node.js and NPM
- Development environment
- Scripting samples in embedded devices
- Development story of a surveillance application
- Demo
- Questions

Scripting Samples in Embedded Devices

- GPIO/LED on/off

var exec = require('child_process').exec;

```
/* EXEC SYSTEM CALL */
var setLedOn = function(){
  exec('echo 1 > /sys/class/leds/d440\:red/brightness', function(){
    console.log('Led is on!');
  });
}
```

Scripting Samples in Embedded Devices

- CPU / Memory Usage

var os = require('os');

```
/* CPU USAGE */
var printCPULoad = function(){
  var load = os.loadavg();
  var message = "1 Min: %" + Math.floor(load[0]*100) + " 5 Min: %" + Math.floor(
  load[1]*100) + " 15 Min: %" + Math.floor(load[2]*100);
  console.log(message);
}
```

```
/* MEMORY USAGE */
var printMemLoad = function(){
  var freemem = os.freemem();
  var totalmem = os.totalmem();
  var message = "Free Mem: " + Math.floor(freemem/1000000000) + " Total Mem: " +
  Math.floor(totalmem/1000000000);
  console.log(message);
}
```

Scripting Samples in Embedded Devices

- Timers

```
/* TIMER EXAMPLE */
var val = 0;
var setLed = function(){
   var cmd = "echo " + val + " > /sys/class/leds/d440\:red/brightness";
   console.log(cmd);
   exec(cmd , function(){
        if(val === 0)
            val = 1;
        else
            val = 0;
        console.log('value: '+ val);
   });
}
setInterval(setLed, 1000);
```

Scripting Samples in Embedded Devices

- Ini Parser

```
/* INI PARSER using node-iniparser module */
var iniparser = require('iniparser');
iniparser.parse('./config.ini', function(err,data){
    var version = data.version;
});
```

config.ini:

```
;sample config ini file
name = Iniparser Demo
version = 0.1
```

“npm install iniparser”

Outline

- Introduction to Node.js
- Cross-compiling Node.js and NPM
- Development environment
- Scripting samples in embedded devices
- Development story of a surveillance application
- Demo
- Questions

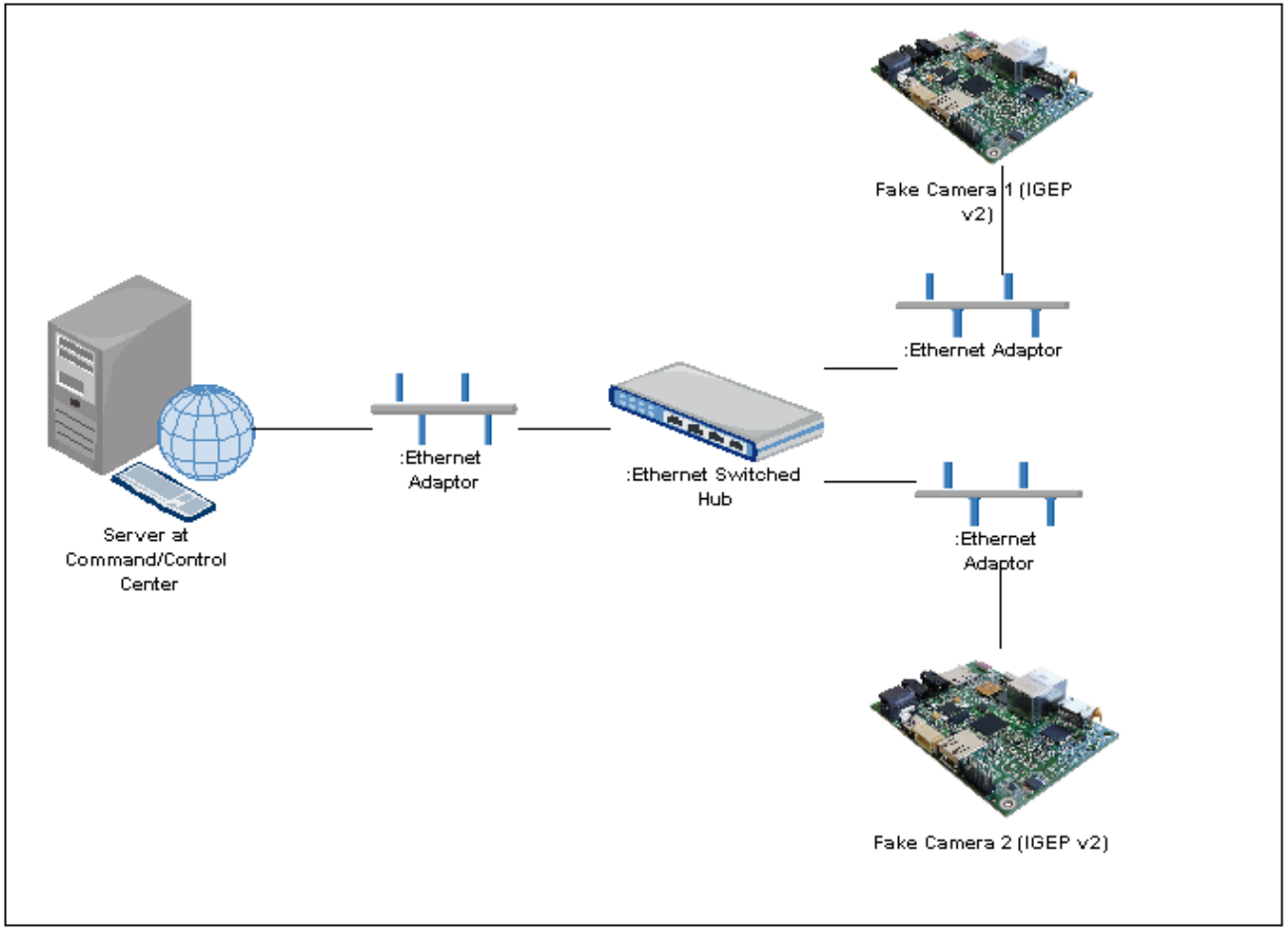
Development Story of a Surveillance Application - Overview

- Main Purpose
  - Demonstrate usage scenario of Node.js in real world app
  - Give more information about network classes of Node.js
  - Run complete Node.js app in embedded platform

- Example Surveillance System
  - Smart IP Cameras
    - to detect motion, etc.
  - Server PC
    - at Command Control Center
    - monitoring cameras, their alarms, etc.

Development Story of a Surveillance Application - Overview

- Deployment Model of Example System



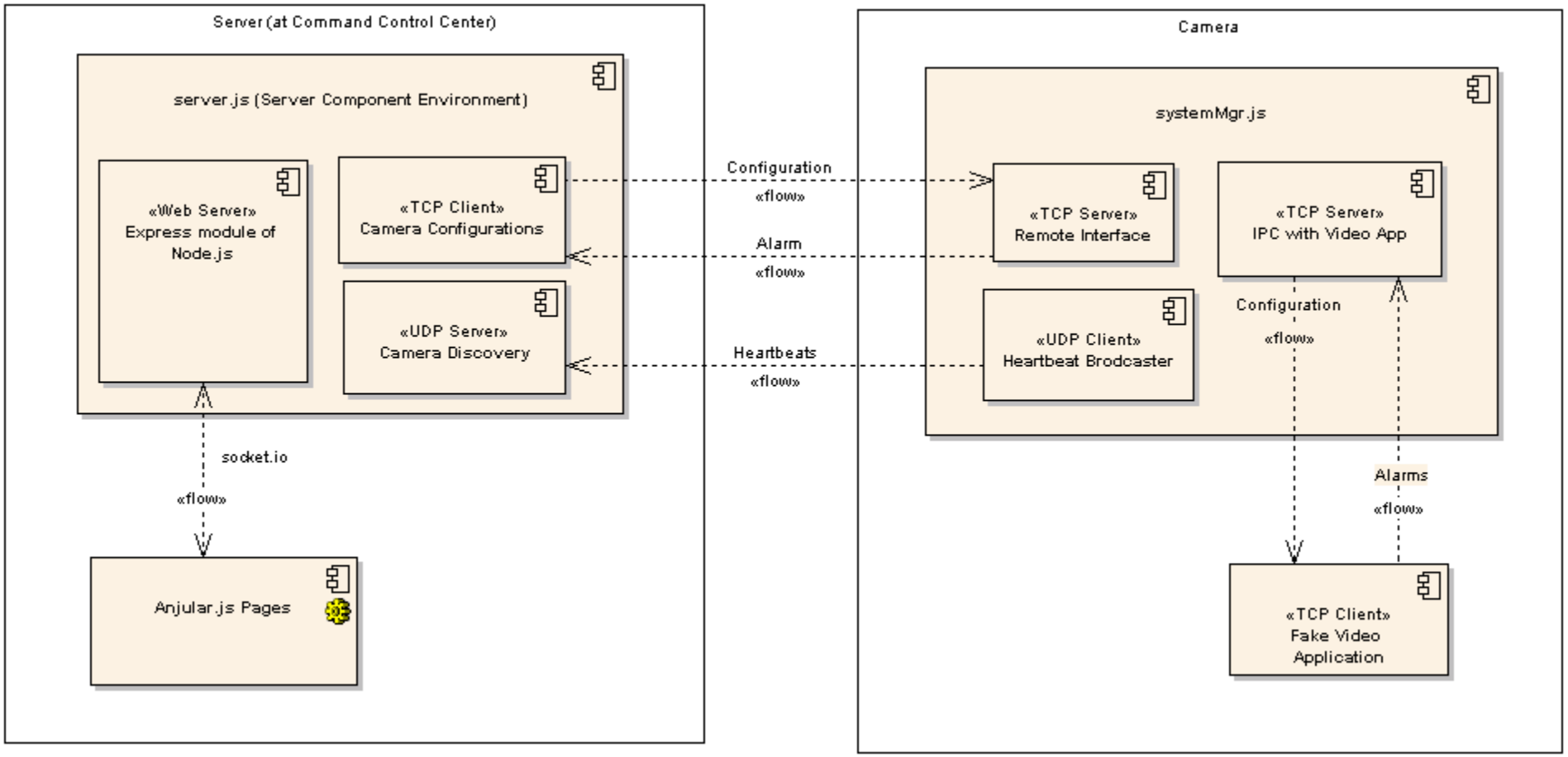
aselsan

Development Story of a Surveillance Application - Overview

- Used Components
  - IGEPv2 Boards
    - DM3730 Texas Instruments processor
    - ARM Cortex A8 1GHz
    - Camera ISP
    - 512 Megabytes RAM / 512 Megabytes FLASH
    - Ethernet 10/100 MB BaseT
    - And more (C64+ DSP 800MHz, 3D Accelerator SGX530 @ 200 MHz, ...)
  - Kernel
    - 2.6.37
  - Compiler:
    - GCC version 4.6.3

Development Story of a Surveillance Application - Overview

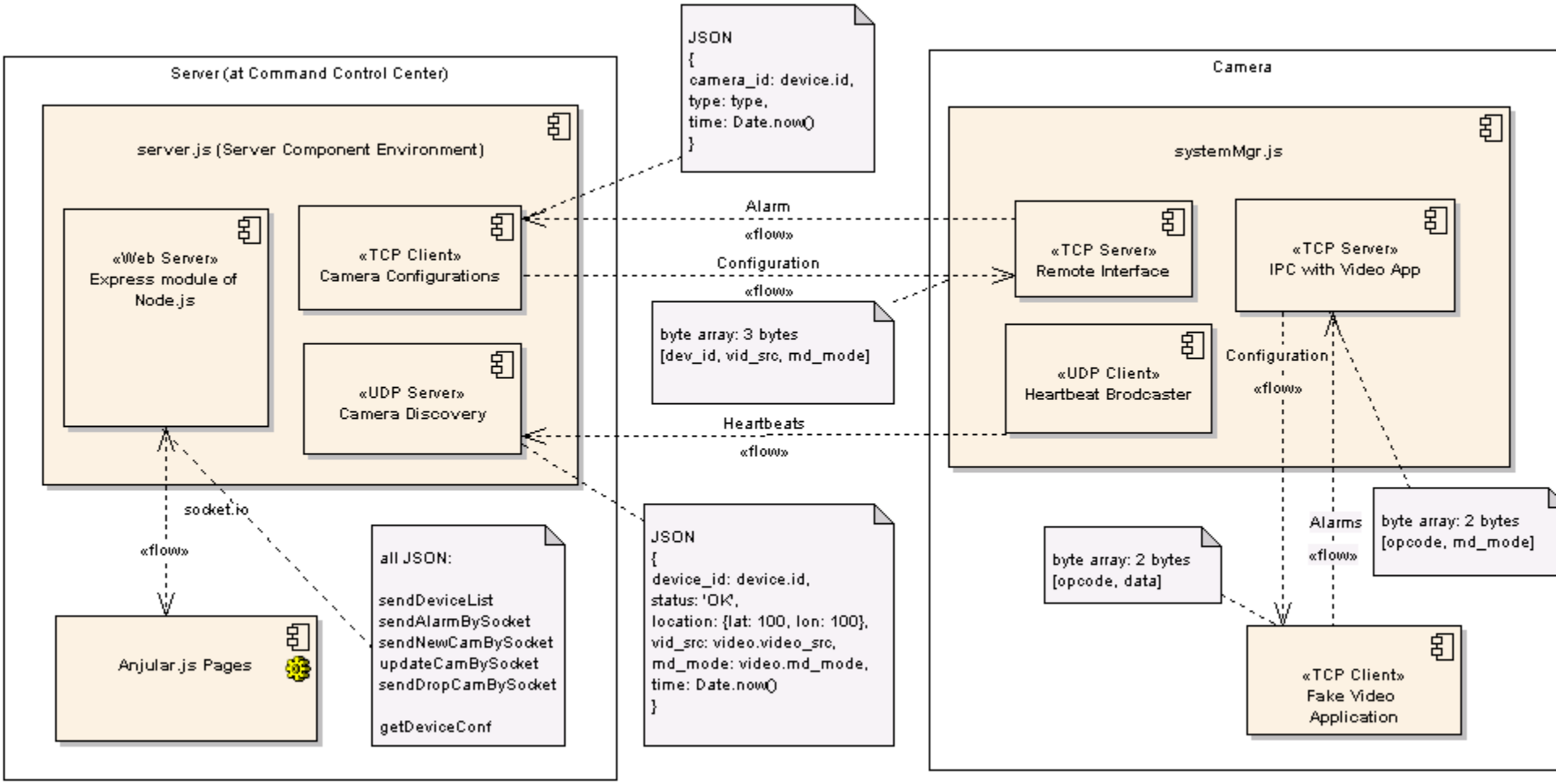
- Component Diagram



aselsan

Development Story of a Surveillance Application - Overview

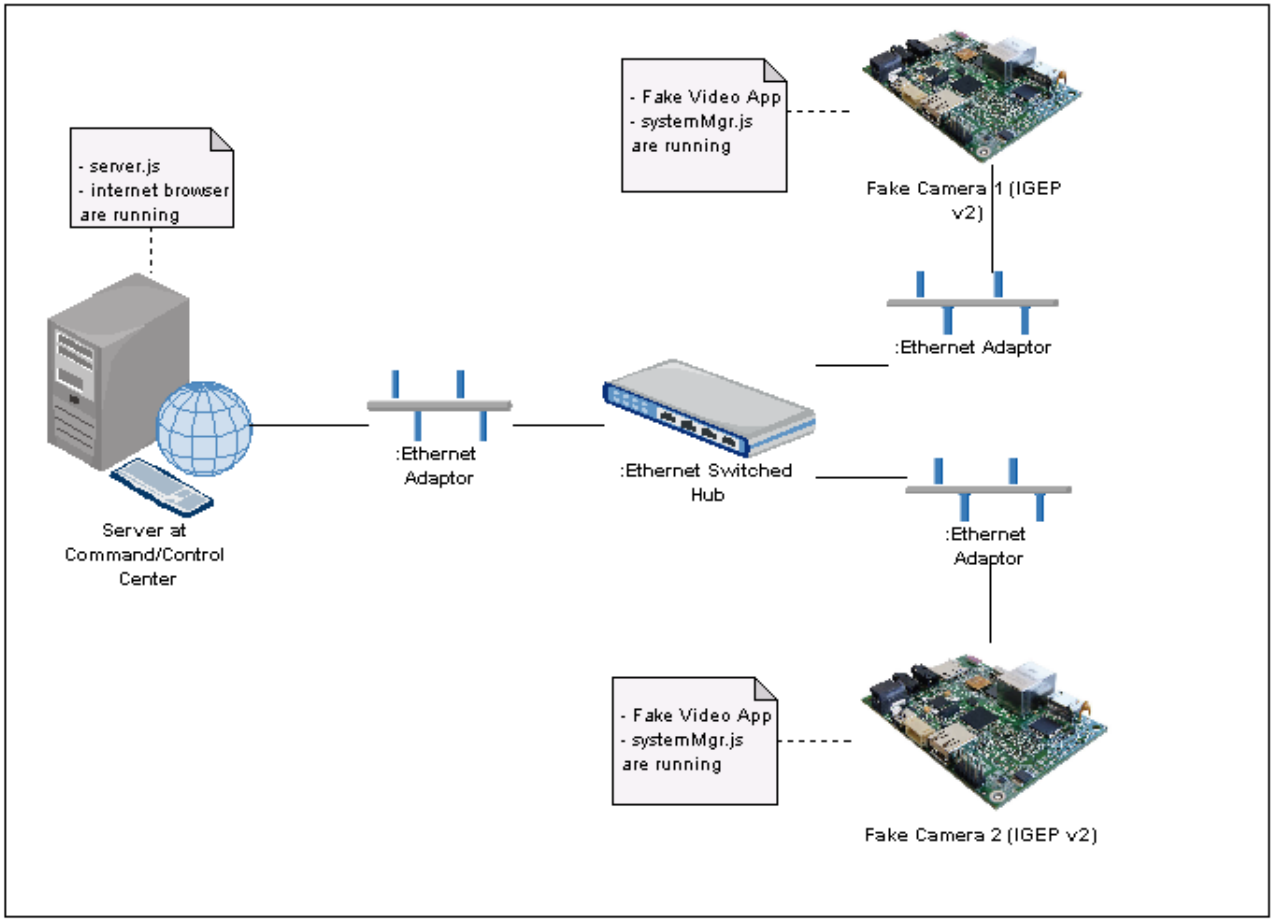
- Component Diagram (with messages)



aselsan

Development Story of a Surveillance Application - Overview

- Deployment Model of Example System



aselsan

Development Story of a Surveillance Application - Video App

- Video App
  - Fake Applicaton to simulate motion detection
  - Written using Boost CPP Libraries
  - Connected to Camera System App with TCP
  - Produces periodic alarm messages
  - Receives configuration messages

Development Story of a Surveillance Application - Video App

```
119     t.async_wait( boost::bind(periodicTimer, boost::asio::placeholders::error, &t) );
120     boost::thread thread1(boost::bind(&boost::asio::io_service::run, &io));
121     //Start Fake Video App
122     while (1) {
123         read_length = boost::asio::read((socket_), boost::asio::buffer((char*) &(buffer[0]), (size_t) 2),
124                 boost::asio::transfer_at_least(2), ec);
125         if (!ec && read_length == 2) {
126             cout << "** VideoAppMain: configuration message is received" << endl;
127             opcode = buffer[0];
128             data = buffer[1];
129             switch (opcode) {
130             case OPCODE_VIDEO_SRC_CHG:
131             {
150             case OPCODE_MD_MODE_CHG:
151                 if (data == MD_MODE_OFF) {
152                     md_mode = MD_MODE_OFF;      cout << "** VideoAppMain: MD Mode: OFF ** " << endl;
153                 } else if (data == MD_MODE_1) {
154                     md_mode = MD_MODE_1;        cout << "** VideoAppMain: MD Mode: 1 ** " << endl;
155                     alarm_period = 10;
156                 } else if (data == MD_MODE_2) {
157                     md_mode = MD_MODE_2;        cout << "** VideoAppMain: MD Mode: 2 ** " << endl;
158                     alarm_period = 30;
159                 }
160                 //Send ACK
161                 buffer[0] = OPCODE_MD_MODE_CHG;
162                 buffer[1] = (char) md_mode;
163                 write_length = boost::asio::write((socket_),
164                         boost::asio::buffer(buffer, 2),
165                         boost::asio::transfer_all(), ec);
166
```


aselsan

Development Story of a Surveillance Application - Camera SystemMgr App

- Camera SystemMgr App
  - Written using Node.js
  - Manages other software elements running on camera through IPC
    - Store configurations
  - Connection interface of camera to server
  - Includes 2 TCP servers and 1 UDP client
    - 1 TCP Server for IPC with VideoApp
    - 1 TCP Server
      - to get configuration messages from Server
      - to send alarm messages to Server
    - 1 UDP broadcaster to send heartbeat

Camera SystemMgr App

```
29
30  /* VIDEO APP TCP SERVER STARTS: TCP Connection between Video App and Camera System App (IPC) */
31  var net = require('net');
32  var video = {video_src: 1, md_mode: 0, socket: null};
33  var video_app_server = net.createServer(function (socket) { //'connection' listener
34    console.log('Video app connected');
35      socket.on('data', function(data){   //Emitted when data is received
36      if(data.length === 2){
37          if(data[0] === VIDEO_SRC_OPCODE){   //ACK taken from VideoApp
38              console.log('Video source message received! Source: ' + data[1]);
39              video.video_src = data[1];      //Save state
40          }
41          else if(data[0] === VIDEO_MODE_OPCODE){
45          else if(data[0] === VIDEO_ALARM_OPCODE){    //Alarm taken from VideoApp
46              console.log('Alarm message received!');
47              if(device.tcp_socket != null){
48                  sendAlarm(device.tcp_socket, data[1]);  //Send alarm to Server
49              }
50          }
51          else{
54      }
55  });
56  socket.on('close', function(){    //Emitted once the socket is fully closed
57          video.socket = null;
58  });
59  video.socket = socket;    //save socket
60  });
61  video_app_server.listen(1337, '127.0.0.1');
62  console.log('Video app server listening to 127.0.0.1:1337');
63  /* VIDEO APP TCP SERVER ENDS */
```

Camera SystemMgr App

```
/* SYSTEM TCP SERVER STARTS: TCP Connection between Camera System App and Server */
function setVideoSrc(socket, src){
    var msg = new Buffer([VIDEO_SRC_OPCODE, src]);
    socket.write(msg);
}
function setMdMode(socket, mode){
    var msg = new Buffer([VIDEO_MODE_OPCODE, mode]);
    socket.write(msg);
}

var system_server = net.createServer(function (socket) {
  console.log('Server app connected');
  socket.on('data', function(data){
    if(data.length === 3 && data[0] === device.id){
        console.log('Conf Received!');
        if(video.socket != null){
            setVideoSrc(video.socket, data[1]); //Send configuration to VideoApp
            setMdMode(video.socket, data[2]);
        }
     }
  });
  socket.on('close', function(){
        device.tcp_socket = null;
  });
  device.tcp_socket = socket;
});

system_server.listen(1338);
console.log('System server listening to port 1338');
/* SYSTEM TCP SERVER ENDS */
```


aselsan

Camera SystemMgr App

```
97
98  /* UDP HEARTBEAT SENDER: UDP Broadcaster */
99  var dgram = require('dgram');
100 var client = dgram.createSocket("udp4");
101 client.bind(function(){
102     client.setBroadcast(true);  //enable broadcasting after socket binding
103 });
104
105 setInterval(function(){
106     var msg =  {device_id: device.id,
107                 status: 'OK',
108                 location: {lat: 100, lon: 100},
109                 vid_src: video.video_src,
110                 md_mode: video.md_mode,
111                 time: Date.now()};
112     var message = new Buffer(JSON.stringify(msg));  //Convert a value/an object  to JSON
113     client.send(message, 0, message.length, 1400, "255.255.255.255", function(err, bytes) {
114       console.log('heartbeat sent!');
115     });
116 }, 2000);
117 /* UDP HEARTBEAT ENDS */
```

Development Story of a Surveillance Application - Server App

- Server App
  - Connection with online cameras
    - Camera discovery by listening Broadcast/UDP heartbeat messages
    - Connecting TCP port of cameras
  - Configuring cameras
  - Collects alarms from cameras
  - Serve all information to web interface
    - Static Content Serving
    - Express and Socket.io modules of Node.js
  - Handles user inputs from web interface
    - Management of all cameras
  - Handling multi-client/multi-camera systems

Server App

```
udp_server.on("message", function (msg, rinfo) {    //rinfo Object. Remote address information
    var device_info = JSON.parse(msg);
    for(var i = 0; i < devices.length; i++){
        if(devices[i].device_id === device_info.device_id){ //if device is found, update and notify WEB clients
    }
    if(i === devices.length){                                   //no device is found; add to list
        devices[i] = {};
        devices[i].device_id = device_info.device_id;           devices[i].status = device_info.status;
        devices[i].location = device_info.location;             devices[i].vid_src = device_info.vid_src;
        devices[i].md_mode = device_info.md_mode;               devices[i].time = device_info.time;
        devices[i].ip = rinfo.address;
        /* TCP CLIENT */
        devices[i].client = new net.Socket();   //create socket (TCP client) to connect new device
        devices[i].client.connect(PORT, devices[i].ip, function() {
            console.log('CONNECTED TO: ' + devices[i].ip + ':' + PORT);
            sendNewCamBySocket(devices[i]);
        });
        devices[i].client.on('data', function(data) {    // Add a 'data' event handler for the client socket
            var alarm_data = JSON.parse(data);           // data is what the server sent to this socket
            console.log('Alarm from ' + alarm_data.camera_id);
            sendAlarmBySocket(alarm_data);
        });
        devices[i].client.on('close', function() {       // Add a 'close' event handler for the client socket
            sendDropCamBySocket(devices[i].device_id);
            devices.splice(i,1);
            console.log('Connection closed');
        });
        devices[i].client.on('error', function() {       // Add a 'error' event handler for the client socket
    }
});
```

Server App

```
/* WEBSERVER */
var express = require('express');
var app = express();
var webserver = require('http').createServer(app);
var io = require('socket.io').listen(webserver);

io.configure(function(){
  io.enable('browser client etag');
  io.set('log level', 1);

  io.set('transports', [
    'websocket',
    'flashsocket',
    'htmlfile',
    'xhr-polling',
    'jsonp-polling'
  ]);
});

app.configure(function(){
  app.use(express.static(__dirname + '/static/app'));
});

webserver.listen(3000);
console.log("Webserver listening 3000");
```

Server App

```
116  /* WEBSOCKET */
117  var websockets = [];
118  io.sockets.on('connection', function (socket) {
119    console.log('Client Connected!!!');
120    socket.on('getDeviceList', function (data) {
121      sendDeviceList(socket);
122    });
123    socket.on('config', function (data) {
124      var device = findDevice(devices, data.device_id);
125      sendDeviceConf(device.client, device.device_id, data.vid_src, data.md_mode);
126    });
127    websockets.push(socket);
128  });
129  function sendDeviceList(socket){
130    if(devices.length > 0){
131      for(var i=0; i<devices.length; i++){
132          socket.emit('caminfo', {device_id: devices[i].device_id, status: devices[i].status,
133                                  location: devices[i].location, vid_src: devices[i].vid_src,
134                                  md_mode: devices[i].md_mode, time: devices[i].time, ip:devices[i].ip});
135      }
136    }
137  }
138  function sendAlarmBySocket(alarm_data){
139      if(websockets.length > 0){
140          for(var i=0; i<websockets.length; i++){
141              websockets[i].emit('alarm', {alarm: alarm_data});
142          }
143      }
144  }
```

Server App

```
96  /* SYNC FUNCTIONS */
97  function findDevice(devices, id){
98      var device = {};
99      for(var i=0; i<devices.length; i++){
100         if(devices[i].device_id === id){
101             device = devices[i];
102             break;
103         }
104     }
105     return device;
106 }
107
108 /* CONF MESSAGE FUNCTIONS */
109 function sendDeviceConf(socket, dev_id, vid_src, md_mode){
110     var msg = new Buffer([dev_id, vid_src, md_mode]);
111     socket.write(msg);
112     console.log('Configuration sent!');
113 }
114
```

Server App

```
146
147 function sendNewCamBySocket(device){
148     if(websockets.length > 0){
149         for(var i=0; i<websockets.length; i++){
150             websockets[i].emit('newcam', {device_id: device.device_id, status: device.status,
151                                           location: device.location, vid_src: device.vid_src,
152                                           md_mode: device.md_mode, time: device.time, ip:device.ip});
153         }
154     }
155 }
156 function updateCamBySocket(device){
157     if(websockets.length > 0){
158         for(var i=0; i<websockets.length; i++){
159             websockets[i].emit('updatecam', {device_id: device.device_id, status: device.status,
160                                              location: device.location, vid_src: device.vid_src,
161                                              md_mode: device.md_mode, time: device.time, ip:device.ip});
162         }
163     }
164 }
165 function sendDropCamBySocket(device_id){
166     if(websockets.length > 0){
167         for(var i=0; i<websockets.length; i++){
168             websockets[i].emit('dropcam', {device_id: device_id});
169         }
170     }
171 }
```

Development Story of a Surveillance Application - Web Interface

- Web Interface
  - HTML5/Javascript
  - Web Sockets
    - Socket.io
  - Angular.js
  - Ajax Based
    - No need to refresh
  - Browser Compatibility
    - IE9, Firefox, Chrome tested

Web Interface
WebSocket Code Samples

- From “http://socket.io/#how-to-use”

SERVER (APP.JS)

```
var app = require('express').createServer()
  , io = require('socket.io').listen(app);

app.listen(80);

app.get('/', function (req, res) {
  res.sendfile(__dirname + '/index.html');
});

io.sockets.on('connection', function (socket) {
  socket.emit('news', { hello: 'world' });
  socket.on('my other event', function (data) {
    console.log(data);
  });
});
```

CLIENT (INDEX.HTML)

```
<script src="/socket.io/socket.io.js"></script>
<script>
  var socket = io.connect('http://localhost');
  socket.on('news', function (data) {
    console.log(data);
    socket.emit('my other event', { my: 'data' });
  });
</script>
```

Web Interface
WebSocket Code Samples

```
'use strict';
/* Controllers */
angular.module('myApp.controllers', []).
  controller('MyCtrl1', ['socket','$scope', function(socket, $scope) {
    $scope.devices = [];
    $scope.alarms = [];

    $scope.setMdMode = function(device, md_val){
        var msg = {device_id: device.device_id, vid_src: device.vid_src, md_mode: md_val};
        socket.emit('config', msg);
    };
    $scope.setVidSrc = function(device, src_val){
        var msg = {device_id: device.device_id, vid_src: src_val, md_mode: device.md_mode};
        socket.emit('config', msg);
    };
    socket.emit('getDeviceList', {data: 'abc'});

    socket.on('caminfo', function (data) {  //when socket is connected
        $scope.devices.push(data);
    });

```

Web Interface
WebSocket Code Samples

```
22
23      socket.on('newcam', function (data) {    //new camera is found after socket connectio
26      socket.on('updatecam', function (data) {
27          for(var i=0; i<$scope.devices.length; i++){
28              if($scope.devices[i].device_id === data.device_id){
29                  $scope.devices[i].time = data.time;
30                  $scope.devices[i].vid_src = data.vid_src;
31                  $scope.devices[i].md_mode = data.md_mode;
32                  $scope.devices[i].status = data.status;
33                  break;
34              }
35          }
36      });
37      socket.on('dropcam', function (data) {
38          for(var i=0; i<$scope.devices.length; i++){
39              if($scope.devices[i].device_id === data.device_id){
40                  $scope.devices.splice(i,1);
41                  break;
42              }
43          }
44      });
45      socket.on('alarm', function (data) {
46          console.log(data.alarm);
47          $scope.alarms.push(data.alarm);
48      });
49
50  }])
51  .controller('MyCtrl2', [function() {
52  }]);
53
```


aselsan

Outline

- Introduction to Node.js
- Cross-compiling Node.js and NPM
- Development environment
- Scripting samples in embedded devices
- Development story of a surveillance application
- Demo
- Questions

Questions

- Demo Codes
  - github.com/mfkaragoz/elce2013demo
- Thank You!
  - Mehmet Fatih KARAGOZ
    mfatihkaragoz@yahoo.com
  - Cevahir TURGUT
    cevahir.turgut@gmail.com

aselsan