RIPE NCC
RIPE NETWORK COORDINATION CENTRE

RIPE Atlas and IoT

Approach, Experiences and Some Interesting Details

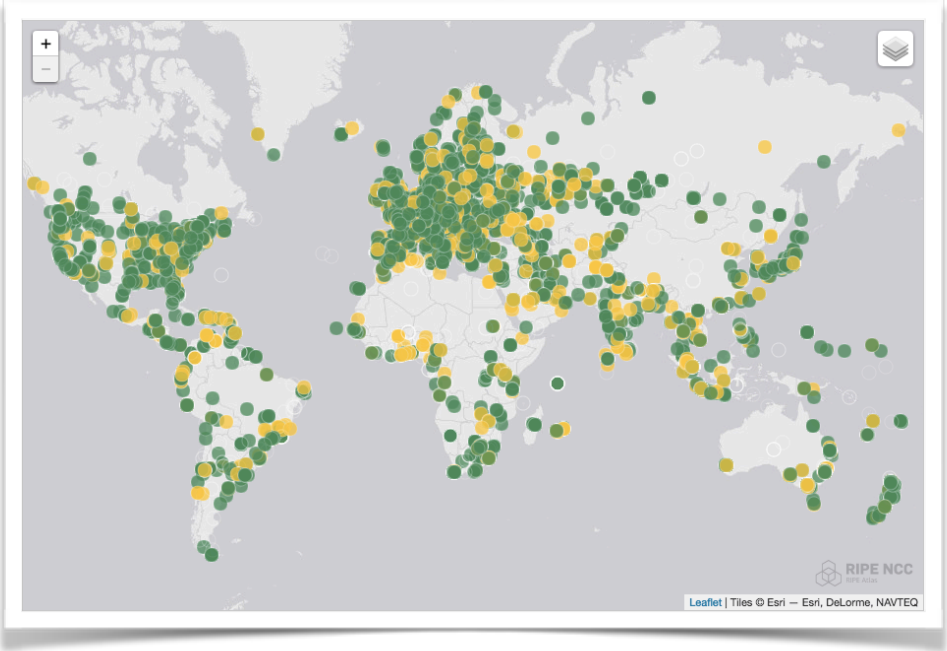
Robert Kisteleki
RIPE NCC

RIPE Atlas and IoT

- The following slides cover:
  - Some “common sense” aspects, and
  - RIPE Atlas specifics
- Most of these we thought through before we started building the network in 2010
- Some of these are experiences gained along the way or results of the system evolving
- RIPE Atlas is unique in may ways
  - What / how we do is pretty unique for sure
- Some technologies that exist today were not available when we started

Quick Update on Current Status

- Number of connected probes: 10.000+
  - Of which almost 300 are anchors
- Covered ASes: ~3.600 (IPv4), ~1.350 (IPv6)
- Collecting 5.000+ results/sec (450M+/day)



Design Principles for RIPE Atlas

- The system should scale to 100K+ probes
- Active measurements only
  - Not observing other traffic, no scanning
- Hardware vantage point
  - May involve VMs later, TBD
- Community involvement right from the start
  - Envisioned to be deployed all over the world, in all kinds of networks, by volunteers
  - So the probes have to behave well in the hosts' network

RIPE Atlas Probe Generations

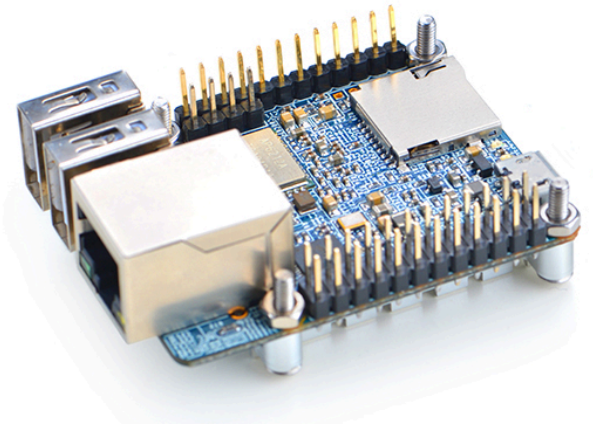
- v1 (v2)
  - Lantronix XPortPro
  - Very low power usage
  - 8 (16) MB RAM, 16MB flash
  - Runs uClinux
  - No FPU, no MMU
  - A reboot costs <15 seconds
  - An SSH connection costs ~30 seconds (!)
- Lived well beyond their anticipated life time
  - We still have ~600 + ~1400 of these up and running
  - Version 1 probes approached their technical limits

RIPE Atlas Probe Generations

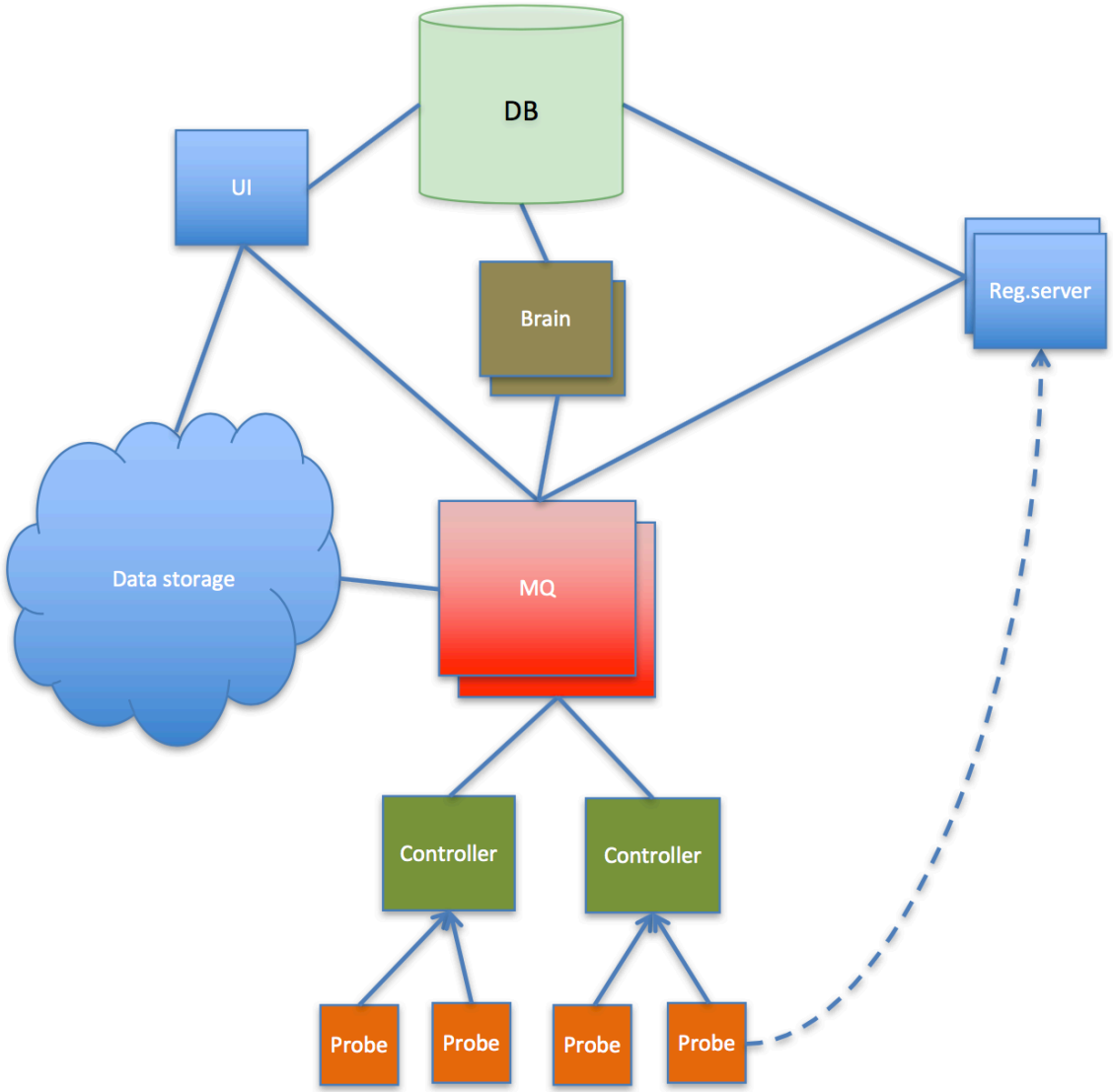
- v3 - TP-Link
  - TP-Link MR3020 + USB disk
  - 32 MB RAM, 4MB flash + 4GB USB disk
  - Can be powered over USB
  - Runs OpenWRT & Busybox
  - Off-the-shelf hardware => cheaper
  - USB disk caused more issues than anticipated

RIPE Atlas Probe Generations

- v4 - Evaluating NanoPi NEO Plus2
  - Raspberry PI “clone”
  - 1GB RAM, 8GB flash
  - Allwinner H5, Quad core Cortex A53
  - No external storage needed
  - Will run either Armbian or OpenWRT as base OS
  - Looks very capable but logistics needs work

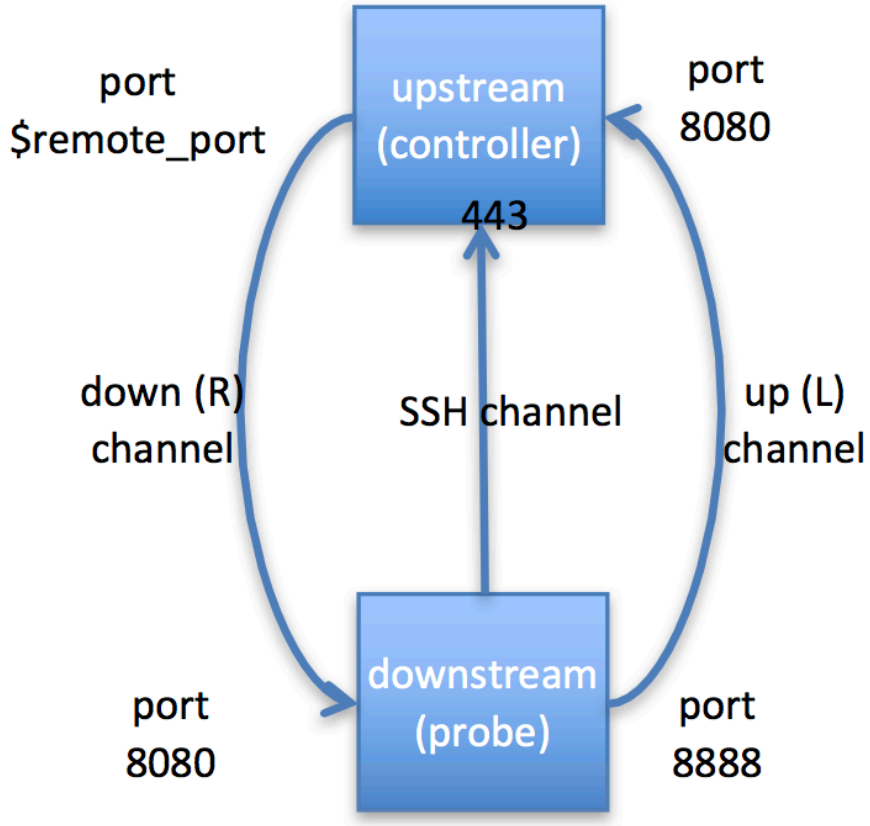


Overall Architecture



Communicating With Probes

- Happens over SSH with port forwarding
  - Plus individual SSH keys, session keys, allocated ports, …



Communicating With Probes

- OpenSSH has good control over local port forwarding, restrictions on remote port forwarding is not implemented for some reason…
  - We implemented this ourselves
  - Submitting the patch upstream was not successful

- If we built the architecture today, we'd probably use web sockets (over HTTPS) instead for bi-directional information flows

RIPE Atlas Probe Initialisation

- All probes go through an initialisation procedure by the RIPE NCC
  - Initial firmware is uploaded
  - Replaces the off-the-shelf firmware
  - Individual keys are generated, registered
  - Verifying, labeling, packaging, …

IPv6 Support

- Everything we do works over IPv6 too
  - All built-ins, User-Defined Measurements, … if we detect that IPv6 is available
  - Exception: local IP configuration is somewhat different
  - IPv4 done via DHCP, IPv6 via RA
  - Other minor differences: see later

Other Bits and Pieces

- IP discovery
  - Probes can connect from anywhere, over IPv4 or IPv6
  - We need to keep track of where they are in the network
  - Three methods are used: SSH connection, IP-echo and local IP reporting

Probe Address Discovery (What's this?)

	IPv4	IPv6
Connection Address 1	-	✔ 2601:646:c000:ea34:fad1:11ff:fea9:f2d2 1
IP Echo Service 2	✔ 73.252.192.40	✔ 2601:646:c000:ea34:fad1:11ff:fea9:f2d2 3
The Local IP 3	192.168.2.195	✔ 2601:646:c000:ea34:fad1:11ff:fea9:f2d2 2

Other Bits and Pieces

- Static IPs
  - Not all probes can use DHCP/RA, e.g. date centres
  - We provide a feature to define static IPs and DNS resolvers and supply these to the probes in-band
  - Probes need to be connected to a DHCP/RA aware network first to hear this
  - Probes fall back to DHCP if static configuration “doesn’t work”
  - This is causing problems, e.g. when DNS resolvers change over time
  - Still, some users insist this is a must have

Connected/Disconnected Probes

- Probes can work offline too
  - They keep on executing tasks they learned about earlier
  - However, they can't be assigned new tasks
  - Most tasks given to probes have a finite life span and are regularly refreshed, in order to prevent runaway probes
  - We keep a log of when (and from where) probes were connected from

Connection History (Showing only the last 25)

Internet Address	Controller	Connected (UTC)	Connected for	Disconnected (UTC)	Disconnected for
2601:646:c000:ea34:fad1:11ff:fea9:f2d2	ctr-ams05	2017-09-18 17:53:18	14h 10m	Still Connected	
2601:646:c000:ea34:fad1:11ff:fea9:f2d2	ctr-ams05	2017-09-16 18:16:57	1d 23h 28m	2017-09-18 17:44:57	0h 8m
2601:646:c000:ea34:fad1:11ff:fea9:f2d2	ctr-ams05	2017-09-16 07:27:09	10h 44m	2017-09-16 18:11:38	0h 5m
73.252.192.40	ctr-ams05	2017-09-16 00:36:59	6h 43m	2017-09-16 07:20:34	0h 6m
2601:646:c000:ea34:fad1:11ff:fea9:f2d2	ctr-ams05	2017-09-15 09:35:05	14h 55m	2017-09-16 00:30:51	0h 6m
2601:646:c000:ea34:fad1:11ff:fea9:f2d2	ctr-ams05	2017-09-14 10:44:25	22h 43m	2017-09-15 09:28:19	0h 6m

Endpoint (probe) Security

- The probes are “for free” for hosts
  - Therefore we want to prevent “reuse” as much as possible
  - We prefer hardware that’s not too easy to repurpose
  - Still, it’s possible — multiple blog posts are available
- We are not using a TPM as it would be prohibitively expensive
  - Even a TPM requires significant expertise to use right

Endpoint (probe) Security

- Each probe has an individual SSH key
  - We can disable each probe separately if needed
- Probes only do active measurements
  - They don't listen to traffic passively
- They don’t provide local services
  - No web server, other services, local configuration, nothing
  - There’s no need to worry about abuse and security of these services
- Local USB disk (in v3) is encrypted with individual probe keys
  - Prevents local firmware attacks

End Point (Probe) Security

- Firmware upgrades:
  - When a new firmware is available, probes upgrade in a lazy fashion, but we can always force them to upgrade faster
  - Each firmware upgrade is cryptographically verified
- v1-v3 probes can also upgrade their OS this way
- Anchors' OS is managed by the operations team, probe component is just a package

Design Principles - Security

- Any kind of compromise should have limited reach / consequences
- “Class A” problem - single device compromise
  - Mostly a fact of life, live with it, prevent it from causing harm
- “Class B” problem - take control of a set of devices temporarily (other than yours)
  - Contain as much as possible, recover / take control back eventually
- “Class C” problem - take over devices for good
  - Prevent to the best of knowledge

More on General Security Approach

- All firmware updates are (should be) signed
- In RIPE Atlas:
  - Firmware signature key used with n-of-m approach
  - Key and signing infrastructure is offline
  - Each probe has pre-installed public key(s) to verify firmware signature before upgrading
- In RIPE Atlas: Trust Anchors are hard coded
  - See initial connection to “reg.server” before
  - Just an entry point, hands keys to both parties for further communication
  - Can be updated with new firmwares

Security Incident Handling

- Bad Stuff will (likely) happen
- Or, some users say it happened to them
  - They may even be right!
- Or, at least some users will try funky stuff
- One should provide a means to allow reporting using responsible disclosure
  - Although some users' preference nowadays is to brag / nag / ask on Twitter instead

Use The Standards, Luke!

- Or at least follow Best Current Practices
  - https://datatracker.ietf.org/doc/draft-moore-iot-security-bcp/ is a good start
  - RIPE Atlas probes / infrastructure are pretty close to this
- It's amazing how often even the basics are done wrong in some IoT devices
  - Default passwords or open telnet service, or both, etc.
- Ultimately, incentives matter
  - Especially about how much energy one spends on security aspects vs. functionality

Some Lessons Learned

- It costs a lot, if possible at all, to implement specialised, “unbreakable” devices
  - See: Android, iOS and their vendors
- End-user support is a lot of work
  - Even if everything works just fine
  - Documentation, FAQs, self-help pages, notifications, etc. can help
- IoT devices pose some special challenges:
  - Could be “headless”, needs to be autonomous
  - Frequently “invisible”, needing no attention, esp. if it “works”
  - Install-and-forget is both a blessing and a challenge

Questions

robert@ripe.net
@kistel