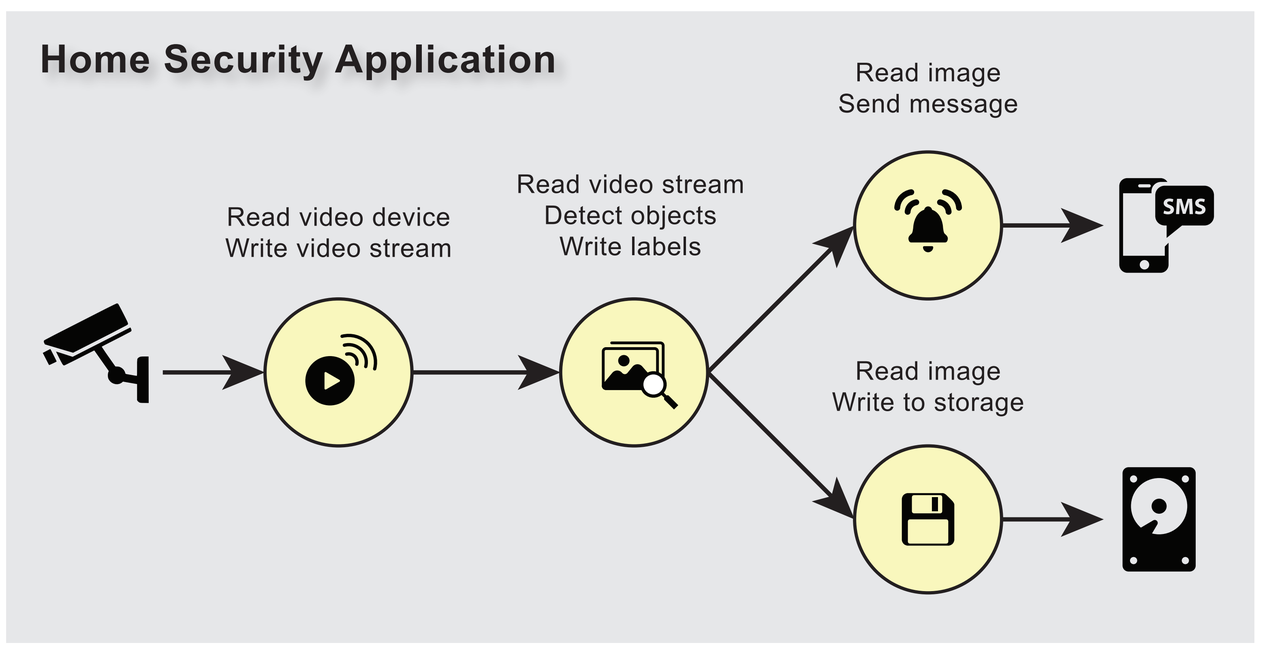
EdgeShell a Language for Composing Edge Applications

Kumseok Jung
kumseok@ece.ubc.ca

Julien Gascon-Samson
julien.gascon-samson@etsmtl.ca

Karthik Pattabiraman
karthikp@ece.ubc.ca

Motivation: Build a Reliable and Performant Edge Application

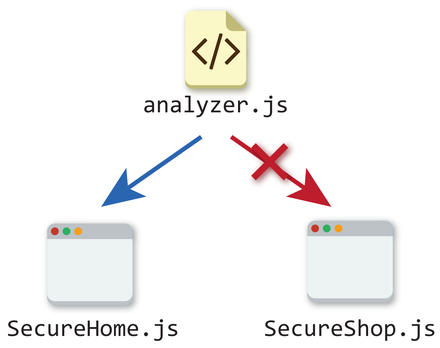


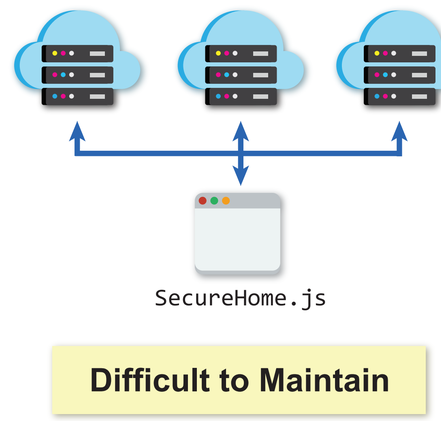
Problem: Tight coupling of Application Logic and Logistics

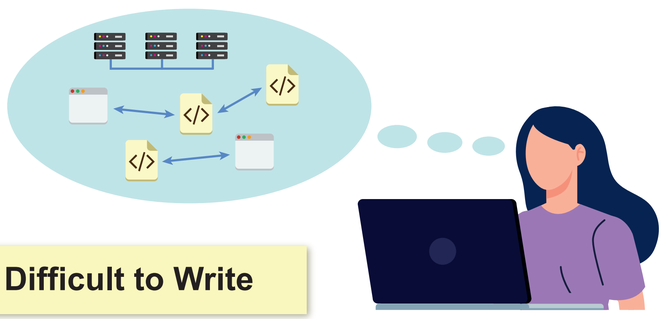
analyzer.js

```
var rtmpClient = rtmp('rtmp://a.example.org');
var httpClient = http('http://b.example.org');
var mqttClient = mqtt('mqtt://c.example.org');
rtmpClient.on('data', async (data) => {
   var frame = getFrame(data);
   var labels = await vision.analyze(frame);
   if (IsInteresting(labels)){
      var message = { frame: frame, labels: labels };
      httpClient.post(message);
      mqttClient.publish(message);
   }
});
```

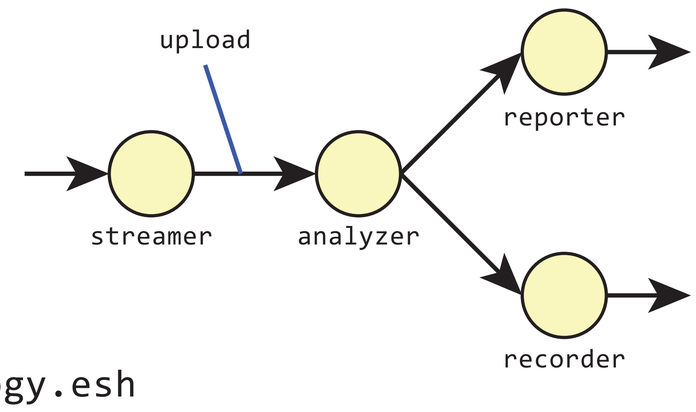
Application Logic Logistics







Solution: *EdgeShell* - a Domain-specific Language to decouple Logic and Logistics



topology.esh

```
graph HomeSecurity (device){
   node streamer: process('ffmpeg', device)
   node analyzer: process('node', 'analyzer.js')
   node reporter: process('node', 'reporter.js')
   node recorder: process('python', 'recorder.js')
   edge upload: streamer -(rtmp)-> analyzer
   analyzer -> reporter
   analyzer -> recorder
}
```

analyzer.js

```
process.stdin.on('data', async (data) => {
   var frame = getFrame(data);
   var labels = await vision.analyze(frame);
   if (IsInteresting(labels)){
      var message = { frame: frame, labels: labels };
      process.stdout.write(message);
   }
});
```