DASH7 Distributed File Management System

By Ronald Pulvermacher, Krzysztof Berezowski, Maarten Weyn, and Michael Andre

Have you ever seen your developers struggle with outwardly simple distributed embedded application? Have you seen them deceived by its functional simplicity then combated and defeated by the complexity of power management, connectivity management, packet routing, message queuing, synchronization, you name it?

DASH7 abstracts those nightmares away. It does not talk to you in the usual language of packets, payloads, source and destination addresses. It does not force you to deal with the technical details of *how to relay data* through the network from one endpoint to the other. Rather than that, it presents itself as a distributed file system – a high level abstraction through which you communicate directly with *information endpoints* in your network, letting the DASH7 protocol do the grunt work in the background. DASH7 files use well-established and - more importantly - well-known abstracts – they are *readable*, *writeable*, *executable*, and finally *queryable*, allowing you to focus on the most important task of the connected embedded system you have at hand – producing and providing information.

So now you understand that DASH7 is really a technology that allows managing your smart connected devices as a multi-point, wirelessly connected, distributed file system. To illustrate the point, we will describe an example of architecture that involves everyone in an office using a coffee cup temperature measurement coaster with a multi-colored LED to indicate conditions to the coffee user. So to begin, every coaster contains a "file". There are three floors of offices and two office buildings next to each other all with smart temperature coasters. The entire system will report to a cloud service via two separate gateways, one in each building. Each floor of each building will contain a sub-controller to talk to all of the coasters on a particular floor.

First of all, we set the coaster to report their temperatures every 2 minutes to the cloud. So they send a "**file"** to the sub-controllers that gets relayed to the gateways and pushed up to the cloud. Now since the President of the company is an important person, we want to change the President's reporting interval to once a minute. So we log into the cloud service and send down an "**interval file**" to the President's coaster. This gets forwarded to the gateways and sub-controllers. This demonstrates "**one-to-one"** communication. We could also read the President's coaster temperature file immediately so we know if he needs a coffee warm-up.

Now since the South American Sales team achieved a record month, we want to turn their coaster's LED blue. I can do this by logging into the cloud and sending down an "**executable file"** to all of the South American Sales people. The coasters are ad-hoc woken up using the low power advertisement protocol. This allows the coasters to not lose energy on constant synchronization. When received, the coasters will execute the file which tells them to turn their blue LED on. This demonstrates sending an "**executable file**" and demonstrates "**one-to-many**".

Now we want all of the coasters that detect cold coffee to report their temperature file to the cloud. A condition occurs that signals the coaster to send a file up to the cloud. This demonstrates "**many-to-one**" and also how a sensory condition can trigger an event to send a file.

Now we want everyone in the Marketing Department to send their new product pricing to the Sales Department, Accounting Department, and to the President. This demonstrates "**many-to-many**".

Lastly, we want to turn the LED red on the President's coaster. However, we do not know which building he is in as he attends many meetings. The cloud service sends the file to the gateway and routes it to each of the sub-controllers, and finally onto the President's coaster. This demonstrates the "**transitory**" feature of DASH7. The President can move around the office buildings and still maintain communication to any coaster or the cloud. The initiation of this command does not have to originate from the cloud but could come from anyone's coaster. Let's pretend that each coaster had a panic button on it that would turn on the president's red LED. Sending an executable file can do this.

Now if the competitor's office were next door, we would not want them to be communicating to our employees. In this case, the data must be encrypted before it is sent. The message will use any DASH7 sub-controller in listening range to route the file. However, with DASH7, coasters do not have to maintain a link or session with the network, and they will not automatically announce or advertise, remaining silent and invisible to competitor's office unless they are specifically interrogated with the right credentials. This illustrates the "**stealth**" feature.

In summary, I hope this helps you understand the power of the flexible "**Distributed File System of DASH7**".

For more information, visit the DASH7 Alliance website at
http://www.dash7-alliance.org