REAL-TIME MESSAGING TO WEBAPPS FROM A PRODUCTION DATABASE

David Johnson, djohnson@ommc.com, www.djohn89.com

OUTLINE

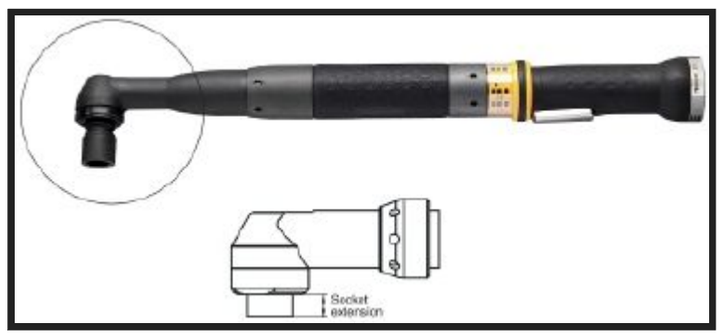
- Automotive Assembly Line
- Break
- TCP Sockets Review
- Websockets

The goal of this talk is to describe an automotive assembly line and to encourage web developers to use websockets in their web applications. See RFC 6455 and Websocket API for more details.

AUTOMOTIVE ASSEMBLY LINE

- Electric nutrunners/bolt runners
- Barcode scanners
- RFID
- Alarms

ELECTRIC NUTRUNNERS/BOLT RUNNERS



- Nuts and bolts are assembled using torque tools
- Popular manufacturers: Stanley, Atlas CopCo
- Ethernet communication module records exact torque, angle, rundown count, etc. per engineering specs

BARCODE SCANNERS

- Before attaching a part, the part number must be scanned to make sure the right part was picked
- The serial number must be scanned for billing and inventory management
- A computer program records these scans into a database

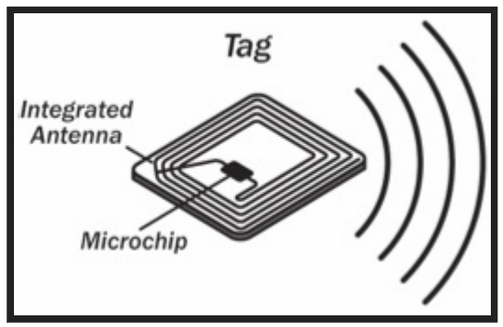
ERROR PROOFING

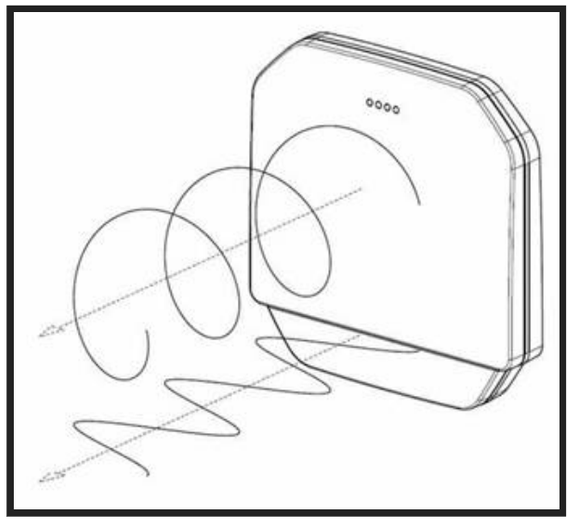
018	#01	Eng/TM/TCase/DSL Fuel Bundle			018
Sequence		VIN	Model	Module	Drive/Market
1860		HL526041	JKJP72	X86	LHD / DOM
			Received Time		Skid Id
			10/18/2016 11:06:06		A082

	Engine	Transmission	Transfer Case	Fuel Bundle
Part No	68082776AF eng gas	52108711AE gas auto	52123497AA t/c auto	52029444AF gas ret/grom
CN Part				
	776AF	711AE	497AA	444AF
Barcode	68082776AF	52108711AE	52123497AA	
Serial	TPHE1277610987	TI2TJ236661002	TDI284628090	
	OK	OK	OK	NO

Message	Part No. OK (52108711AE)

RFID



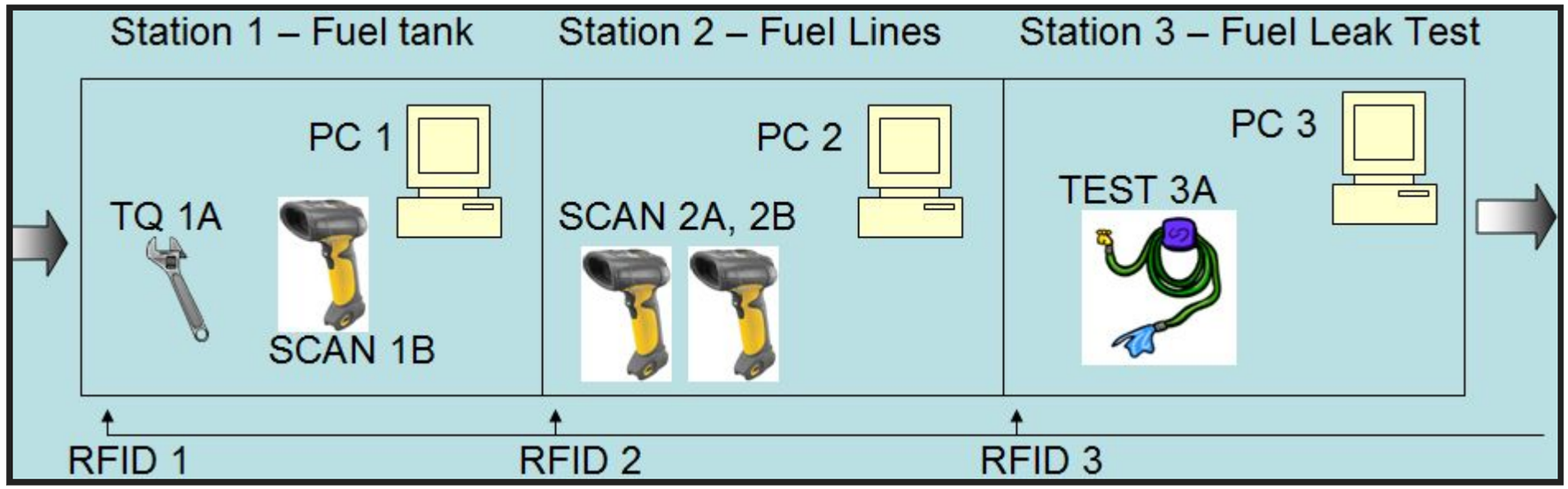


- Skillets ride on a conveyor system tracked by RFID tags and antennas
- Real-time tracking is critical for correctly building vehicles
- A computer program records the RFID events into a database

HOW ALARMS WORK

1. The PLC detects an error condition (e.g., a vehicle presses a limit switch before the torques are done), stops the production line, and turns a bit on
2. A communication program sees the bit turns on and logs it into a database
3. The cell leader sees the alarm and helps the operator to finish their job, and the error condition is removed
4. The PLC turns off the bit and allows the line to resume moving

THEORETICAL ASSEMBLY LINE



- Automotive assembly lines are primarily composed of torque tools, barcode scanners, RFID, and alarms
- Operators must complete their jobs inside the workstation defined by the RFID antennas or else alarms will be activated
- The events from these systems comprise the Manufacturing Execution System (MES), a type of supervisory control and data acquisition (SCADA)

VIDEO: TSAP ASSEMBLY

SUMMARY OF VIDEO

- Multiple assembly lines (frame, axle, ML1, ASRS, ENG, ML2)
- RFID used in each line to track skillets
- Problems are located using alarms

DECKING


Building a Wrangler
Unlimited at the Toledo
Supplier Park in 2006
(Chrysler photo
at allpar.com)
WT.3600LH
L4
L4
UNIT-4
UNIT-2
C2-S9 MIN 1 MAX 1

FINISHED PRODUCT

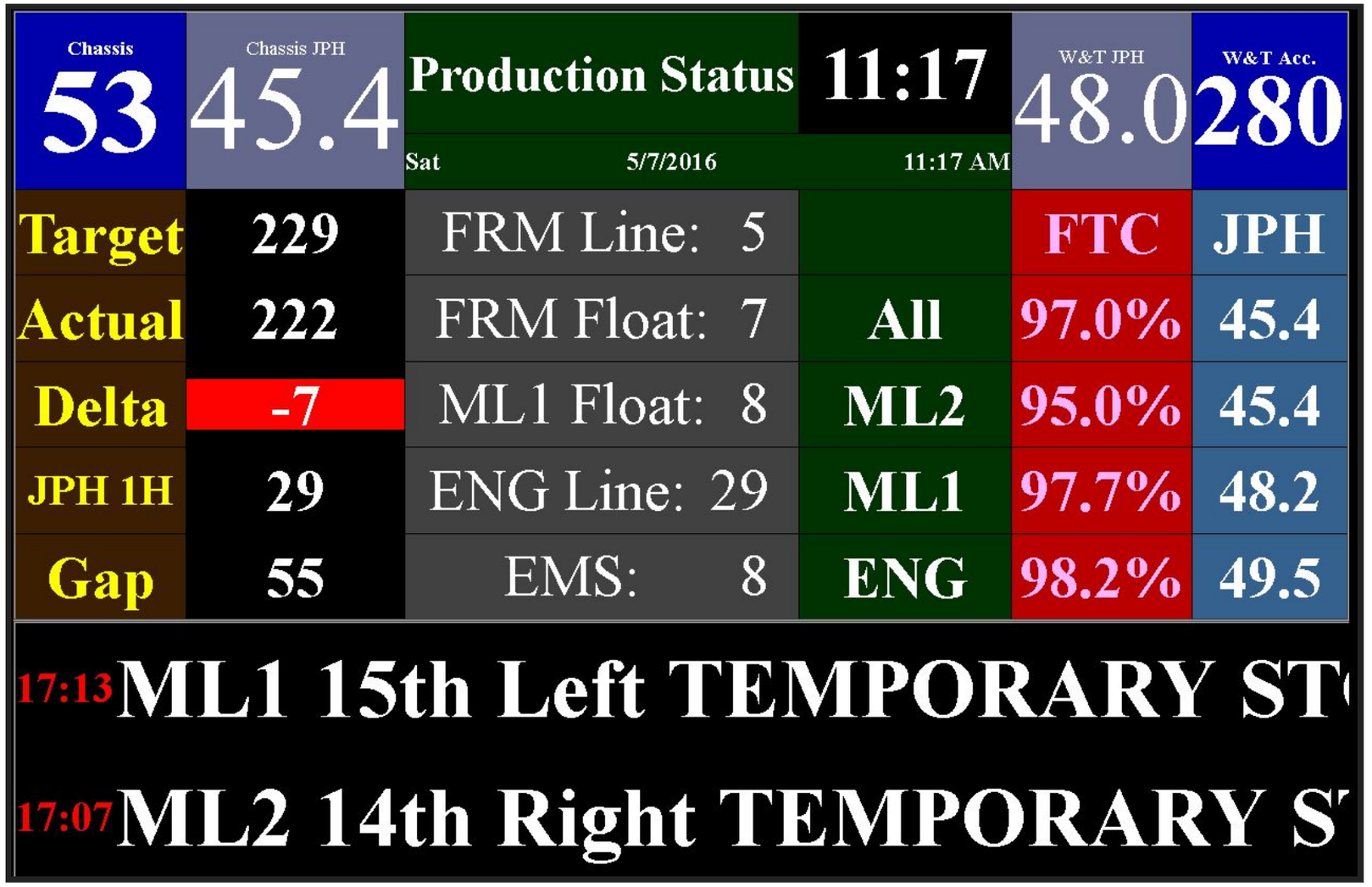
EXAMPLES OF APPLIED WEBSOCKETS

- Andon Board Upgrade (alarms, productivity calculations)
- Broadcast Elimination (paperless production)
- RFID Tracking (faster internal messaging)
- RepairTech webapp

OLD ANDON BOARD

- Light bulbs on a stadium-style scoreboard
- Shows 2 alarms (ML1, ML2)
- Limited to simple statistics

NEW ANDON BOARD WEBAPP



- New 80" TVs with JSP/Tomcat and CometD servlets
- Shows up to 8 alarms
- Calculates detailed statistics

ANDON BOARD MESSAGES

- PLC turns bit on (conveyor stopped due to no torque at station 6)
- Communication program sees bit, logs to database and sends CometD message to Andon Board servlet: {type: Alarm, Line: ML1, Station: 6, Message: Torque Overrun DC Tool}
- Andon Board javascript receives message, displays alarm to alert supervisor

ANDON BOARD EXAMPLE

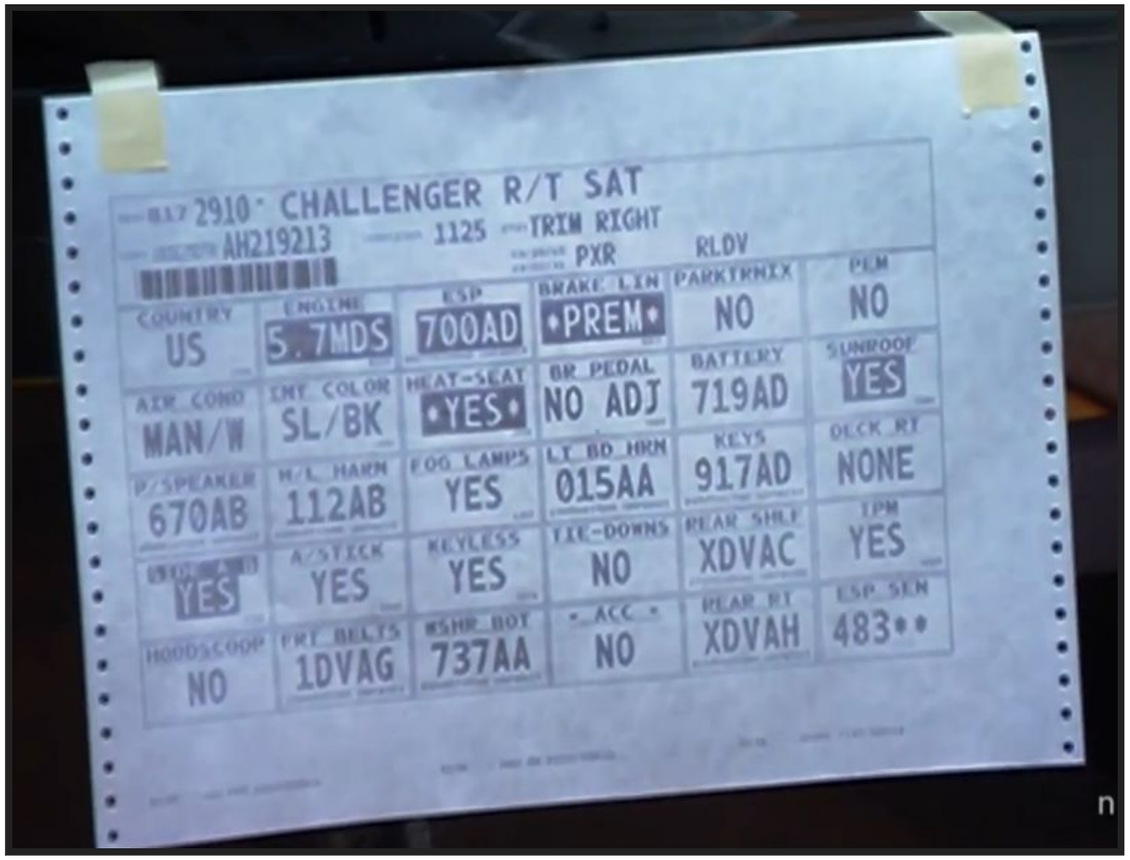
Chassis	Chassis JPH	Production Status	12:07	W&T JPH	W&T Acc.
41	42.3	Tues 10/18/2016	12:07 PM	48.0	237
Target	286	FRM Line: 8		FTC	JPH
Actual	258	FRM Float: 8	All	95.4%	42.3
Delta	-28	ML1 Float: 15	ML2	92.2%	42.3
Gap	5	ENG Line: 34	ML1	96.6%	46.6
JPH 1H	19	EMS: 10	ENG	97.3%	50.4

10:45 ML2 12th Right TEMPORARY STOP

6:28 ML1 Starved by ML2

5:26 Waiting for EMS carrier

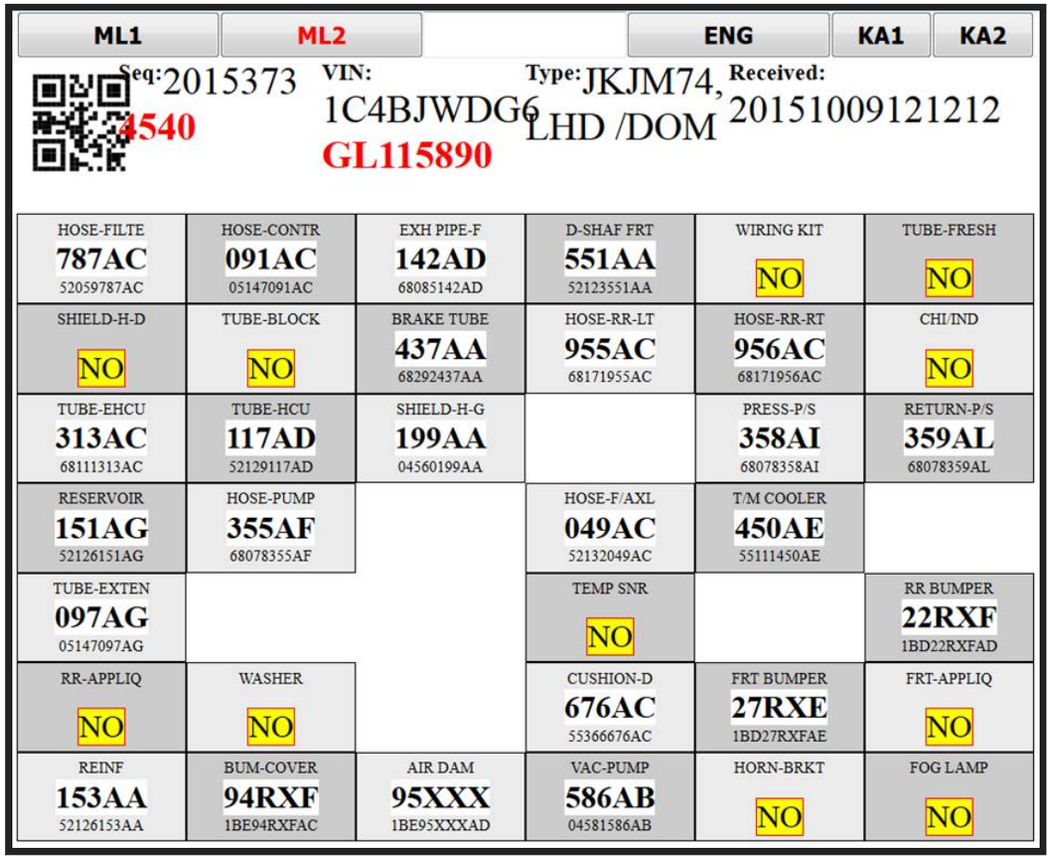
BROADCAST ELIMINATION



Printed broadcasts were wasteful ($50,000+ per year)

- ... and became unnecessary (computer error proofing, barcode scanners, digital displays).
- So we replaced them with tablet PCs displaying a webpage

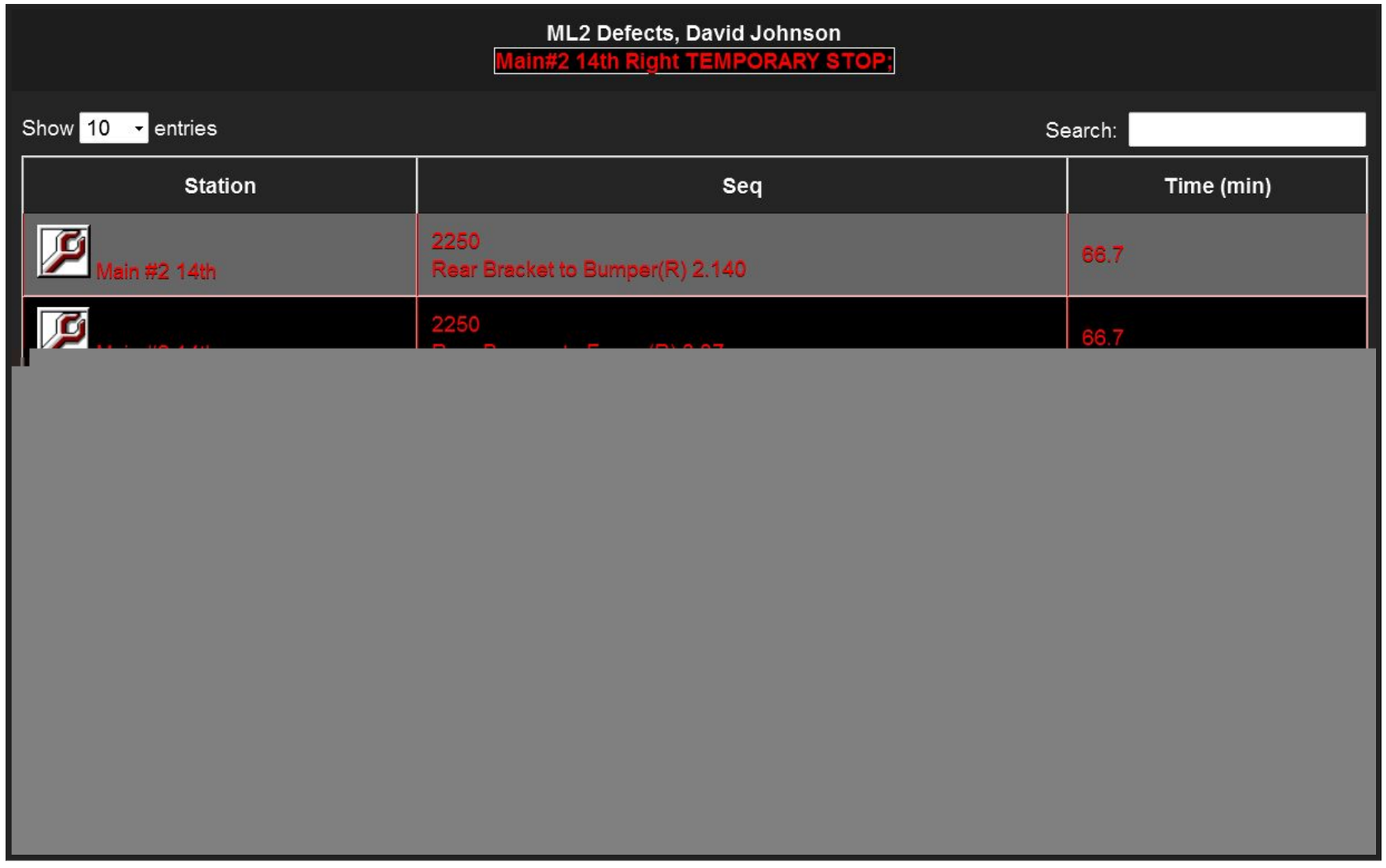
BROADCAST DISPLAY WEBAPP



Broadcast Display webapp receives messages from RFID system: {type: Arrival, Line: ML2, Station: 4, SKID: A053, VIN: 1C4... }

- It receives messages as new vehicle orders come in (for materials tracking)
- It sends messages to the RepairTech webapp (to request repairs)

REPAIRTECH WEBAPP



This webapp tracks vehicles that need repairs

- Repair Technicians physically fix the vehicle and then fill out a form
- Quality Inspectors double-check the repairs
- Internal statistics are calculated (FTC, MTTF)

Back

Repair?

Begin repair

Module Information

Station:	ML2-14.5
Seq:	20166172250
VIN:	1C4BJWFG5HL521139

Tool Information

Tool name:	Rear Bumper to Frame(R)
Tool ID:	3.87
Jumped out by:	ML2 Torque Bypass Board
Jumped out at:	2016-10-18 15:22:10.17 (66.9 min ago)
Current location:	

Back

Enter Repair

Save

Jump Details for 2250

Seq No: 20166172250

VIN: 1C4BJWFG5HL521139

Current location:

Tool Name: Rear Bumper to Frame(R)

Tool ID: 3.87 at ML2-14.5

Range: 49.5 - 60.5 Nm

Jumped out by: ML2 Torque Bypass Board at 2016-10-18 15:22:10.17

Torque count: 0 of 2 required

Choose reason(s) for repair:

- X-Thread
- Operator Behind
- Early Torque
- Missed Operation

Edit Torques

Tool ID: 3.87 at ML2-14.5

Range: 49.5 - 60.5 Nm

Torque count: 2 of 2 required.

Status: OK

Complete repair

49.5

Add new torque

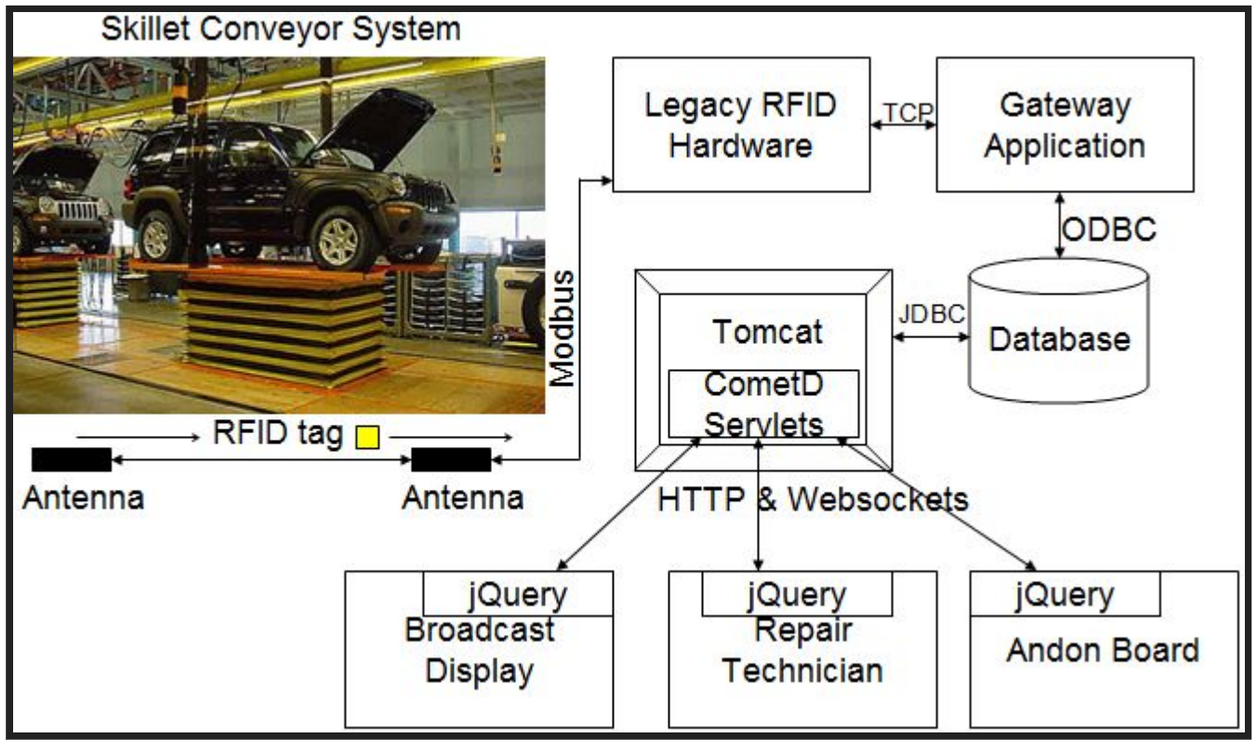
Torque 0: OK

49.5

Torque 1: OK

49.5

MESSAGING OVERVIEW



- Using webapps has been very helpful for efficient communication of events in MES (Alarms, RFID, etc.)
- Websockets are a natural fit for this system
- CometD integrated easily with existing Java infrastructure and internal websites

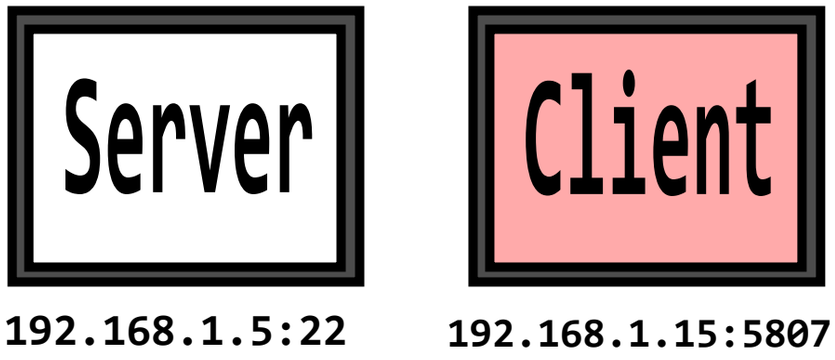
CONCLUSIONS - AUTOMOTIVE ASSEMBLY LINE

- Torque tools, barcode scanners, RFID, and alarms comprise the Manufacturing Execution System
- Messaging patterns vary based on physical design and engineering constraints
- Andon boards, Broadcast Display, and RepairTech use Websockets to display information to production team

BREAK

TCP SOCKETS REVIEW

TCP connects two endpoints (defined by IP addresses and port numbers) and allows symmetric, byte-oriented data transmission over a socket (as if reading from and writing to a file).



- Think of the socket as a file.
- You have to define a message format!
- The server listens on a port; the client connects. Afterward, transmission is symmetric.

HOW MANY BYTES CAN YOU READ IF YOU REQUEST TO READ 1000 BYTES?

- 0
- 10
- 1000
- 2000

TCP SOCKETS AS FILE IO

- Disk file operations: open, close, read, write, seek. File format is defined by application
- Socket operations: open, close, read, write. No seek. Network format is defined by application
- Asynchronous vs. synchronous operations: framing, latency, reliability, complexity

Sockets are very similar to disk files, but the reliability is much worse.

How much data did you actually read or write? How do you know if the other party is still there? Can your program block forever?

CLASSIC SOCKET PROBLEMS - PARTIAL BUFFER READS

You request to read 100 bytes (a complete message) from the socket. You receive 30 bytes.

Message Framing fixes this problem. Messages must have a length prefix or specific bytes at the beginning and end. Continue reading 70 bytes.

You read 170 bytes. The 70 bytes are for the first message; the next 100 bytes are for a subsequent message.

TCP guarantees the order of data is preserved, but it doesn't guarantee the size of any individual operations.

CLASSIC SOCKET PROBLEMS - CONNECTION SILENTLY DROPS

You request to read 100 bytes, but you never receive any data. How do you know if the other party is still there?

Keepalive messages fix this problem. Every 15 seconds, you write a dummy message to the server, which echoes another dummy message (ping/pong). No Keepalive means the connection dropped.

TCP can only detect network failures by writing data to the socket. Reading is a silent operation.

CLASSIC SOCKET PROBLEMS - ASYNCHRONOUS OPERATIONS

Can your program block forever? You request to write 10,000 bytes (100 messages), but the OS only writes up to 1,000 bytes in each data frame.

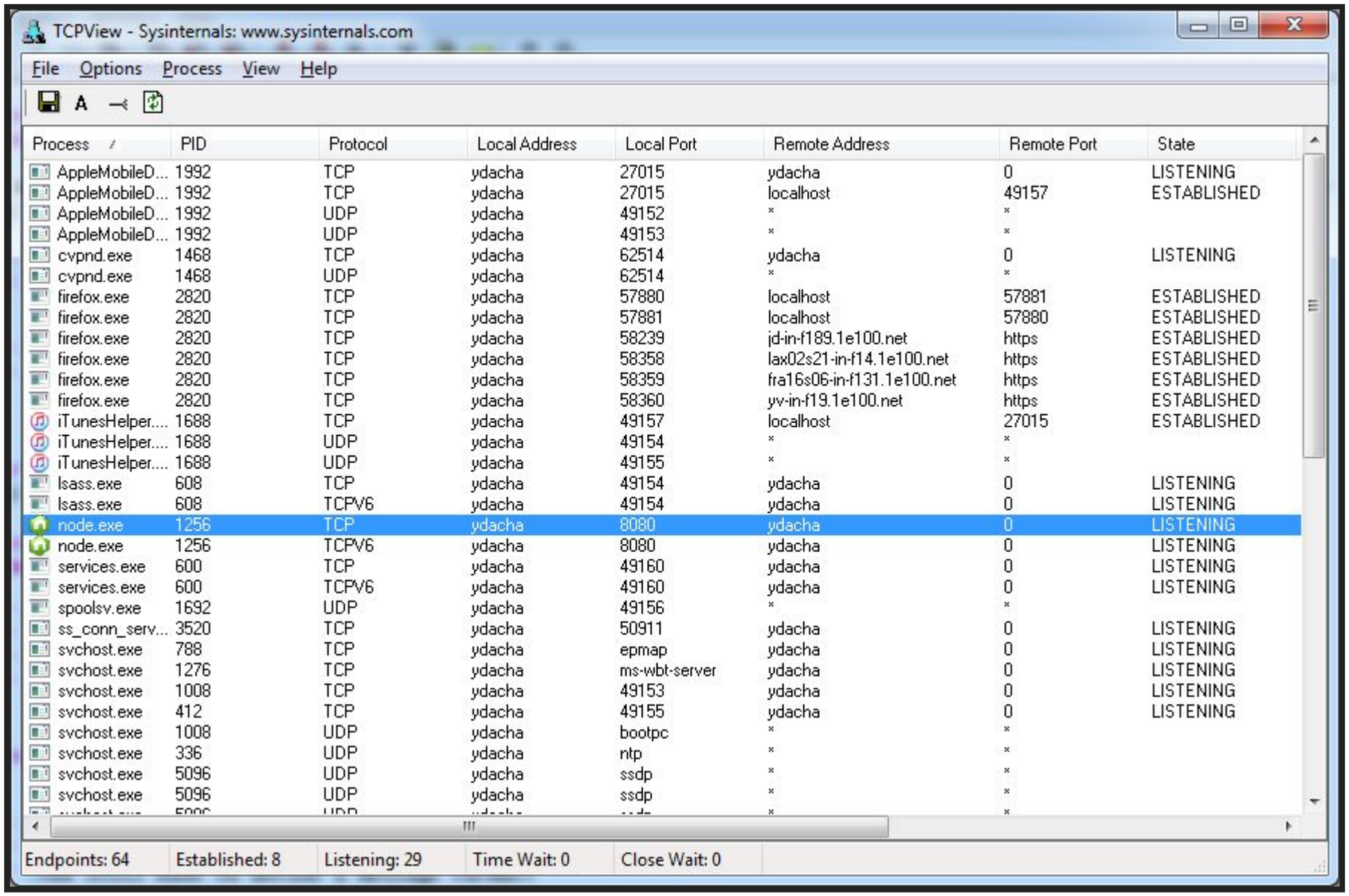
The write() calls begin to block, waiting for acknowledgement because the receiver has a finite buffer size (the TCP window).

TCP cannot guarantee that a read or write operation completes by any specific time, so your program must be asynchronous to remain responsive to the user. Use callbacks, events, threads, or queues.

SOLUTIONS TO CLASSIC SOCKET PROBLEMS

- Any TCP-based protocol must use Message Framing, Keepalive Messages, and Asynchronous Operations.
- A good protocol and library will fix these problems for you, but your application must be aware of them
- Websockets includes free Message Framing, but your library still needs to provide Keepalive Messages, and your program must still use Asynchronous Operations.

TCPVIEW DEMO



Download TCPView here (free!)

LONG POLLING

Long Polling is an older alternative to websockets. Think of an HTTP request that doesn't end.

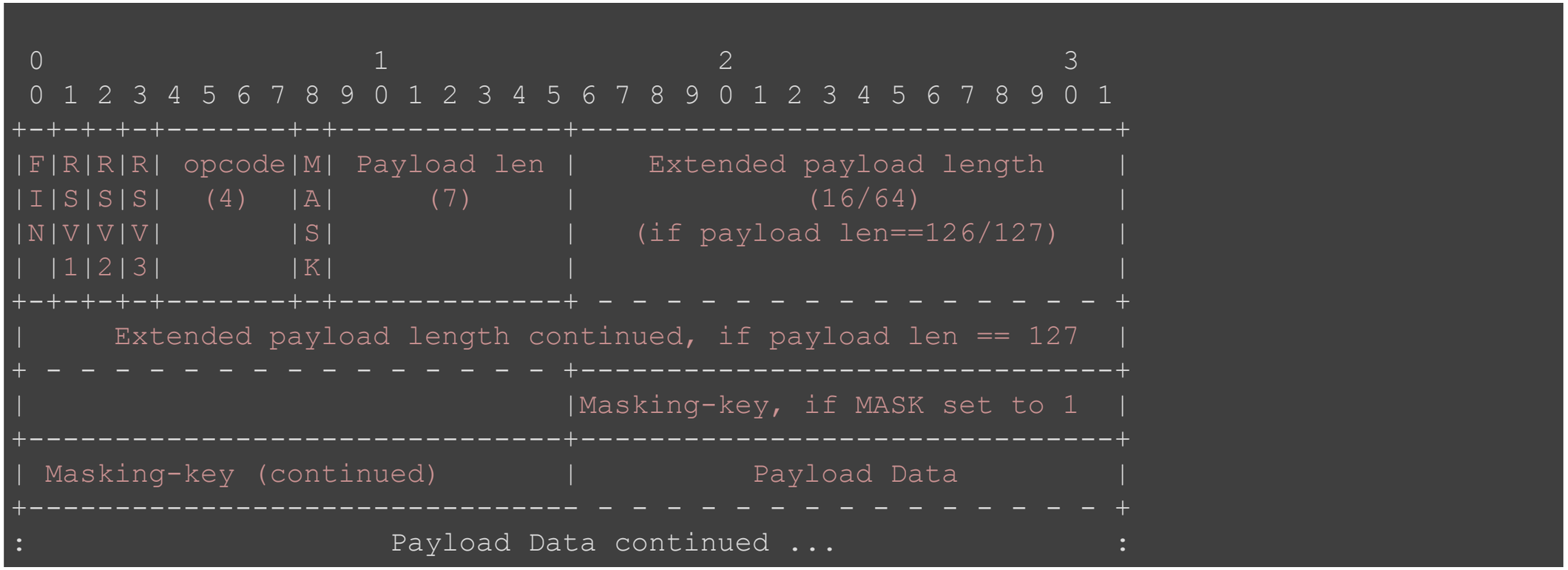
Every request with long polling must resend all headers (~800 bytes out of a 1500 byte MTU). Cookies can eat up a lot of bandwidth. Most headers don't change, anyway.

```
GET /servlets/SomeServlet HTTP/1.1
Host: www.somehost.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1.5) Gecko/20091102 Firefox/3.5.5 (.NET CLR
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Cookie: JSESSIONID=j0Uto+XV00fds-qZCfTUNA__.h422a
Pragma: no-cache
Cache-Control: no-cache
```

WEBSOCKETS

- A websocket is an Upgraded HTTP connection (from HTTP/1.1 to WS/WSS)
- Minimal message framing; no headers; server can push data to client.
- All TCP Caveats apply to websockets

WEBSOCKET DATA FRAME



WEBSOCKET SERVER

- A HTTP server supporting websockets is required. E.g., node, tomcat
- SSL is strongly recommended because the websocket security is very weak, and TCP packet checksums are not immune to collisions
- Also, some deep packet inspecting routers and mobile data carriers will drop websocket traffic
- Your library should provide a fallback to long polling for these situations

WEBSOCKET API EXAMPLE

```
function testWebSocket()
{
  var websocket = new WebSocket("ws://localhost:8080/", "echo-protocol");
  websocket.onopen = function(evt) {
    document.write("Connected!");
    websocket.send("Hello world!");
  };
  websocket.onmessage = function(evt) { document.write("Received " + evt.data); };
}
```

Launch echo demo

NODE JS SERVER CODE

```
var WebSocketServer = require('websocket').server;
var http = require('http');
var server = http.createServer();
server.listen(8080, function() { ...  });
wsServer = new WebSocketServer({ httpServer: server });
wsServer.on('request', function(request) { ... } );
```

MESSAGE QUEUES

- Additional message queue functionality you'll need: more detailed message format, client identification, channels, publish and subscribe, at least once delivery, idempotent messages
- If you don't do these things, be prepared to deal with Fallacies of distributed computing, the Byzantine Generals' Problem, and CAP Theorem

MESSAGE QUEUE PROPERTIES

- **client identification**: a unique identifier is assigned to each client
- **message identification**: a unique identifier is assigned to each message (by application ID, timestamps, sequence numbers, etc.)
- **message channels**: messages are sent on a channel, which segments clients
- **publish and subscribe**: clients can publish messages to a channel, which relays them to some number of subscribing clients
- **at least once delivery**: the server takes ownership of a message and resends it to clients until they acknowledge receiving it, then sends a confirmation to the originator
- **idempotent messages**: receiving the same message more than once has no effect.

ADDITIONAL PROPERTIES

- **message persistance**: if you need persistance, use a real message queue system (not just websockets)
- Atomicity, Consistency, Isolation, Durability: if you need these, use a

database

EXAMPLES OF REAL WORLD LIBRARIES

- Javascript: Socket.IO, SockJS, and many others
- Java: CometD, Jetty, Resin
- ASP.NET: SignalR
- Various Message Queues: RabbitMQ with STOMP, Apache ActiveMQ, IBM MQ Series
- Now you can finally build an application and solve a business problem!

COMETD SERVLETS AND NODE JS

CometD(Java) and socket-io(node.js) provide:

- message framing (start, stop, length)
- keepalives (aka. heartbeats, to detect communication loss)
- client identification
- channels (segmented clients) with publish and subscribe
- fallback to long polling (IE8 and older, mobile data providers, etc.)

They do NOT provide:

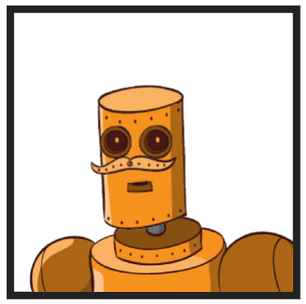
- message persistance
- at least once delivery guarantee
- Atomicity, Consistency, Isolation, Durability

CONCLUSIONS - WEBSOCKETS

- Websockets are fast and efficient, but be aware of limitations of all TCP sockets
- Some message queue functionality is available
- Choose a client and server library with these limitations in mind

END

PRIZES



Your IP: ::1

Your SID:
uZi0M0K8YIp3pe3JAAAI

Outcome:

Prize ID:

Winner SID:

Outcome: