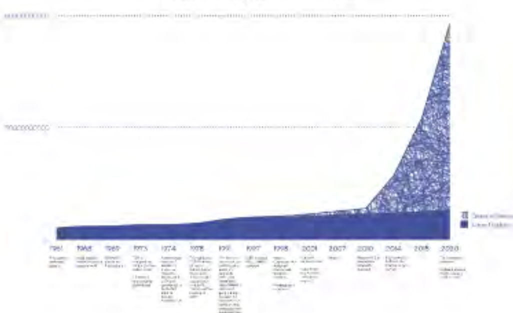
The Internet of Things

Also known as the "Industrial Internet", "Machine to Machine (M2M)", and the "Internet of Everything"

General Electric's "Industrial Internet" is perhaps the most exciting vision because it directly envisions new applications.

"the convergence of machine and intelligent data... to create brilliant machines."

A brief history of the Internet...



So what exactly is it?

What is all the hype?

Fun Facts

- 80% of doctors use mobile devices allowing remote monitoring of patients
- Nearly 60% of consumers user smart phones to shop
- 80 "things" per second are connecting to the Internet.

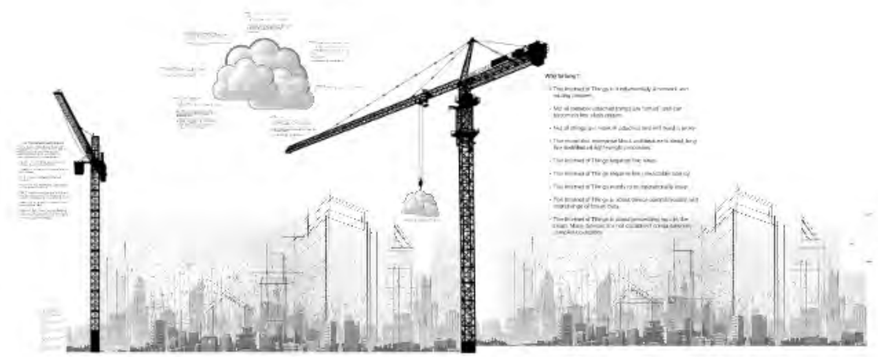
The connected home...

The connected city...

The connected industry...

The Internet of Things

The Real Challenges of Building the IoT



And so many more use cases...

Rushing into the Internet of Things

PREZI

The Internet of Things

Bryan Hughes

CTO and Co-Founder
Go Factory
bryan@go-factory.net

The Internet of Things

Bryan Hughes

CTO and Co-Founder
Go Factory
bryan@go-factory.net

Go-Matic available in the iTunes app store

The Internet of Things

Bryan Hughes

CTO and Co-Founder
Go Factory
bryan@go-factory.net

Go-Matic available in the iTunes app store

Also known as, the “Industrial Internet”, “Machine to Machine (M2M)”, and the “Internet of Everything”.

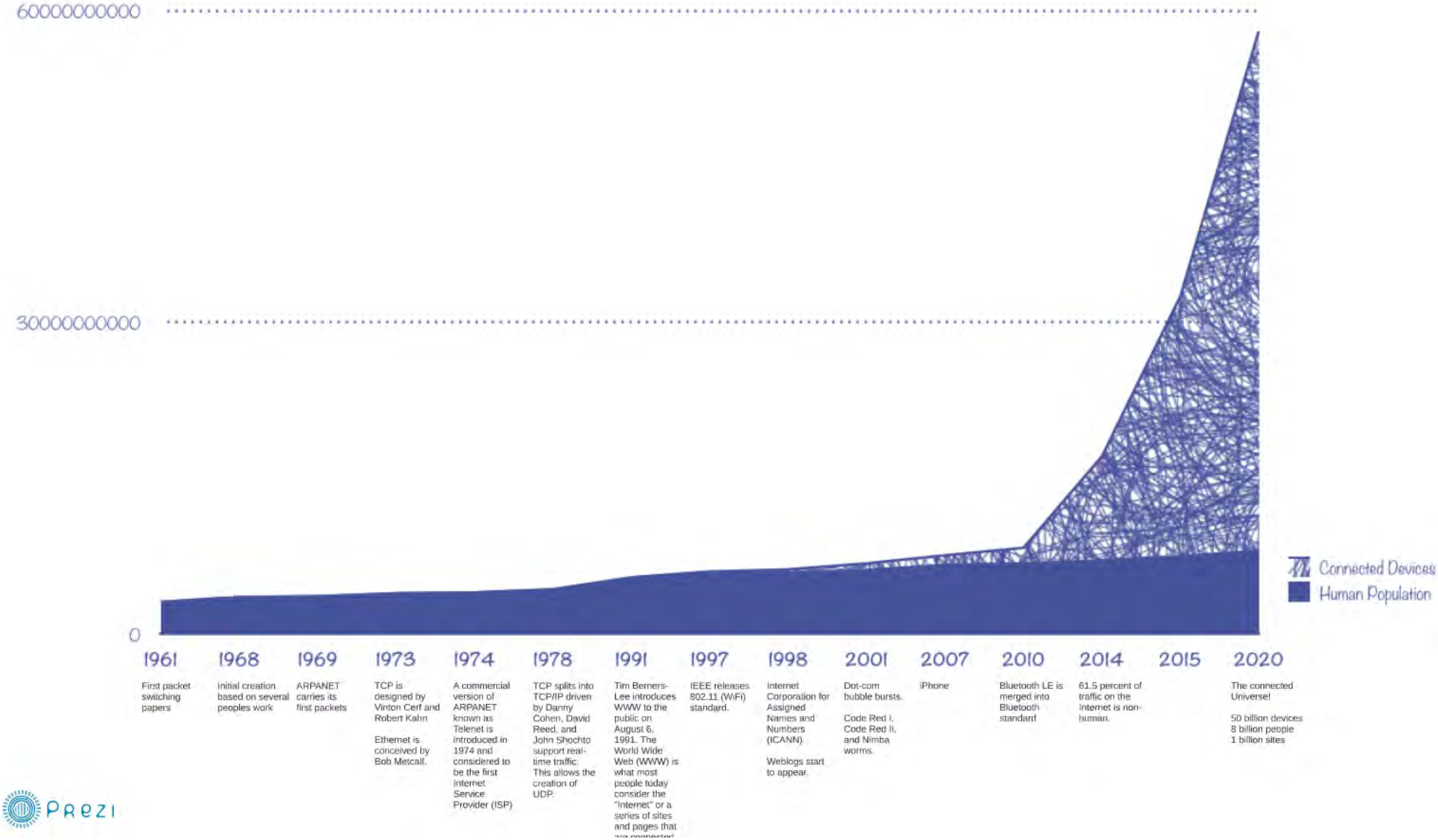
General Electric’s “Industrial Internet” is perhaps the most exciting vision because it directly envisions new applications.

“the convergence of machine and intelligent data… to create brilliant machines.”

So what exactly is it?

What is all the hype?

A brief history of the Internet...



0

1961

First packet switching papers

1968

Initial creation based on several peoples work

1969

ARPANET carries its first packets

1973

TCP is designed by Vinton Cerf and Robert Kahn

Ethernet is conceived by Bob Metcalf.

1974

A commerc
version of
ARPANET
known as
Telenet is
introduced
1974 and
considered
be the first
Internet
Service
Provider (I

69

PANET
ies its
packets

1973

TCP is designed by Vinton Cerf and Robert Kahn

Ethernet is conceived by Bob Metcalf.

1974

A commercial version of ARPANET known as Telenet is introduced in 1974 and considered to be the first Internet Service Provider (ISP)

1978

TCP splits into TCP/IP driven by Danny Cohen, David Reed, and John Shochto support real-time traffic. This allows the creation of UDP.

1991

Tim Berners-Lee introduces WWW to the public on August 6, 1991. The World Wide Web (WWW) is what most people today consider the "Internet" or a series of sites and pages that are connected with links.

1997

IEEE releases 802.11 (WiFi) standard.

1998

Internet Corporation for Assigned Names and Numbers (ICANN).

Weblogs start to appear.

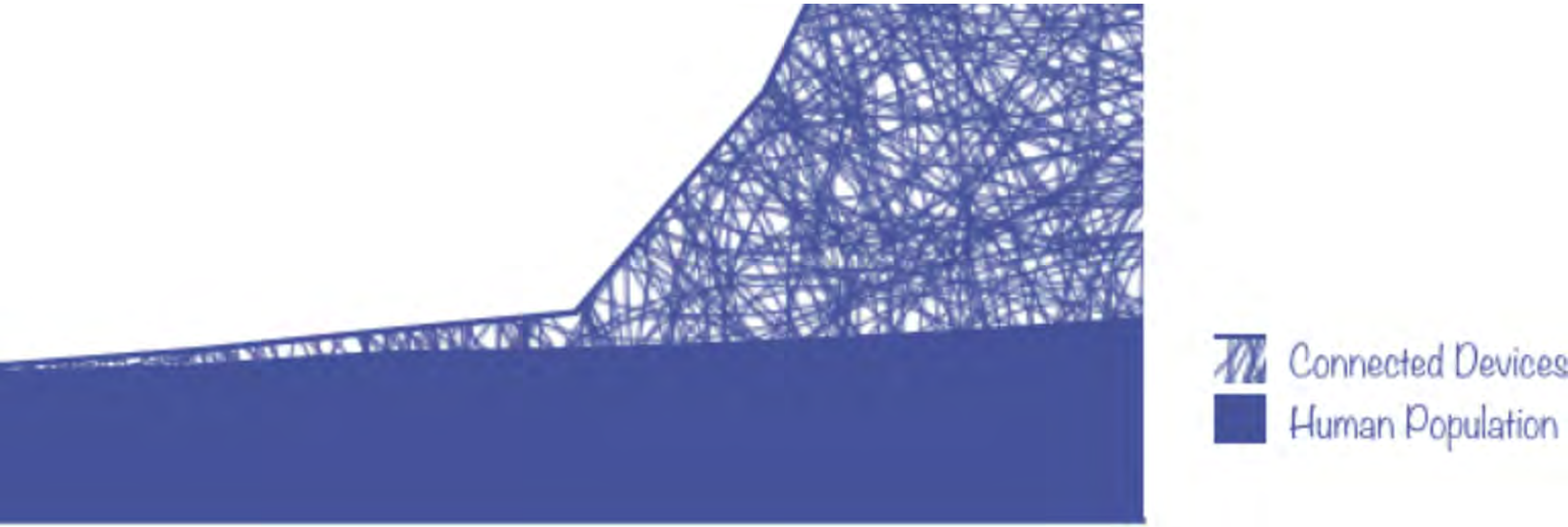
2001

Dot-com bubble bursts.

Code Red I, Code Red II, and Nimba worms.

200

iPhone



98

rnet
poration for
gned
nes and
nbers
NN).

logs start
ppear.

2001

Dot-com
bubble bursts.

Code Red I,
Code Red II,
and Nimba
worms.

2007

iPhone

2010

Bluetooth LE is
merged into
Bluetooth
standard

2014

61.5 percent of
traffic on the
Internet is non-
human.

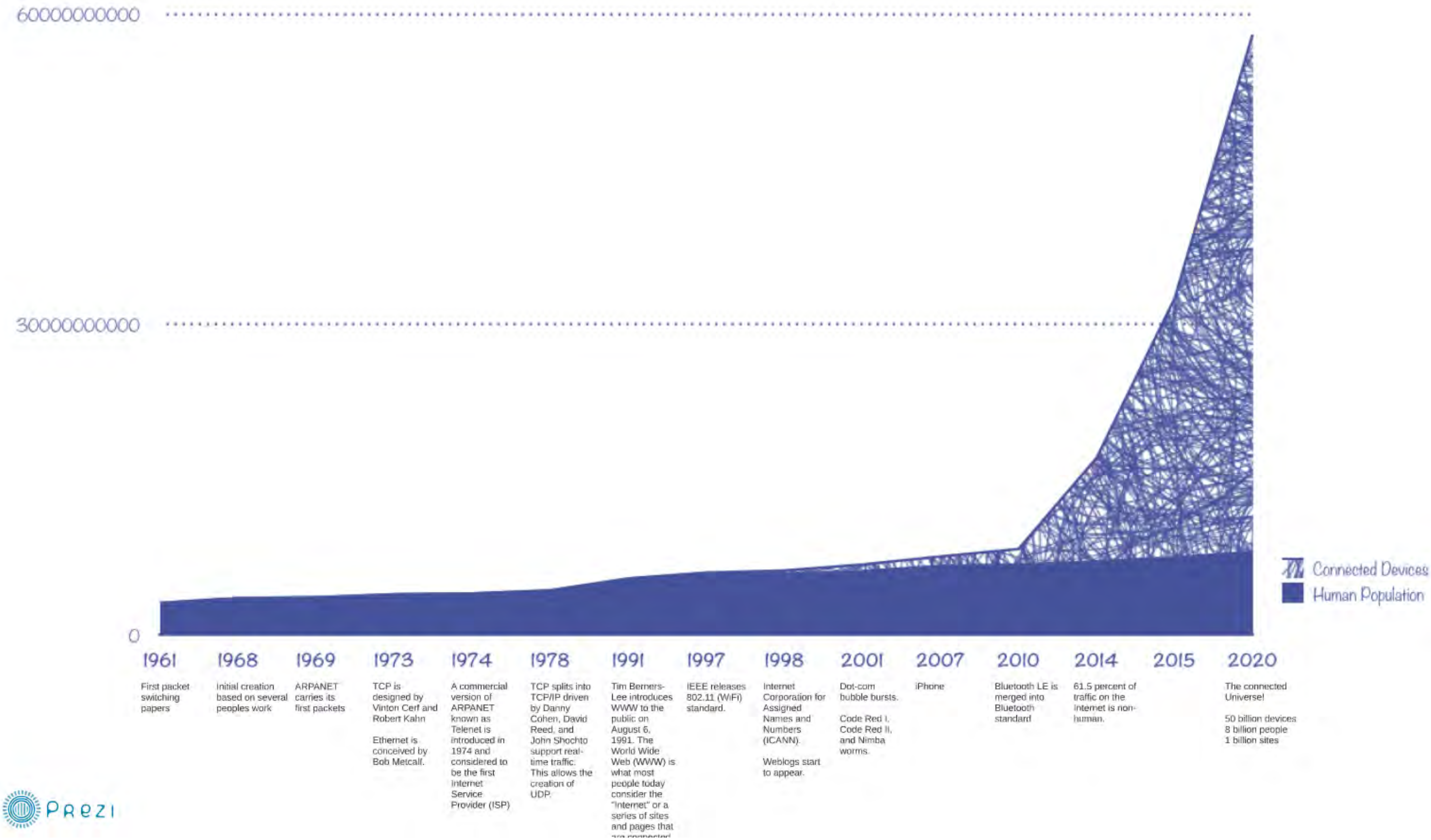
2015

2020

The connected
Universe!

50 billion devices
8 billion people
1 billion sites

A brief history of the Internet...



PRezI

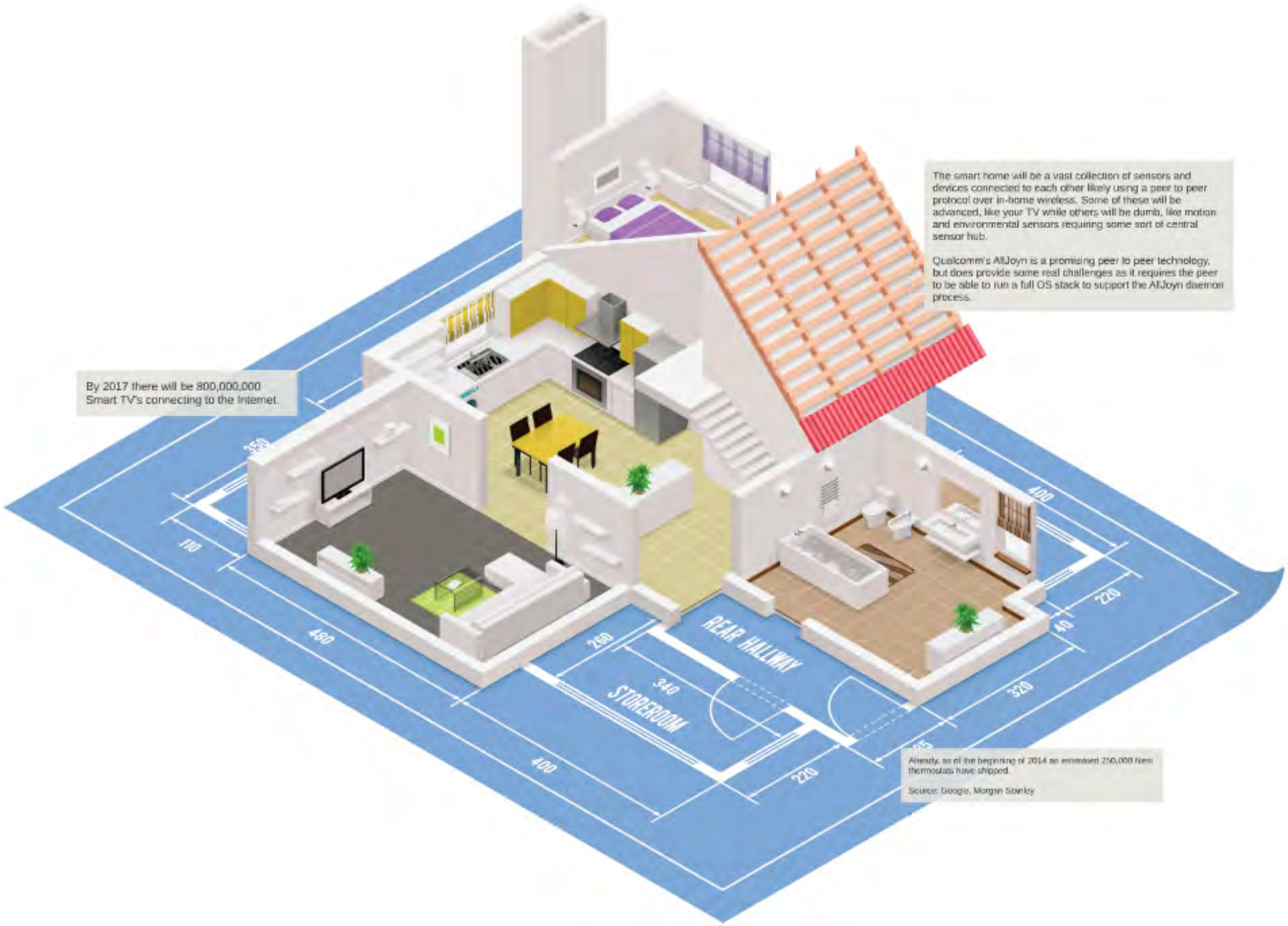
Fun Facts

- 80% of doctors use mobile devices allowing remote monitoring of patients

- Nearly 60% of consumers user smart phones to shop

- 80 “things” per second are connecting to the Internet.

The connected home...



By 2017 there will be 800,000,000 Smart TV's connecting to the Internet.

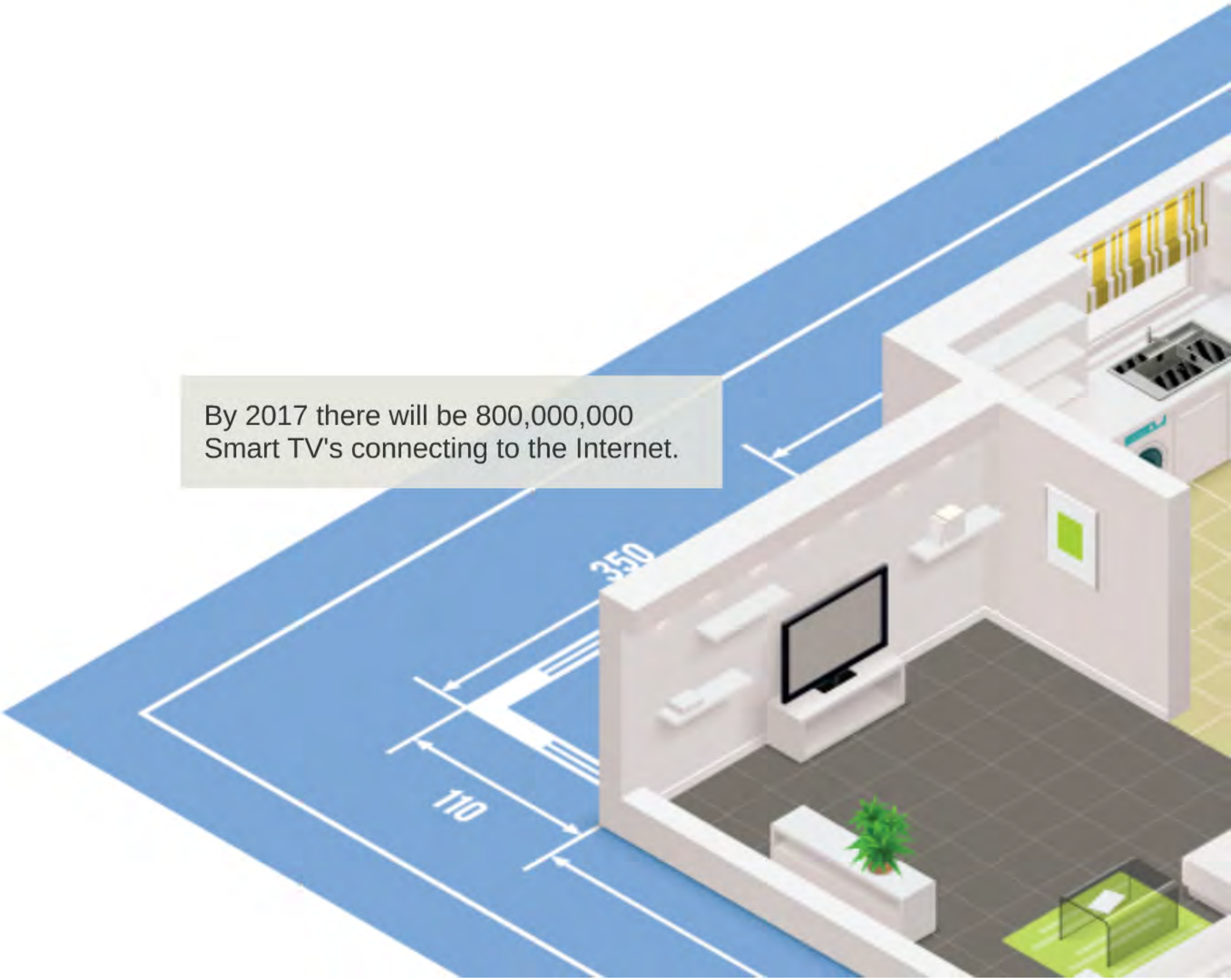
The smart home will be a vast collection of sensors and devices connected to each other likely using a peer to peer protocol over in-home wireless. Some of these will be advanced, like your TV while others will be dumb, like motion and environmental sensors requiring some sort of central sensor hub.

Qualcomm's AllJoyn is a promising peer to peer technology, but does provide some real challenges as it requires the peer to be able to run a full OS stack to support the AllJoyn daemon process.

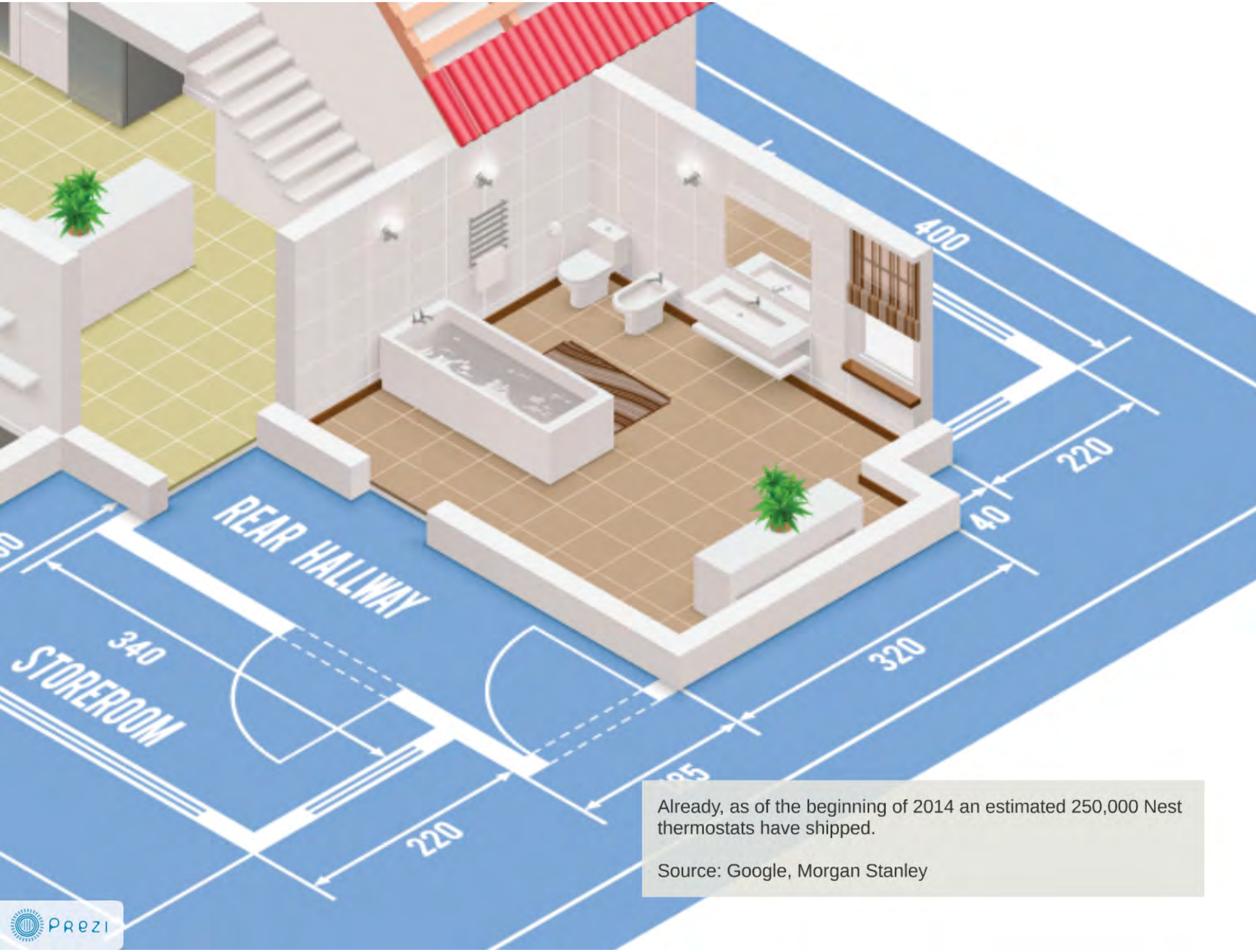
Already, as of the beginning of 2014 an estimated 250,000 Nest thermostats have shipped.

Source: Google, Morgan Stanley

By 2017 there will be 800,000,000
Smart TV's connecting to the Internet.

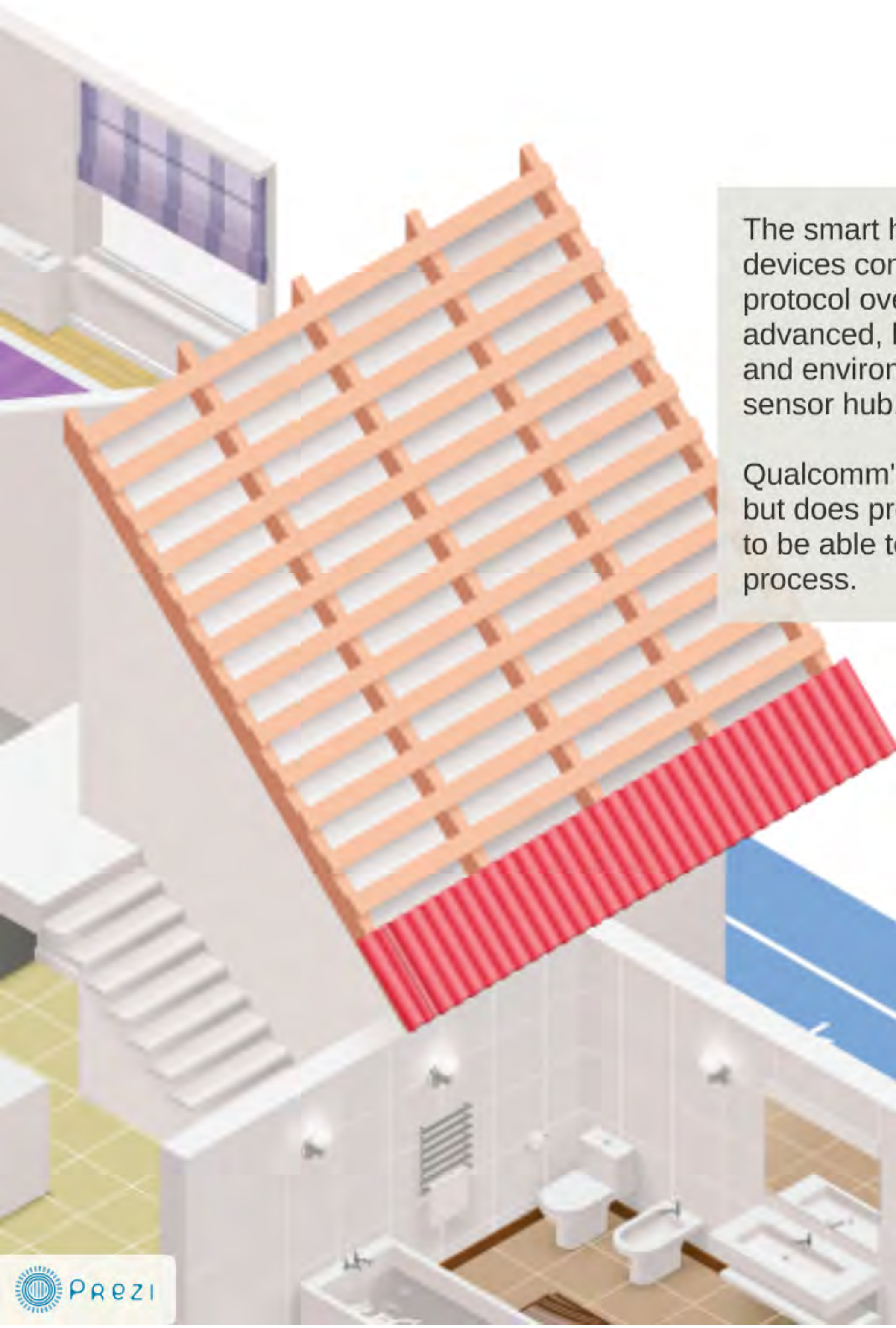


Prezi



Already, as of the beginning of 2014 an estimated 250,000 Nest thermostats have shipped.

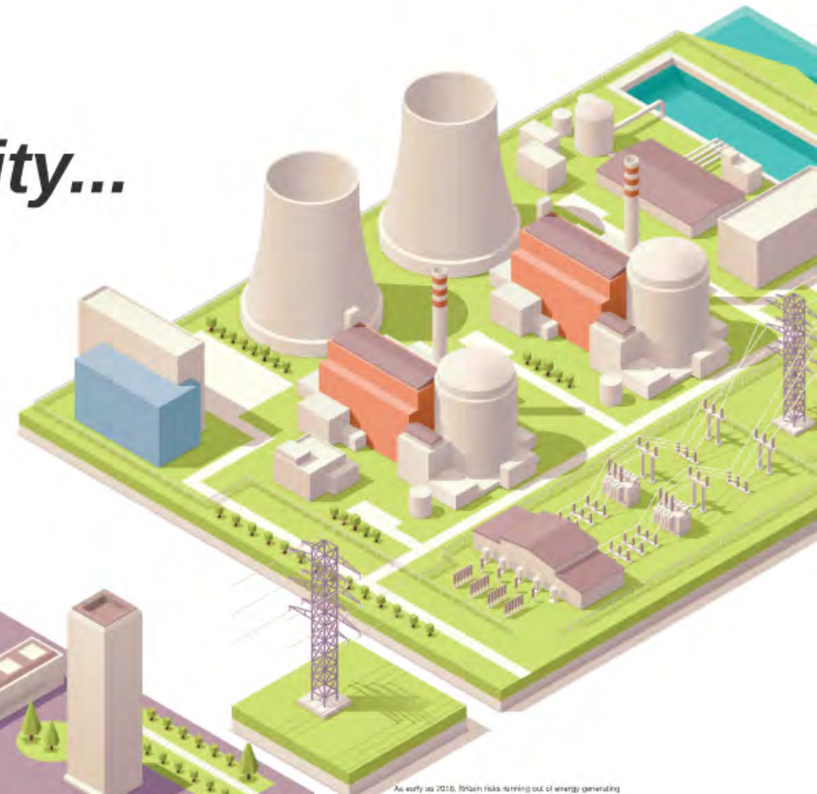
Source: Google, Morgan Stanley



The smart home will be a vast collection of sensors and devices connected to each other likely using a peer to peer protocol over in-home wireless. Some of these will be advanced, like your TV while others will be dumb, like motion and environmental sensors requiring some sort of central sensor hub.

Qualcomm's AllJoyn is a promising peer to peer technology, but does provide some real challenges as it requires the peer to be able to run a full OS stack to support the AllJoyn daemon process.

The connected city...



As early as 2016, Britain risks running out of energy generating capacity, according to the energy regulator Ofgem .

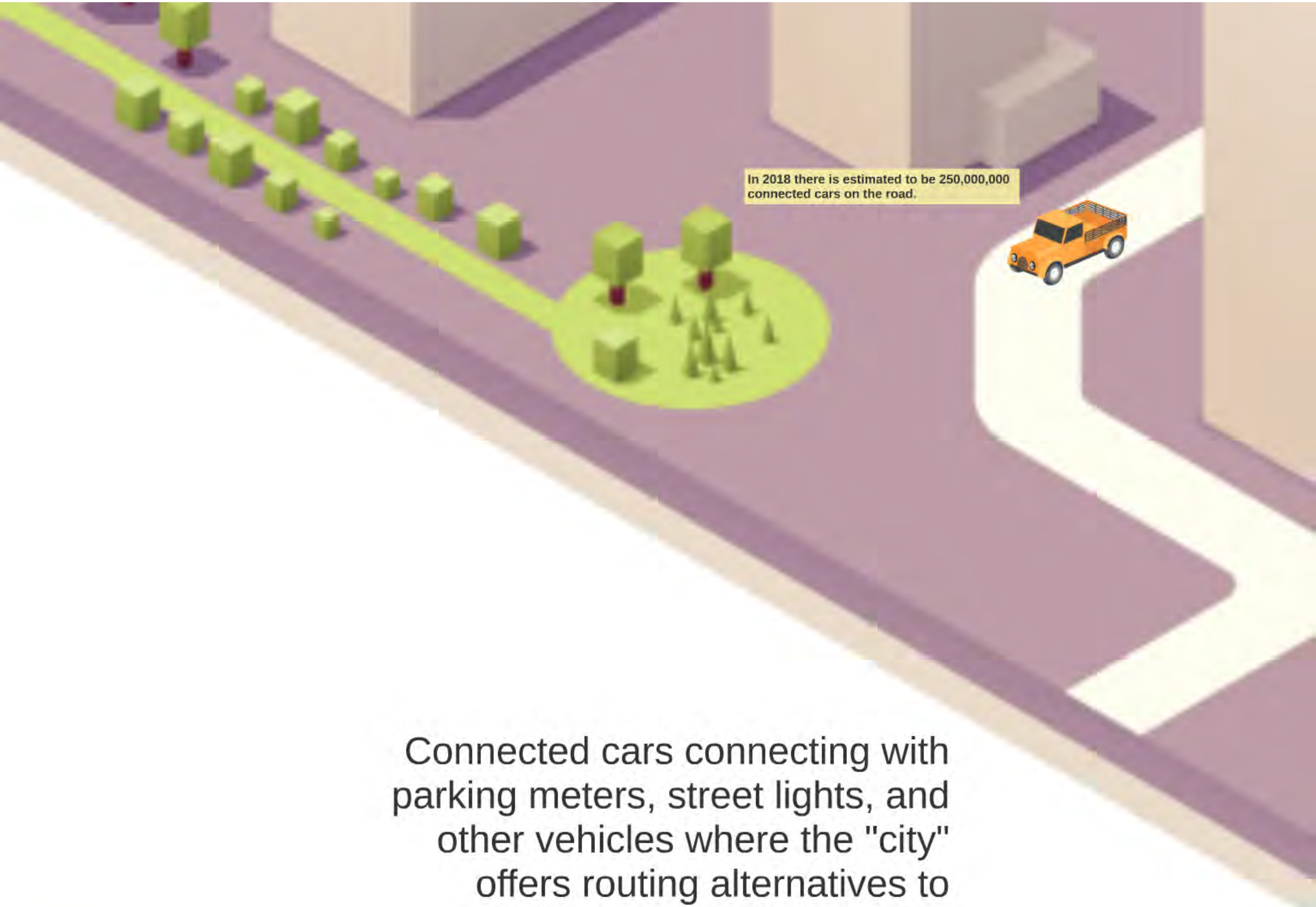
Its report predicted that the amount of spare capacity could fall from 14% now to only 4% in three years.

Source: BBC News, 5 October 2012

Connected cars connecting with parking meters, street lights, and other vehicles where the "city" offers routing alternatives to optimize traffic patterns.

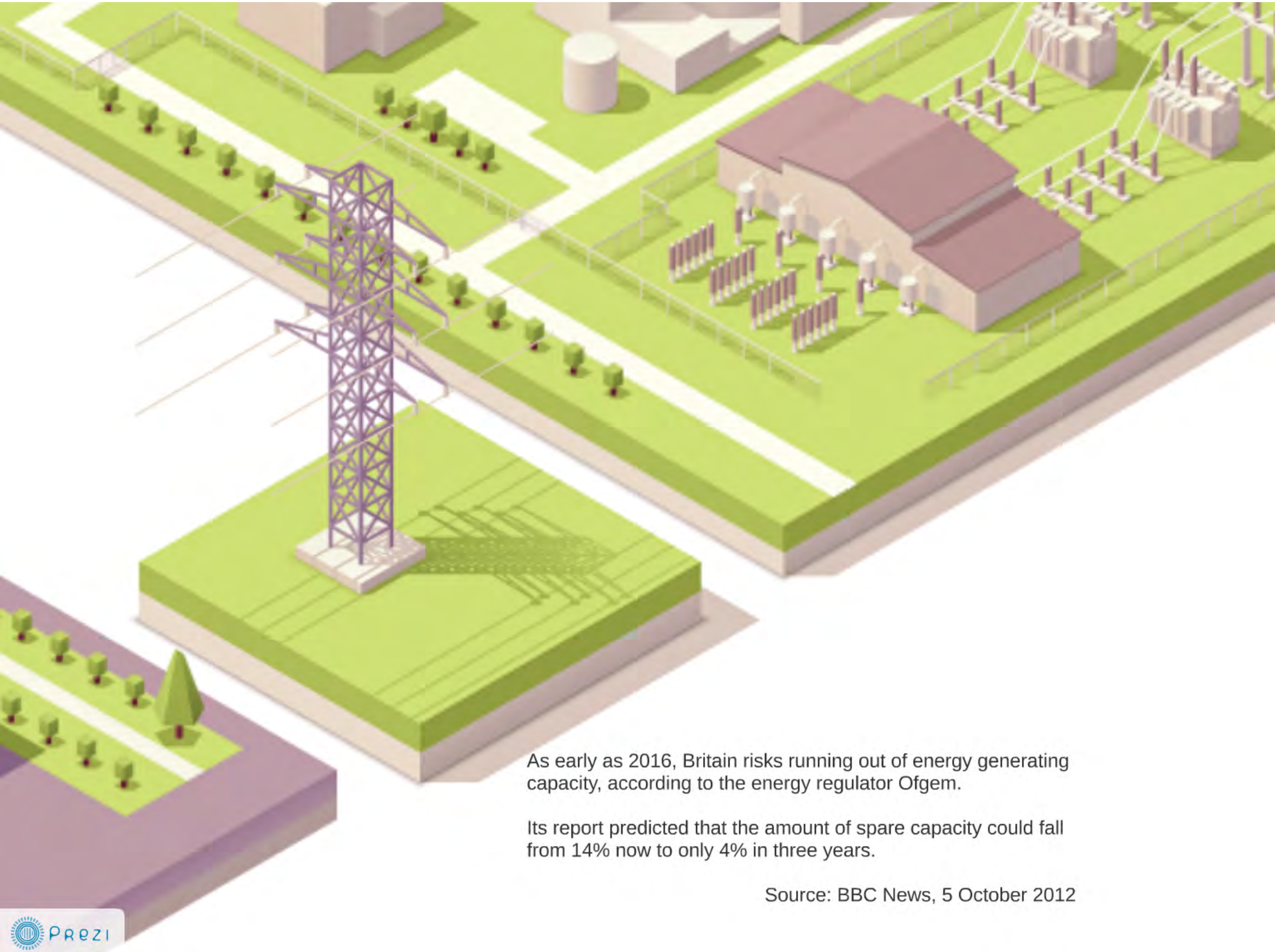
LED street lights with sensors to optimize power usage while creating a BLE mesh network for other BLE devices like parking meters or even municipal assets.

LED street lights with sensors to optimize power usage while creating a BLE mesh network for other BLE devices like parking meters or even municipal assets.



Connected cars connecting with parking meters, street lights, and other vehicles where the "city" offers routing alternatives to optimize traffic patterns.

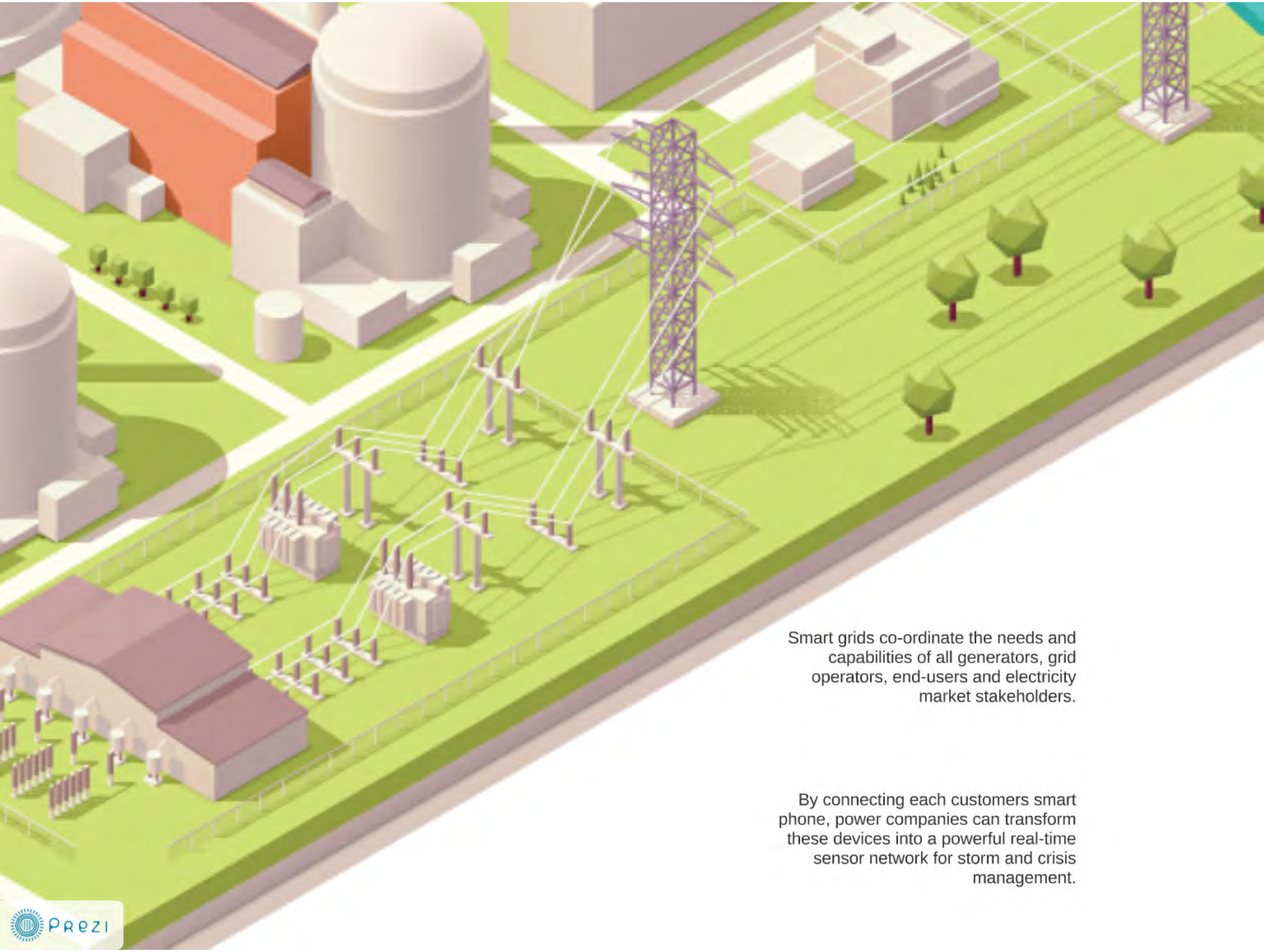
In 2018 there is estimated to be 250,000,000 connected cars on the road.



As early as 2016, Britain risks running out of energy generating capacity, according to the energy regulator Ofgem.

Its report predicted that the amount of spare capacity could fall from 14% now to only 4% in three years.

Source: BBC News, 5 October 2012

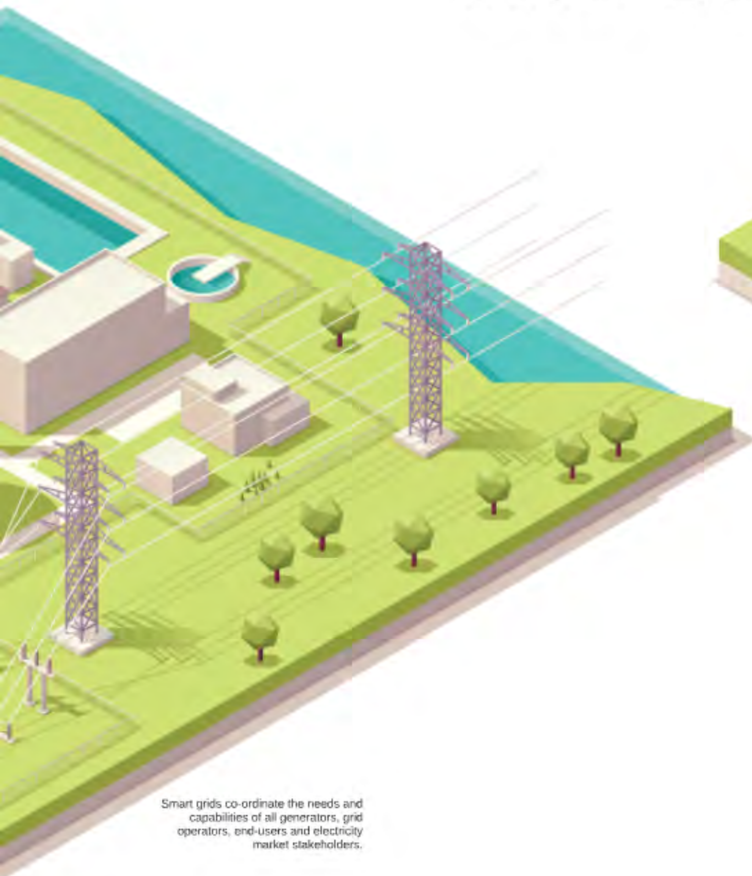


Smart grids co-ordinate the needs and capabilities of all generators, grid operators, end-users and electricity market stakeholders.

By connecting each customers smart phone, power companies can transform these devices into a powerful real-time sensor network for storm and crisis management.

A single intelligent jet engine can generate 1TB of data during a five-hour flight

The connected industry...



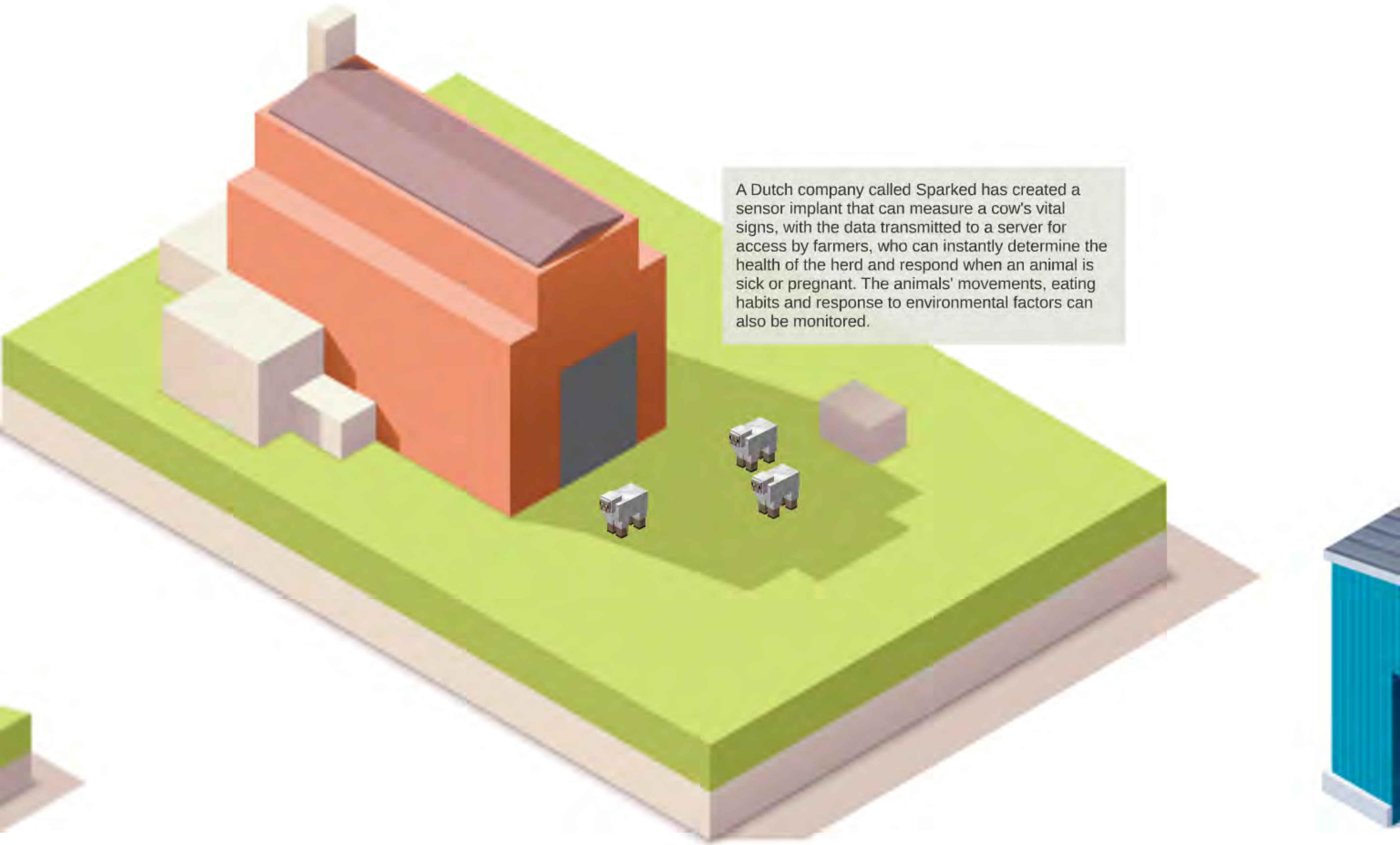
A Dutch company called Sparked has created a sensor implant that can measure a cow's vital signs, with the data transmitted to a server for access by farmers, who can instantly determine the health of the herd and respond when an animal is sick or pregnant. The animals' movements, eating habits and response to environmental factors can also be monitored.

Asset tracking of goods on the move

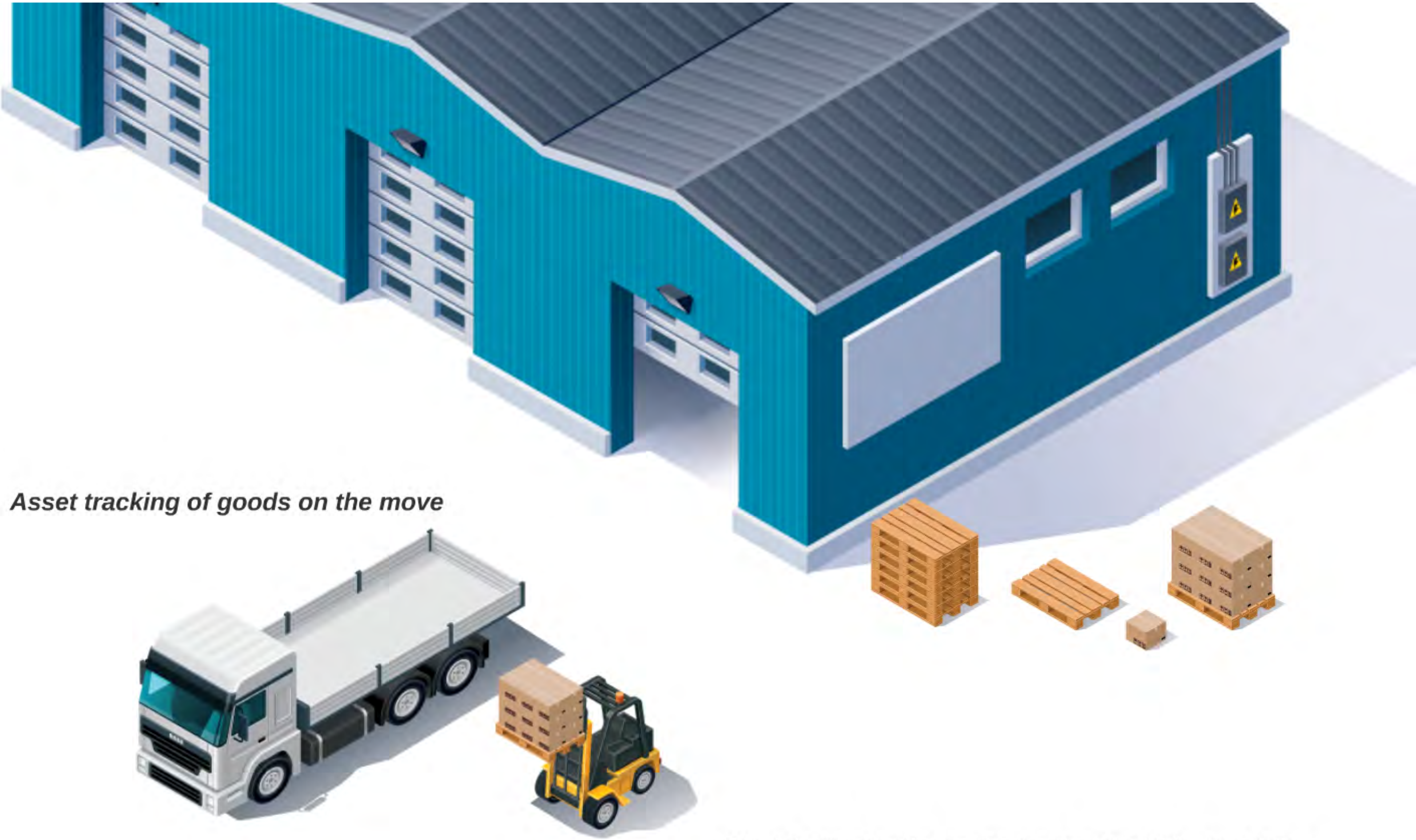
With 1.2 million trucking companies operating 15.5 million trucks in the united states, trucking is basically the countries commercial circulatory system.

Smart grids co-ordinate the needs and capabilities of all generators, grid operators, end-users and electricity market stakeholders.

By connecting each customers smart phone, power companies can transform these devices into a powerful real-time sensor network for storm and crisis management.



A Dutch company called Sparked has created a sensor implant that can measure a cow's vital signs, with the data transmitted to a server for access by farmers, who can instantly determine the health of the herd and respond when an animal is sick or pregnant. The animals' movements, eating habits and response to environmental factors can also be monitored.



Asset tracking of goods on the move

With 1.2 million trucking companies operating 15.5 million trucks in the united states, trucking is basically the countries commercial circulatory system.

A single intelligent jet engine can generate
1TB of data during a five-hour flight

And so many more use cases...

- Machine-to-machine communication
- Machine-to-infrastructure communication
- Connected Defense
- Connected Schools
- Telemedicine: remote or real-time pervasive monitoring of patients, diagnosis and drug delivery
- Asset tracking of goods on the move
- Automatic traffic management
- Remote security and control
- Environmental monitoring and control
- "Smart" applications, including cities, water, agriculture, buildings, grid, meters, broadband, cars, appliances, tags, animal farming and the environment, to name a few

Rushing into the Internet of Things

Most firms plan to deploy IoT in the near future. But what does it really mean?

When are you most likely to implement an 'Internet of Things' solution?

- We already have an IoT solution in place – 15%
- In the next 12 months – 28%
- In the next 1-2 years – 25%
- In the next 2-5 years – 14%
- Do not plan to implement IoT in the long term – 7%
- No sure – 11%

I would argue that they really do not understand IoT.

Source: Forrester Consulting/Zebra Technologies, June 2012

Building the Internet of Things

1. Devices must be able to discover each other, both through an address book, or ephemerally in an ad-hoc nature as devices encounter each other in the real world. The later will be truer.

2. Devices need to be able to communicate with each other directly, in a peer environment, or via the cloud connection.

3. Device data must then be collected and sent to the cloud, and stay synchronized with each other, even when constantly disconnecting and reconnecting.

4. Within the cloud, there will be actors that will need to operate on the device data for operational, situational and application purposes.

5. The actors in the cloud will need to communicate back to each device to take action and potentially change the behavior of the device. The ability to close the loop will be required.

IoT Technologies and Protocols

There are many protocols that can be used when building the Internet of Things. Several of them are widely adopted and some with at least 10 implementations each. All need to allow for a continuous near real time communications.

- Bluetooth LE - Quickly becoming a ubiquitous protocol in every consumer device.
- Zigbee - Not available in consumer smart devices (i.e. phones)
- DDS - A fast bus for integrating intelligent machines
- MQTT - A protocol for collecting device data and communicating it to servers
- XMPP - A protocol best for connecting devices to people, a special case of the device to server pattern, since people are connected to the servers
- AMQP - A queuing system designed to connect servers to each other
- Protocol Buffers - A way of encoding structured data in an efficient yet extensible format. Google uses Protocol Buffers for almost all of its internal RPC protocols and file formats.

The Real Challenges of Building the IoT

This is all fine and dandy, but there are some very real challenges:

IoT Technologies and Protocols

There are many protocols that can be used when building the Internet of Things. Several of them are widely adopted and some with at least 10 implementations each. All need to allow for a continuous near real time communications.

- Bluetooth LE - Quickly becoming a ubiquitous protocol in every consumer device.
- Zigbee - Not available in consumer smart devices (i.e. phones)
- DDS - A fast bus for integrating intelligent machines.
- MQTT - A protocol for collecting device data and communicating it to servers.
- XMPP - A protocol best for connecting devices to people, a special case of the device to server pattern, since people are connected to the servers.
- AMQP - A queuing system designed to connect servers to each other.
- Protocol Buffers - A way of encoding structured data in an efficient yet extensible format. Google uses Protocol Buffers for almost all of its internal RPC protocols and file formats.

Identity

- Devices are not people. People are not always people for that matter.
- Today, there are 1.5 billion smart devices in the world (iPhone, Android, etc).
- For privacy reasons, these devices no longer publish a unique way to identify them. This means there is no way to establish a stable identify for 1.5 billion devices that will survive a reset.
- This is a problem.

Discovery

- How do you build an address book of 50 billion devices?
- There will still be an address book of known devices.
- The world of tomorrow will be about discovering devices around you. This means mesh, or fine-grained proximity detection.
- What about privacy?
- What about security – can I trust that light pole?

Advertising

- Even with advertising what you have and what you are looking for, how do you advertise this to 50 billion other things in real-time?

Connect

Not all "Things" are created equal:

- Almost all wearable sensors are not network attached, but are connected by BLE, Zigbee, etc. This will require a proxy to route onto the network. Only BLE is present on consumer devices.
- Legacy network attached devices and semi-smart network attached devices that are not first class citizens in the cloud. How do you connect these?
- Smart devices are "first class citizens of the cloud"

Collect / Collaborate / Communicate

- The challenges of mobile devices:
  - Battery
  - Cellular networks
  - The promise of always connected, the reality of always trying to connect

- When collaborating with a heterogeneous group of devices of different class, how do you synchronize them and stay synchronized when everything is constantly appearing and disappearing in real-time?

- Not all devices can afford TCP/IP or fat packets. Matter of fact, almost every industrial application will need to be very thin.
  - Many devices are UDP.
  - Too costly to have a high throughput sensor network that requires cellular. Just not going to happen.
  - Will require sensor hubs and gateways

Analyze / Visualize / Operate

- Taking device data input in real-time and analyze and visualize the data to create operational and situationally intelligent systems.

- Move from Situational Awareness, to Situational Intelligence, to Situational Optimization -- the virtuous circle.

Collect / Collaborate / Communicate

Optimize Behavior and Communicate Change

- Creating a closed loop system where situational and operational awareness and intelligence is transformed into action and communicated back to the devices to optimize behavior in real-time.

- Predictive analytics can be applying fine-tune controls, turning big data into actionable intelligence.

The Real Challenges of Building the IoT

This is all fine and dandy, but there are some very real challenges:

IoT Technologies and Protocols

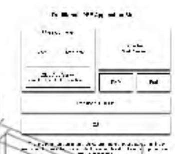
There are many protocols that can be used when building the Internet of Things. Several of them are widely adopted and some with at least 10 implementations each. All need to allow for a continuous near real time communications.

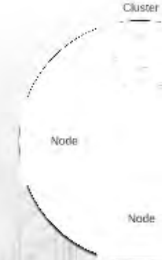
- Bluetooth LE - Quickly becoming a ubiquitous protocol in every consumer device.
- Zigbee - Not available in consumer smart devices (i.e. phones)
- DDS - A fast bus for integrating intelligent machines.
- MQTT - A protocol for collecting device data and communicating it to servers.
- XMPP - A protocol best for connecting devices to people, a special case of the device to server pattern, since people are connected to the servers.
- AMQP - A queuing system designed to connect servers to each other.
- Protocol Buffers - A way of encoding structured data in an efficient yet extensible format. Google uses Protocol Buffers for almost all of its internal RPC protocols and file formats.

Why Erlang?

- The Internet of Things is fundamentally a network and routing problem.
- Not all network-attached things are “smart” and can become a first class citizen.
- Not all things are network attached and will need a proxy.
- The monolithic enterprise block architecture is dead, long live **distributed** light-weight processes.
- The Internet of Things requires five nines.
- The Internet of Things requires low predictable latency.
- The Internet of Things needs to be operationally easy.
- The Internet of Things is about device communication and interchange of binary data.
- The Internet of Things is about processing logic in the cloud. Many devices are not capable of computationally complex operations.

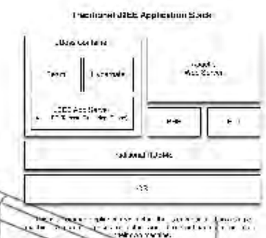
An example Codec, just 150 lines...



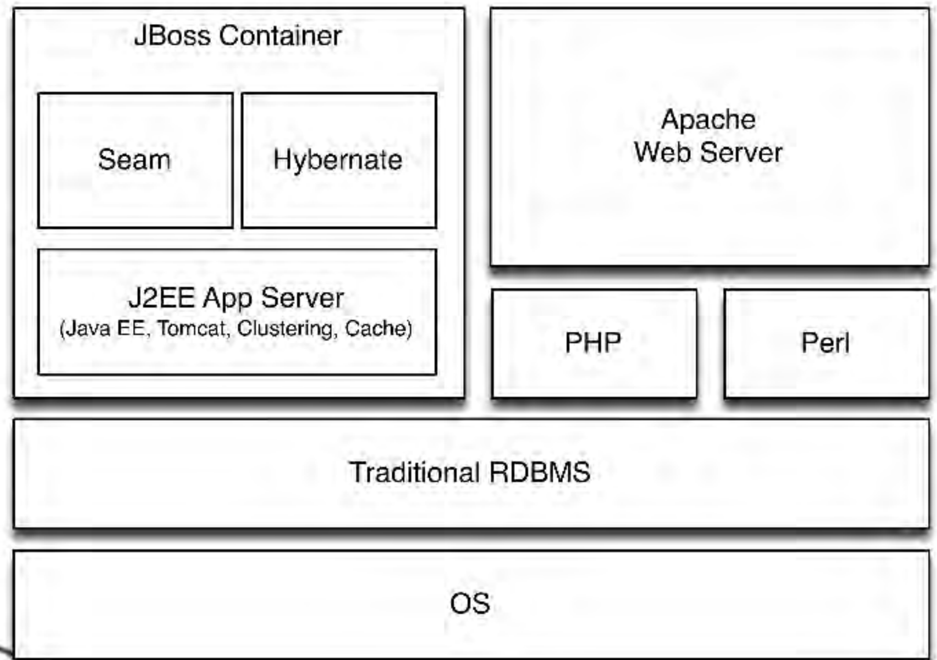


Why Erlang?

- The Internet of Things is fundamentally a network and routing problem.
- Not all network-attached things are “smart” and can become a first class citizen.
- Not all things are network attached and will need a proxy.
- The monolithic enterprise block architecture is dead, long live **distributed** light-weight processes.
- The Internet of Things requires five nines.
- The Internet of Things requires low predictable latency.
- The Internet of Things needs to be operationally easy.
- The Internet of Things is about device communication and interchange of binary data.
- The Internet of Things is about processing logic in the cloud. Many devices are not capable of computationally complex operations.

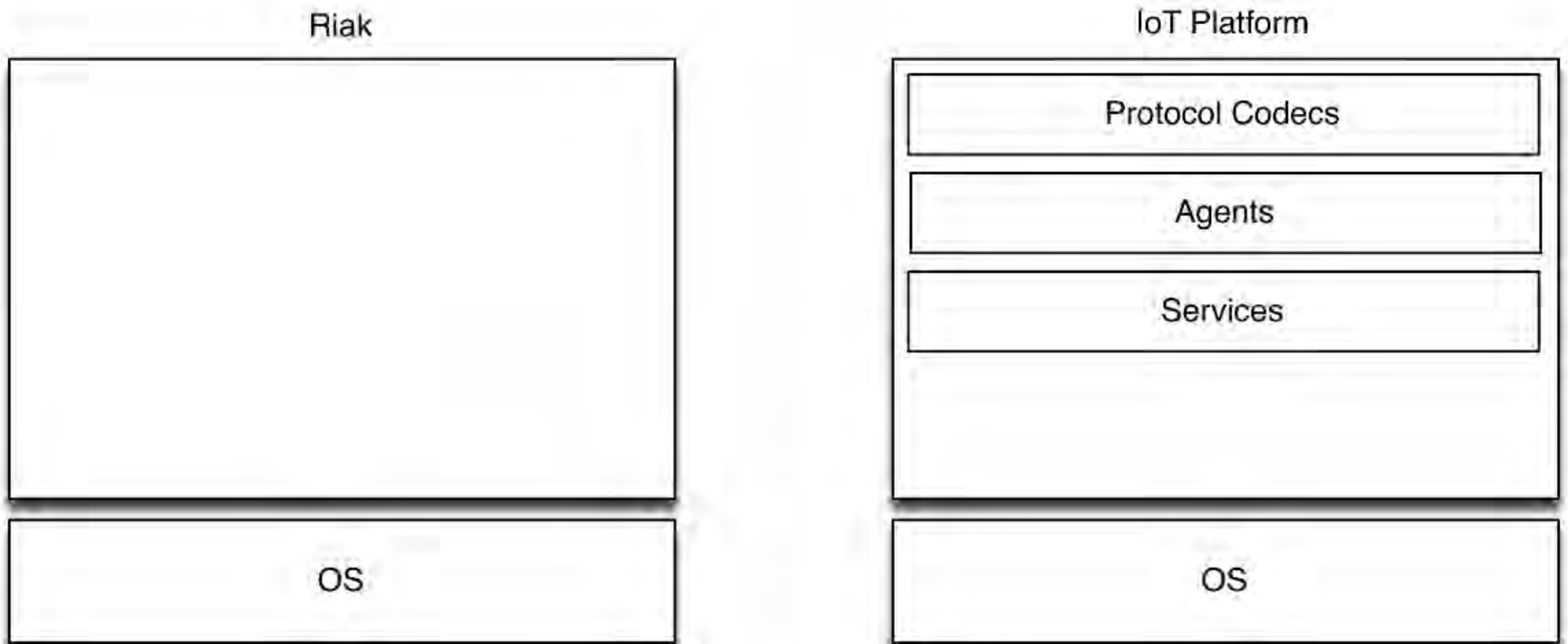


Traditional J2EE Application Stack



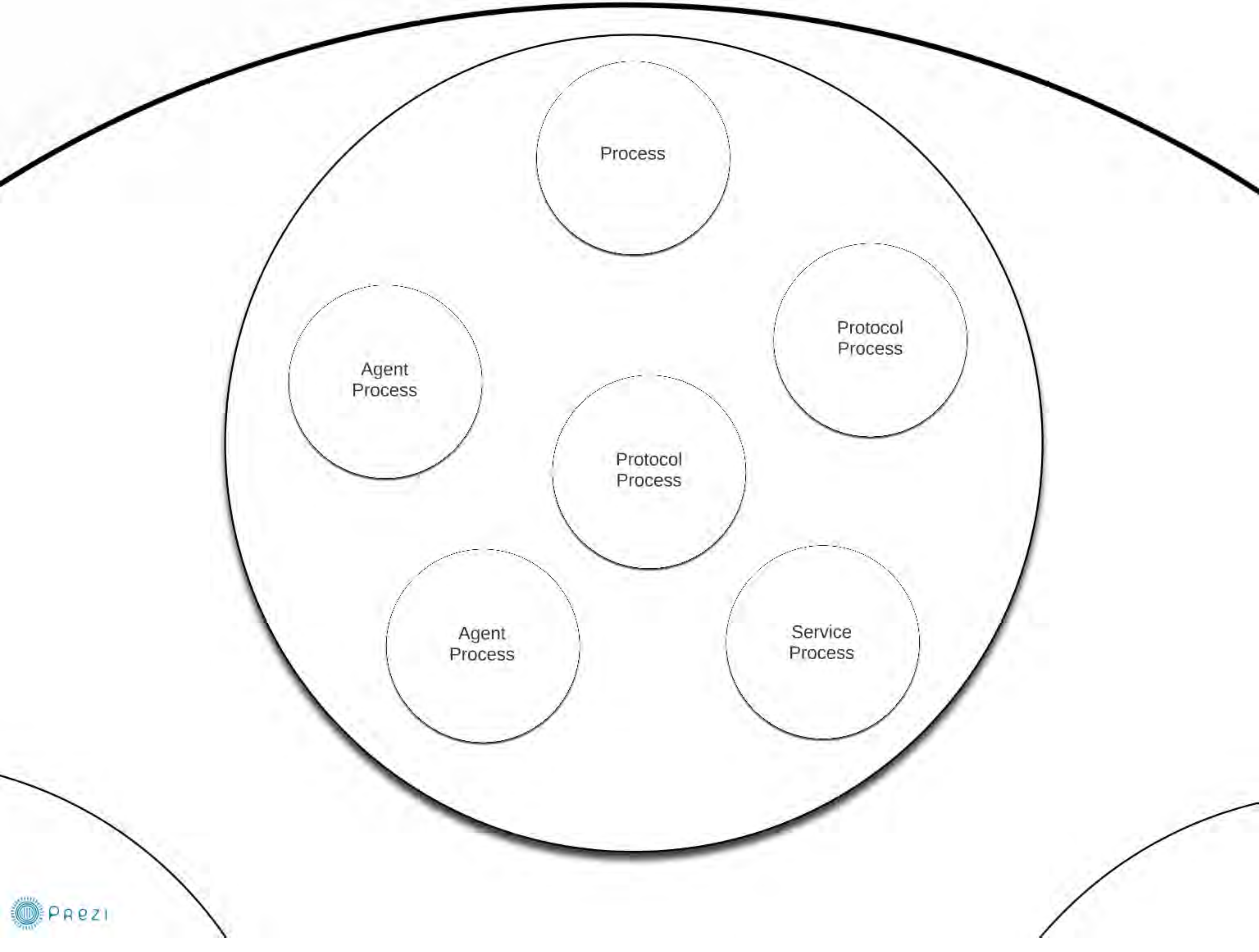
This is a common appliance architecture that is encapsulated on a single machine. A more enterprise architecture would break out each component onto their own machine.

The IoT Stack

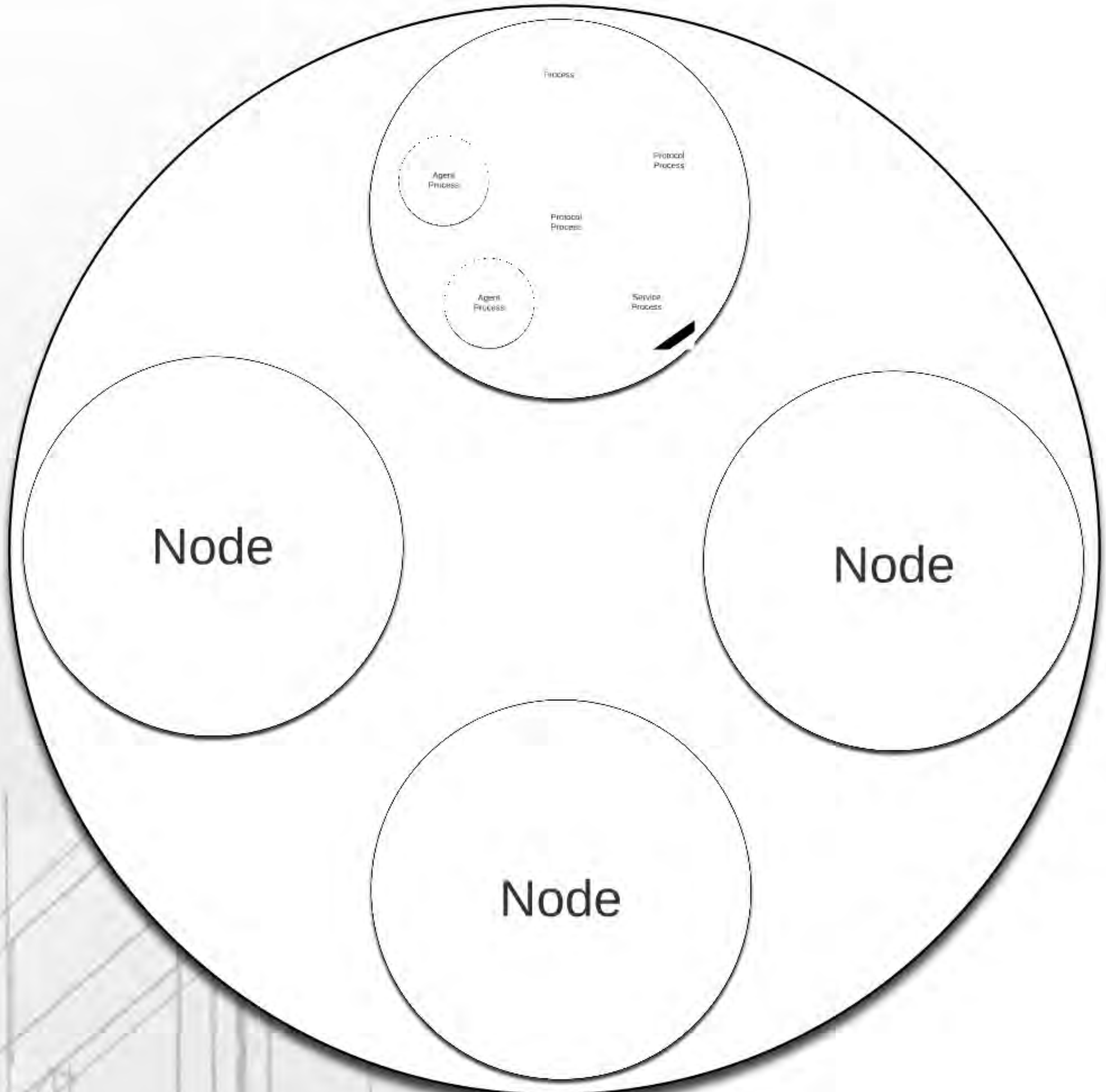


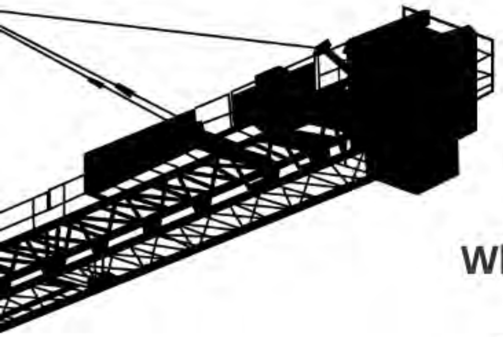
The IoT Platform has two types of machines, one running a IoT node, the other running a Riak node. A Riak cluster is comprised of a minimum of 5 nodes. The IoT cluster is comprised of a minimum of 2 nodes.

Process


Process
Agent
Process
Protocol
Process
Protocol
Process
Agent
Process
Service
Process

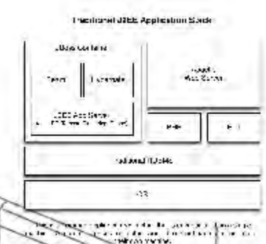
Cluster


Process
Agent Process
Protocol Process
Protocol Process
Agent Process
Service Process
Node
Node
Node



Why Erlang?

- The Internet of Things is fundamentally a network and routing problem.
- Not all network-attached things are “smart” and can become a first class citizen.
- Not all things are network attached and will need a proxy.
- The monolithic enterprise block architecture is dead, long live **distributed** light-weight processes.
- The Internet of Things requires five nines.
- The Internet of Things requires low predictable latency.
- The Internet of Things needs to be operationally easy.
- The Internet of Things is about device communication and interchange of binary data.
- The Internet of Things is about processing logic in the cloud. Many devices are not capable of computationally complex operations.



An example Codec, just 150 lines...

```erlang
%% @doc Codec for the Morey MC-1/MC-3 UDP protocol.
%% @reference
%%   "Morey: Morey MC-1/MC-3 Reference Manual" Rev. 01.00.34.

-module(morey_mc_msg).

-export([ parse_header/1
        , parse_elements/1
        , encode/1
        , encode_ack/1
        , encode_elements/1 ]).

-include_lib("morey_mc/include/morey_mc.hrl").
-define(MSG_ACK, 16#22).

-spec parse_header(binary()) -> undefined | {#morey_mc_header{}, binary()}.
parse_header(B) ->
    case B of
        << ProtocolVersion
        , DeviceType
        , Reserved
        , MessageType
        , MessageLength:16
        , MessageID:16
        , UDID:64
        , ServerField:64
        , Rest/binary >> when byte_size(Rest) =:= MessageLength andalso
                               ProtocolVersion =:= 21 ->
          {#morey_mc_header{ protocol_version = ProtocolVersion
                           , device_type = DeviceType
                           , reserved = Reserved
                           , message_type = MessageType
                           , message_id = MessageID
                           , udid = UDID
                           , server_field = ServerField },
           Rest};
        _ ->
          undefined
    end.

-spec parse_elements(binary()) -> [mc_msg_element()].
parse_elements(<<>>) ->
    [];
parse_elements(B) ->
    {V, B1} = parse_element(B),
    [V | parse_elements(B1)].

-spec parse_element(binary()) -> {mc_msg_element(), binary()}.
parse_element(<<ID:4, EID:12, Rest/binary>>) ->
    {V, Rest1} = parse_element(ID, Rest),
    {{EID, V, ID, morey_mc_tables:lookup(EID)}, Rest1}.

-spec parse_element(0..15, binary()) -> {mc_msg_value(), binary()}.
parse_element(?T_ZERO, Rest) ->
    {null, Rest};
parse_element(?T_UINT32, <<V:32, Rest/binary>>) ->
    {V, Rest};
parse_element(?T_INT32, <<V:32/signed, Rest/binary>>) ->
    {V, Rest};
parse_element(?T_UINT16, <<V:16, Rest/binary>>) ->
    {V, Rest};
parse_element(?T_INT16, <<V:16/signed, Rest/binary>>) ->
    {V, Rest};
parse_element(?T_UINT8, <<V:8, Rest/binary>>) ->
    {V, Rest};
parse_element(?T_INT8, <<V:8/signed, Rest/binary>>) ->
    {V, Rest};
```

n example Codec, just 150 lines

```
%% @doc Codec for the Morey MC-1/MC-3 UDP protocol.
%% @reference
%%   "Morey: Morey MC-1/MC-3 Reference Manual" Rev. 01.00.14.

-module(morey_mc_msg).

-export([ parse_header/1
        , parse_elements/1
        , encode/1
        , encode_ack/1
        , encode_elements/1 ]).

-include_lib("morey_mc/include/morey_mc.hrl").
-define(MSG_ACK, 16#02).

-spec parse_header(binary()) -> undefined | {#morey_mc_header{}, binary()}.
parse_header(B) ->
    case B of
        << ProtocolVersion
        , DeviceType
        , Reserved
        , MessageType
        , MessageLength:16
        , MessageID:16
        , UDID:64
        , ServerField:64
        , Rest/binary >> when byte_size(Rest) =:= MessageLength andalso
                              ProtocolVersion =:= 21 ->
            {#morey_mc_header{ protocol_version = ProtocolVersion
                             , device_type = DeviceType
                             , reserved = Reserved
                             , message_type = MessageType
                             , message_id = MessageID
                             , udid = UDID
                             , server_field = ServerField },
             Rest};
        _ ->
            undefined
    end.

-spec parse_elements(binary()) -> [mc_msg_element()].
parse_elements(<<>>) ->
    [];
parse_elements(B) ->
    {V, B1} = parse_element(B)
```

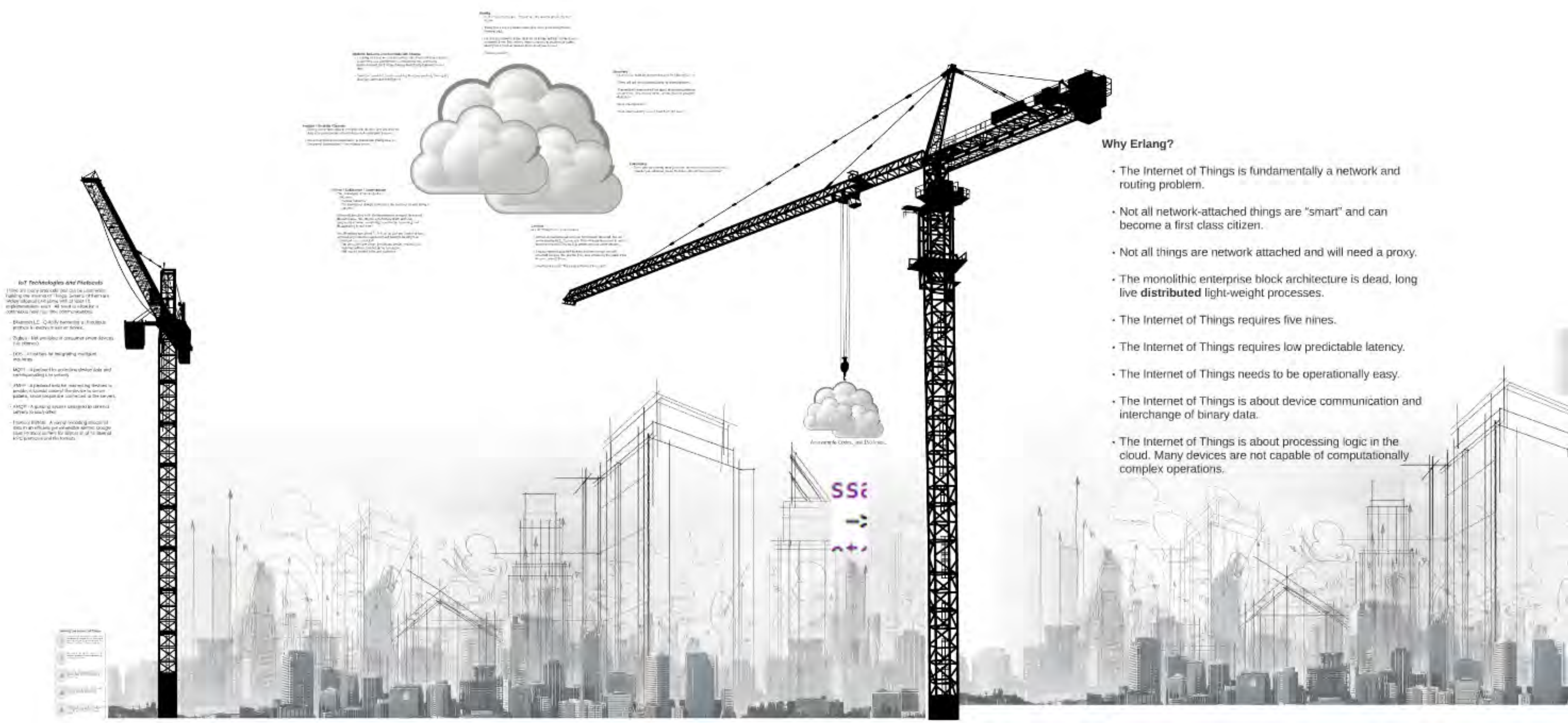
```
    {V, B1} = parse_element(B),
    [V | parse_elements(B1)].

-spec parse_element(binary()) -> {mc_msg_element(), binary()}.
parse_element(<<ID:4, EID:12, Rest/binary>>) ->
    {V, Rest1} = parse_element(ID, Rest),
    {{EID, V, ID, morey_mc_tables:lookup(EID)}, Rest1}.

-spec parse_element(0..15, binary()) -> {mc_msg_value(), binary()}.
parse_element(?T_ZERO, Rest) ->
    {null, Rest};
parse_element(?T_UINT32, <<V:32, Rest/binary>>) ->
    {V, Rest};
parse_element(?T_INT32, <<V:32/signed, Rest/binary>>) ->
    {V, Rest};
parse_element(?T_UINT16, <<V:16, Rest/binary>>) ->
    {V, Rest};
parse_element(?T_INT16, <<V:16/signed, Rest/binary>>) ->
    {V, Rest};
parse_element(?T_UINT8, <<V:8, Rest/binary>>) ->
    {V, Rest};
parse_element(?T_INT8, <<V:8/signed, Rest/binary>>) ->
    {V, Rest};
parse_element(?T_POINT, <<V1:32/signed, V2:32/signed, Rest/binary>>) ->
    {{V1, V2}, Rest};
parse_element(?T_ARRAY, <<ID, Count:16, Rest/binary>>) ->
    parse_array(ID, Count, [], Rest);
parse_element(?T_BOOL, <<B, Rest/binary>>) ->
    {B =/= 0, Rest};
parse_element(?T_STRING, <<Len:16, Rest/binary>>) ->
    <<S:Len/binary, Rest1/binary>> = Rest,
    {S, Rest1};
parse_element(?T_REPORT, <<EID:16, Count:16, Rest/binary>>) ->
    WCount = 2 * Count,
    <<B:WCount/binary, Rest1/binary>> = Rest,
    Vs = [V || <<V:16>> <= B],
    {{report, EID, Vs, morey_mc_tables:lookup(EID)}, Rest1}.

parse_array(ID, Count, Acc, Rest) when Count > 0 ->
    {V, Rest1} = parse_element(ID, Rest),
    parse_array(ID, Count - 1, [V | Acc], Rest1);
parse_array(ID, 0, Acc, Rest) ->
    {{lists:reverse(Acc), ID}, Rest}.

-spec encode({#morey_mc_header{}, binary()}) -> binary().
encode({#morey_mc_header{ protocol_version = ProtocolVersion
                        , device_type      = DeviceType
                        , reserved         = Reserved
                        , message_type     = MessageType
                        , message_id       = MessageID
                        , udid             = UDID
                        , server_field     = ServerField },
        Body}) ->
    << ProtocolVersion
    , DeviceType
    , Reserved
    , MessageType
    , (byte_size(Body)):16
    , MessageID:16
    , UDID:64
    , ServerField:64
    , Body/binary >>.

-spec encode_ack(#morey_mc_header{}) -> binary().
encode_ack(Header=#morey_mc_header{ message_type = MsgType })
    when MsgType =/= ?MSG_ACK ->
    encode({Header#morey_mc_header{ message_type = ?MSG_ACK }, <<>>});
encode_ack(_Header) ->
    <<>>.

-spec encode_elements([mc_msg_element()]) -> binary().
encode_elements(Elems) ->
    << <<(encode_element(Elem))/binary>> || Elem <- Elems >>.
```

```
    {{lists:reverse(Acc), ID}, Rest}.

-spec encode({#morey_mc_header{}, binary()}) -> binary().
encode({#morey_mc_header{ protocol_version = ProtocolVersion
                         , device_type      = DeviceType
                         , reserved         = Reserved
                         , message_type     = MessageType
                         , message_id       = MessageID
                         , udid             = UDID
                         , server_field     = ServerField },
        Body}) ->
    << ProtocolVersion
    , DeviceType
    , Reserved
    , MessageType
    , (byte_size(Body)):16
    , MessageID:16
    , UDID:64
    , ServerField:64
    , Body/binary >>.

-spec encode_ack(#morey_mc_header{}) -> binary().
encode_ack(Header=#morey_mc_header{ message_type = MsgType })
    when MsgType =/= ?MSG_ACK ->
    encode({Header#morey_mc_header{ message_type = ?MSG_ACK }, <<>>});
encode_ack(_Header) ->
    <<>>.

-spec encode_elements([mc_msg_element()]) -> binary().
encode_elements(Elems) ->
    << <<(encode_element(Elem))/binary>> || Elem <- Elems >>.

-spec encode_element(mc_msg_element()) -> binary().
encode_element({EID, V, ID, _Info}) ->
    << ID:4, EID:12, (encode_element(ID, V))/binary >>.

encode_element(?T_ZERO, null) ->
    <<>>;
encode_element(?T_UINT32, V) ->
    <<V:32>>;
encode_element(?T_INT32, V) ->
    <<V:32/signed>>;
encode_element(?T_UINT16, V) ->
    <<V:16>>;
encode_element(?T_INT16, V) ->
    <<V:16/signed>>;
encode_element(?T_UINT8, V) ->
    <<V>>;
encode_element(?T_INT8, V) ->
    <<V:8/signed>>;
encode_element(?T_POINT, {V1, V2}) ->
    <<V1:32/signed, V2:32/signed>>;
encode_element(?T_ARRAY, {Vs, ID}) ->
    << ID
     , (length(Vs)):16
     , << << (encode_element(ID, V))/binary >> || V <- Vs >>/binary
     >>;
encode_element(?T_BOOL, B) ->
    <<(case B of true -> 1; false -> 0 end)>>;
encode_element(?T_STRING, V) ->
    <<(byte_size(V)):16, V/binary>>;
encode_element(?T_REPORT, _V) ->
    <<>>.
```

The Internet of Things

The Real Challenges of Building the IoT

This is all fine and dandy, but there are some very real challenges:



Why Erlang?

- The Internet of Things is fundamentally a network and routing problem.
- Not all network-attached things are "smart" and can become a first class citizen.
- Not all things are network attached and will need a proxy.
- The monolithic enterprise block architecture is dead, long live **distributed** light-weight processes.
- The Internet of Things requires five nines.
- The Internet of Things requires low predictable latency.
- The Internet of Things needs to be operationally easy.
- The Internet of Things is about device communication and interchange of binary data.
- The Internet of Things is about processing logic in the cloud. Many devices are not capable of computationally complex operations.