Lessons from the Internet

David D. Clark
Senior Research Scientist

Computer Science and Artificial Intelligence Laboratory (CSAIL)
Massachusetts Institute of Technology

INTERNET (OF THINGS)

This topic is about how “things” are networked together.

Why say *Internet* of Things?

- It is not the first buzzword for this space…
- Answer: it looks like the only game in town.

But is Internet the right system for “things”?

The suitability of the Internet—a recurring debate.

- The wrong question: Can a thing run the Internet protocols?

This topic is concerned with what we can learn from the history of the Internet that will offer insights about how to make things into a “network of things”.

THE INTERNET—GOALS AND FLAWS

Goals:

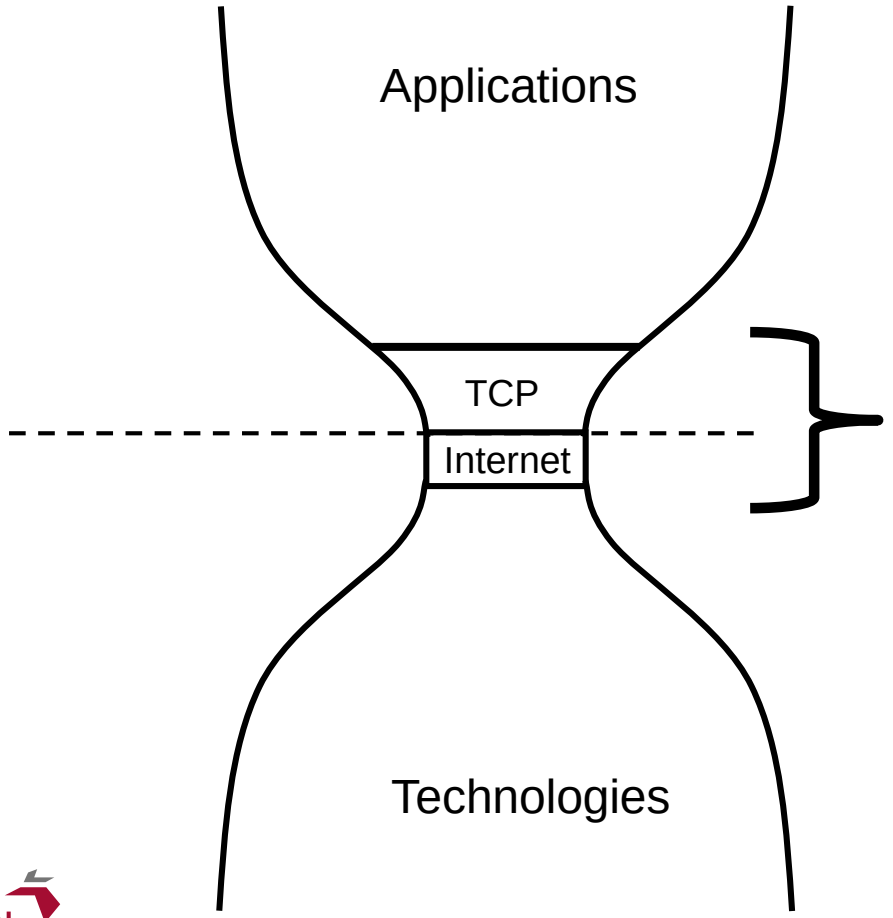
- Constraints that de-constrain
  - The “Internet hourglass”
- Global reach
- Heterogeneous interconnection
- Generality

Flaws:

- Network management
- Security
- Mobility
- Longevity
- Application design.

And some general lessons to close

THE INTERNET HOURGLASS MODEL



To a user, this whole picture is “the Internet”.

To an Internet engineer, this bit is the Internet.

These layers defines how data moves across the net and is reassembled at the receiver.

The stuff above is “just” the applications.

The stuff below is the network technology.

WHY AN HOURGLASS?

We draw the central part narrow to suggest that at this point, there is less diversity in the architecture.

The IP layer is the specification to which all parties must agree.

- Any application that can operate on top of the service defined by the IP layer can be deployed on the Internet.
- Any technology that can carry bytes from one point to another can be used to provide the IP service.

So by constraining in a very specific way the narrow waist of the picture, we decouple what happens above from what happens below.

- The constraint that de-constrains.

THIS ARCHITECTURE HAS LIMITS

If the technology below the narrow waist has special features (e.g., broadcast) the applications cannot easily take advantage of it.

If the technology below the narrow waist has limitations, there is no direct way for the application to discover them.

- It can measure what is happening and try to guess, but there is no cross-layer optimization interface.

The wireless community has argued that the limits of this design outweigh its benefits, and for impaired networks it would be better to design apps knowing exactly what network technology they will be using.

I will return to IoT and the hourglass model.

The next segment: Additional goals

GOALS OF THE INTERNET

Heterogeneous interconnection:

- The Internet was intended to connect computers of different sorts together.
  - 9 bit bytes, 36 bit words, for example.
- The interconnection provided by the Internet has driven a lot of this low-level heterogeneity out of the ecosystem.

Global reach:

- The conception of the Internet was that any computer might have a reason to connect to any other computer.
  - All machines on the Internet were directly addressable by each other.
  - Empowered new sorts of applications.
  - Also made various sorts of security attacks much easier to carry out.
  - Does the IoT need this sort of reach?

The next segment: generality

GENERALITY

The Internet was designed to hook computers together.

- Computers are general purpose devices.
- So the Internet is also a general purpose platform.

It has no idea what it is being used for.

- In contrast to the telephone system or the (original) cable TV system.

Things, in contrast, are fixed function.

- But they (may) hook to general purpose servers running services that support things.

GENERALITY: THE POWER AND COST

Because the Internet is a general platform:

Because it does not know about the applications that run over it:

- Another term: the Internet is “open”.

It is a great platform for innovation.

- New ideas can (and are) tried out every day, without having to change the Internet itself.

It separates the role of the network provider from that of the application provider.

Because it is general (and “dumb”) it cannot help with:

- Management
- Application level security
- And so on…

THINGS: TWO VIEWS OF GENERALITY

First view: Things are miniature general devices.

- Like a general purpose computer but smaller.
- Run different applications
- Connect into different higher-level systems.

Second view: They are fixed function and fixed purpose.

- In particular, they are intended to be used in a specified context.
- Thermostat, industrial controller, heart monitor.

What does it mean for a device to be “in a context”?

- By analogy, the email system or the web is an application context on the Internet.
  - At design time, the context must be specified.
  - At run time, the device must be introduced into the context.
- This take us to the topic of “management”. The first of my flaws with the Internet.

MANAGEMENT

The term refers to the setup, oversight, and diagnosis of systems.

- A human component.

Applies to the network itself, and to applications and services running over the net.

This topic received little attention in the early days of the Internet.

- Not surprising—we had no framework to think about it.
- Just moving data was hard enough.

That tradition continues today.

- Academics focus on data transport—performance, patterns of communication, etc.
- Leads to a misguided focus for things:
  - Should a thing run Internet protocols?

STATE OF INTERNET MANAGEMENT

Poor.

Our best tools for network diagnosis:

- Ping and traceroute.
- It is an embarrassment that normal users (sort of) know how to use these.

If anything, worse at application level:

- Mail system failures—how diagnose and correct?
- Web site failures

Both users and network operators deal daily with inexplicable failures and faults.

- Think about setting up a home network, configuring a printer, setting up security parameters.

NOW CONSIDER THINGS

Computers are somewhat powerful.

- Have displays, keyboards, etc.

Things may be very simple.

- No display. No keyboard. Perhaps a LED or two.
- How to connect a switch to a light.

How can a user diagnose what is wrong if a thing does not seem to be working properly?

- Try to connect to it with a web browser?
- Outsource to a service provider?

How to fix a thing:

- Power cycle it or replace the batteries.
- Replace it.

LESSON: MANAGEMENT

The problems of management are poorly dealt with in the architecture of the Internet, and the issues will get more pressing:

- Things get more simple and less able to help out with configuration and fault diagnosis.

When we think about “Internet of Things”:

- Don’t ask about how data gets transferred.
- Ask about the architecture for setup and diagnosis.
- Saying “Internet of Things” tells you nothing about these issues. Dig deeper.

The next segment: Internet security.

SECURITY ON THE INTERNET

I can only discuss this briefly here.

- A very complex topic with many dimensions.
- Security *was* considered in the early days of the Internet, but the issues and requirements have greatly evolved.

The dimensions of Internet security:

- End points that trust each other are attacked while trying to communicate.
  - Solution: encryption and complex key management architecture.
- One end point attacks another.
  - Solution: cannot avoid all risky behavior—system and application must prevent harms.
  - Solution: upgrade components to correct vulnerabilities as they are detected.
- One part of the network attacks another.
  - Solution: better design of technical mechanisms to tolerate presence of untrustworthy actors.

SECURITY AND CONTEXT

Earlier, I used the word “context” to describe the environment in which a thing (or an application) exists.

An important aspect of defining a context is its security parameters.

- The Web: the security context is provided by the Certificate Hierarchy System (flawed).
- Email: a poor security context, although there are proposals for more secure email.
- Centralized apps (think Facebook): the security context is provided by the app designer.

What is the security context in which a thing will exist?

- Another wrong question: can a thing encrypt its data?
- Right question: how is the security context set up?
- Brings us back to management. Security management is a major dimension of that issue.

SECURITY OF THINGS

Perhaps the problem is simpler in some ways:

- Perhaps things don't need to attempt to talk to untrustworthy end-points.
- Perhaps things only need to talk to a known set of end-points.
- Perhaps we can hide things behind a protection device (like a firewall).
  - But can a designer of a thing presume that this protection is always there?

If a thing is a fixed-function device, perhaps its security context can as well be more fixed.

- But if configuration of a thing includes typing an encryption key to a device without a keyboard, we are doomed.

How can we deal with vulnerabilities?

- Many things are so simple they cannot be patched if a flaw is discovered.

LESSON: SECURITY

Saying that a thing can encrypt its data is not enough to make a system of things secure.

Need a “security context” or “security architecture”.

- To bring a thing to market, there must be a given framework or context within which that thing is to be positioned.
- Today, many consumer things (e.g., smart home devices) have to solve this by designing their context as part of designing the product.
  - Thermostat (Nest), Amazon Echo, Light controllers.
- Industrial control things are often designed to fit into a pre-specified but closed architecture.

The Internet only teaches us the power and cost of open systems.

- If there is any guidance about security, it will be from application contexts, not “the Internet”.

The next segment: Mobility

MOBILITY

Mobility was a part of the early Internet, but at the time we did not understand all the issues.

- We dealt with a *wireless* network, but not mobility.

How does a device maintain its identity and its reachability when it moves?

Today, both identity and reachability are defined by the IP address of a device.

- When a device moves, it has to re-establish its identity.
- Not a good design.

Some things will be very mobile.

- But they may be fixed relative to their path into the larger Internet.

Other things will never move.

WIRELESS CONNECTIVITY

Wireless connectivity is a basic challenge to the design of the Internet.

- It provides modes of communication (e.g., broadcast) that Internet protocols cannot exploit.
- The strict layering of the Internet prevents cross-layer adaptation that can greatly help to mitigate wireless impairments.
- The potential variation in connection quality is a challenge to the congestion control and error recovery algorithms in the Internet.

People use wireless to connect to the Internet anyway.

- It works well enough.
- They need to get to the Internet.

But the IoT space may be different for some wireless things.

INDUSTRY STRUCTURE

Wireless access is not just a technical issue.

Another lesson from the Internet:

- Technical standards define industry structure.
- Industry structure defines who controls the shape of a context.

Two wireless stories:

- WiFi: Edge empowerment, highly decentralized. Implies edge management.
- Cellular: Operator empowerment, more central control.
  - Does a thing buy a monthly service plan to connect?

The design of the thing can influence how it fits into these contexts.

- More than one radio? Which actor controls which radio is used?
- How mobile does each thing really need to be?

LESSON: MOBILITY

A point that will come up several times:

- Different things will have very different requirements, capabilities, and objectives.
- There is no single “Internet of Things”.

Variation:

- Simple to complex.
- Mobile vs. fixed.
- Consumer vs. industrial vs. medical vs. ….

Modes of connectivity are just one example of this diversity.

- When looking at what seem to be technical issues, look beyond the technology.

The next segment: Longevity

LONGEVITY

With respect to longevity, the Internet itself has been a great success.

- So why did I list this topic as a failure?
- Higher-level issues.

Names.

- We use Domain Names to separate a name and a location. This scheme has worked.
- But those names were designed to name computers
  - That is not really what we need to name.

Naming information.

- The web is a de facto standard,
  - And using domain names in URLs is a Bad Idea. It binds information to the computer that hosts it.

WHAT SHOULD BE NAMED?

Information:

- Names should be independent of location.
- The architectural challenge (which we ducked): how to find information as it moves.

Services:

- What the user wants to reach is a service. The user normally does not care where the service is.
- Some part of the system must map from a service name to a location.
  - What part of the system should have that responsibility?
- Example: DNS itself—an Internet addressing mode maps from service to location.
- For many higher-level services, the DNS is used to map service to location.
  - This is either a good idea or an afterthought. It ‘just happened”.

LONGEVITY FOR IOT

What sorts of elements need to have continuity (long-lived names)?

- Physical objects?
- Roles? ("Second floor thermostat")
- Data from sensors?
- Actions?

What *will* last a long time?

- Some physical devices.
  - Embedded in larger systems.
- We have no framework to deal with devices that last a long time.
  - Compare the useful life of a computer to a car, a furnace, a door lock, a house.
  - Is computerization going to make everything in our lives disposable? Upgradable?

The next segment is application level design.

APPLICATION DESIGN

I included this in my list of flaws:

- Perhaps not quite fair. We just ignored it.
- For good reason—we had no control over how applications were designed.

But we had an over-simple model:

- Applications would involve direct communication among relevant end-node computers.
- This was true for the first application—remote login (`telnet`).

The second application was email. It broke this simple model.

- Email uses servers between sender and receiver.

The web similarly started out with a simple client-server model.

- It quickly acquired content services (CDNs), security services (Certificate Authorities), etc.

What we did not initially focus on: real applications exploit services (servers).

SUPPORTING SERVICES

Services are functions that make sophisticated applications function.

Some services define the application:

- Facebook, Twitter, etc

Some are supporting components:

- Certificate Authorities (CAs) for Web security,

Some are conceptually centralized:

- Facebook, Twitter

Some are decentralized:

- Mail servers, CDNs, DNS, CAs,

INDUSTRY STRUCTURE

Just as with the Internet itself, the technical design of a application architecture (the set of services that make up the app) defines the industry structure.

Centralized apps emerge as single-firm innovation.

- Facebook, Twitter, games, …

Decentralized apps emerge from design communities with less focus on commercialization

- The Web, email.

Supporting services often end up with a more decentralized design.

- DNS, Certificate Authorities,

Some services have triggered contested designs:

- Identity systems.

EARLIER I SAID "CONTEXT"

IoT devices, especially if they are fixed function devices, will be deployed in a context.

- That is the same thing as an application-level architecture.
- It is the architecture, or context, that will define the issues of management, security, and so on.

A key question—who controls the context?

- Things can report to a centralized master: Facebook for things?
- A simple context can be server-less.
  - My light switch directly connects to my light bulb.
- The server architecture can be highly decentralized.
  - Extreme point—under the control of the individual user.

Example—the space of consumer IoT.

- Consider the smart home.

OPTIONS FOR THE SMART HOME

How to hook a light switch to a light bulb.

Option 1: introduce them directly.

Option 2: have a smart home controller local to the home.

- Pro: the things are not “on the Internet”; they only need to reach the local server. Protected.
- Pro: Control is local. (Cloud becomes “fog”).
- Con: Consumer must set up and administer the smart home controller.

Option 3: a third party administers a server in the home.

- No business model to make this happen today. Example of highly decentralized outcome.

Option 4: the control function is in the cloud.

- Pro: A third party can administer the service.
- Con: The things must go out onto the Internet to reach the service.
- Con: Control is remote.

LESSON: APPLICATION DESIGN

The development of an IoT may depend on specification at a “higher level” than the specification of the Internet.

- There may be an “hourglass”—a point of common convention—but it may be at a higher level.
- Like Internet applications, there will be more than one.

This ends the discussion of flaws or limits of the Internet design.

- Management, security, mobility, longevity, application design.

The next segment will end this topic with some general lessons.

SOME GENERAL LESSONS

Things may seem to be physical objects.

- Their real character will emerge from the context in which they are intended to operate.
- Is the hourglass model the right way to conceptualize how things will work?

Applications define the way the generality of the Internet is restricted to a specific purpose.

- Just as there are many applications on the Internet, there will be multiple contexts for things.
- There will be more than one “Internet of Things”.

Technical designs induce industry structure.

- Design challenge: do all the entities defined by the architecture have an incentive to play nice?

MORE GENERAL LESSONS

The power of the Internet was its open and general character.

- New devices and applications could be added by anyone.

Open architectures do not emerge naturally.

- Require a motivated and well-supported design group not seeking financial advantage.
- Will architectures (contexts) for things be open?
- Only with a suitable architecture will a rich space of things come to the market.

Do not underestimate the demand for generality.

THANK YOU!

David D. Clark
Senior Research Scientist

Computer Science and Artificial Intelligence Laboratory (CSAIL)
Massachusetts Institute of Technology