



# AN `openssl` `socat` CONNECTION ADMINISTRATION TOOL

---

By Charalampos Kapolonaris (Technical Lead) &  
Vasilios Koutlas (Software Tester)

# THE PROBLEM WE ARE TRYING TO SOLVE

1

Secure general-purpose IP communication over any unsecured network.

2

Secure serial communication over any unsecured network for specialized industrial applications.

3

Interconnect serial legacy communications equipment over any modern IP network.

4

Low cost secure IP and serial communication between end-points.

# OUR SOLUTION

.Fluidity is a **scalable, low-cost, open-source** solution that utilizes the SSH protocol and openSSL SOCAT connections, to **securely** deliver data over any unsecured network.

- **Scalable:** For a single point-to-point connection you can just use two Raspberry computers. Use more machines as the number of the required end-points increase. Use more powerful equipment as bandwidth requirements increase.
- **Low-cost:** A minimum client-server setup can be implemented with just two Raspberries Pls. The OS can be any Debian-based distro.
- **Open-Source:** Source code written in BASH.
- **Open Architecture:** Use .Fluidity in any DEBIAN based supported system, unhinged from specific hardware restrictions.
- **Customizable:** Feel free to write new code to cover your specialized needs.
- **Securely:** Use the latest encryption technologies to secure your data.

# WHO WILL USE IT?

Internet Service Providers

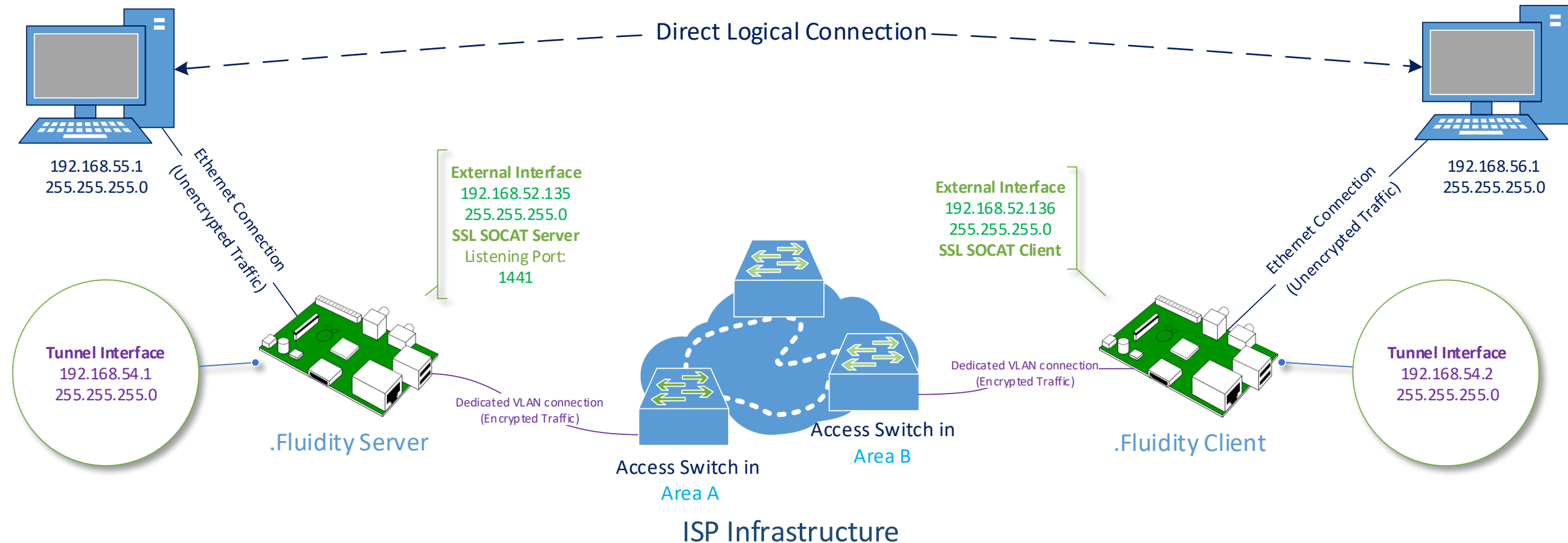
Manufacturing Plants

Government Agencies

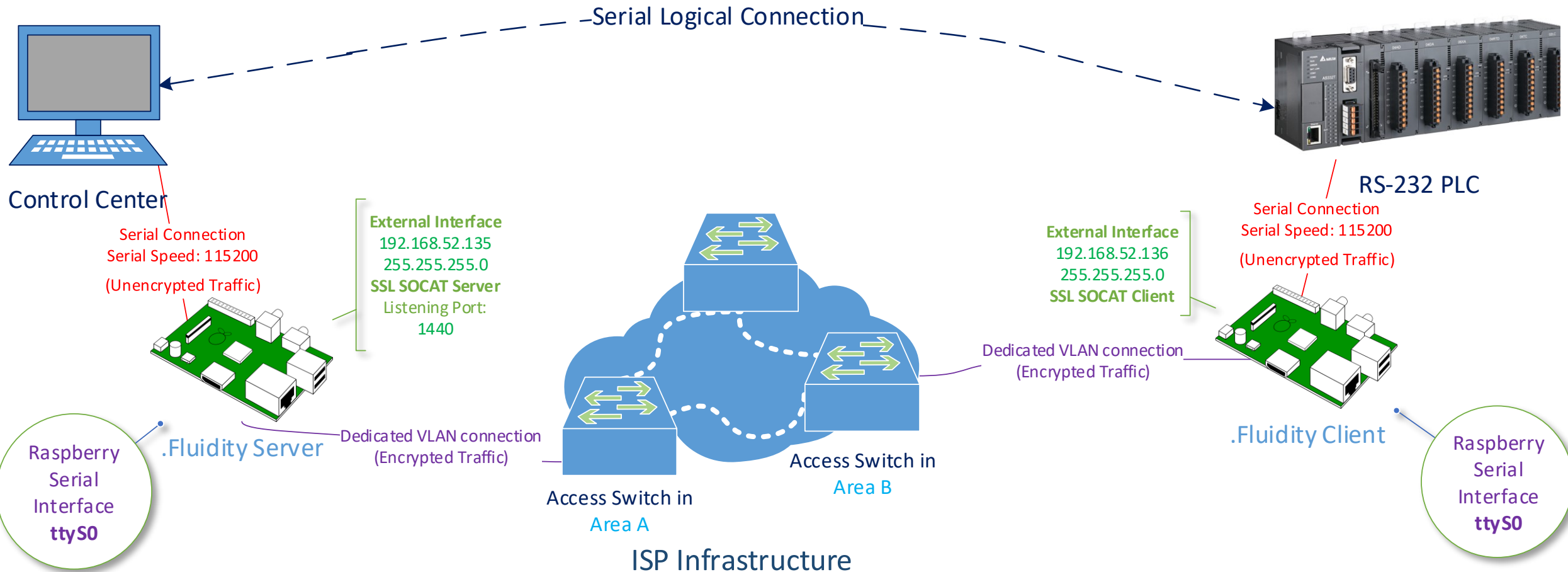
Private Citizens

Any entity that wishes to move data securely over a private or a public network.

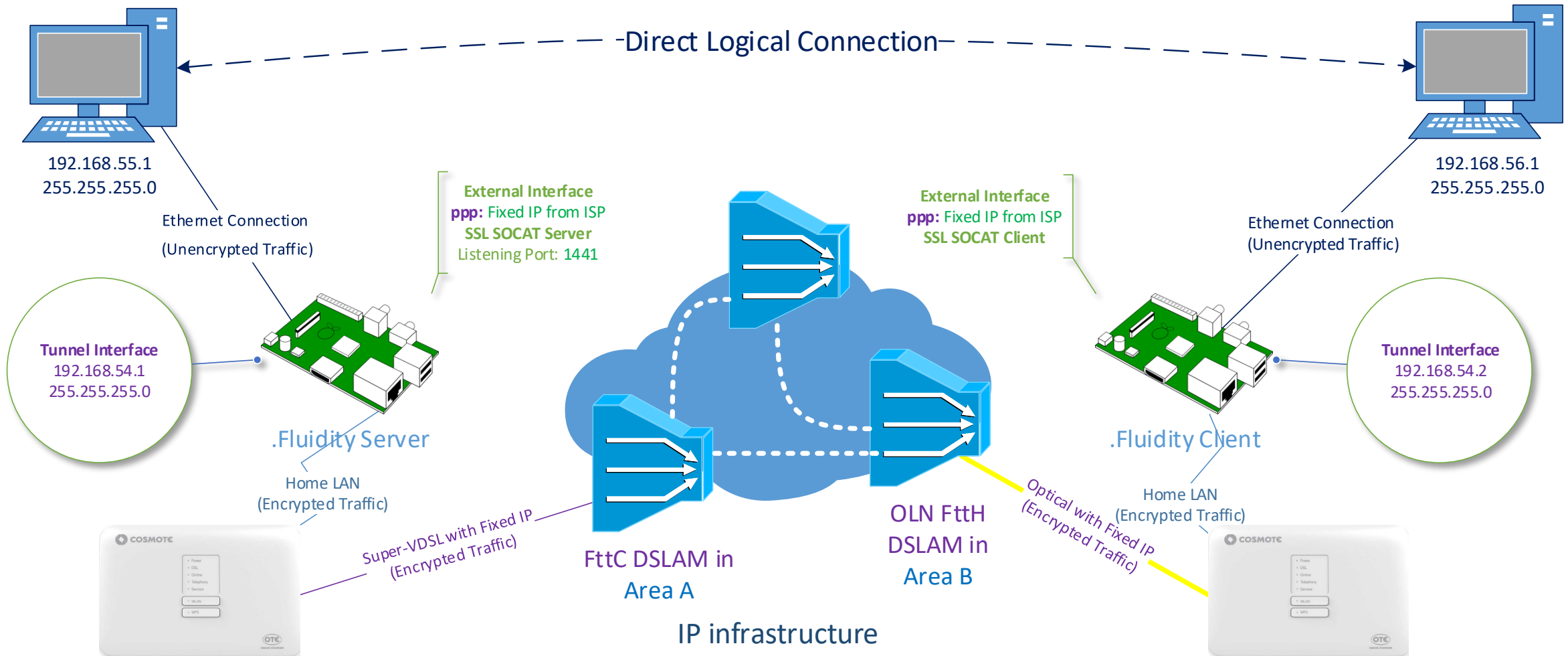
# BASIC EXAMPLE — .FLUIDITY IP TUNNELING COMMUNICATION OVER ISP'S INFRASTRUCTURE (PRIVATE NETWORK, PRIVATE IP)



# BASIC EXAMPLE — .FLUIDITY SERIAL COMMUNICATION OVER ISP'S INFRASTRUCTURE (PRIVATE NETWORK, PRIVATE IP)



# BASIC EXAMPLE — .FLUIDITY IP TUNNELING OVER PPPoE (THROUGH THE INTERNET, PUBLIC IP)



The diagram illustrates the IP infrastructure for the Fluidity project. It shows a central cloud representing the IP infrastructure, with various components connected to it:

- Control Center:** A computer icon at the top left, connected to the Raspberry Serial Interface via a "Serial Logical Connection" (dashed line).
- Raspberry Serial Interface ttyS0:** A Raspberry Pi board icon, connected to the Control Center via a "Serial Connection (Unencrypted Traffic)" (red line) and to the Fluidity Server via a "Serial Connection (Unencrypted Traffic)" (red line).
- Fluidity Server:** A server icon, connected to the Raspberry Serial Interface via a "Serial Connection (Unencrypted Traffic)" (red line) and to the FttC DSLAM via a "Home LAN (Encrypted Traffic)" (blue line).
- Fluidity Client:** A server icon, connected to the Raspberry Serial Interface via a "Serial Connection (Unencrypted Traffic)" (red line) and to the FttH OLT DSLAM via a "Home LAN (Encrypted Traffic)" (blue line).
- FttC DSLAM in Area A:** A DSLAM icon, connected to the Fluidity Server via a "Home LAN (Encrypted Traffic)" (blue line) and to the FttH OLT DSLAM via a "Super-VDSL with Fixed IP (Encrypted Traffic)" (purple line).
- FttH OLT DSLAM in Area B:** A DSLAM icon, connected to the Fluidity Client via a "Home LAN (Encrypted Traffic)" (blue line) and to the FttC DSLAM via a "Super-VDSL with Fixed IP (Encrypted Traffic)" (purple line).
- External Interface:** Both the Fluidity Server and Fluidity Client have an "External Interface" connected to the IP infrastructure cloud, labeled "ppp: Fixed IP from ISP" and "SSL SOCAT Server/Client" (green text).
- Serial Connection:** A red line connects the Raspberry Serial Interface to the Fluidity Server/Client, labeled "Serial Connection (Unencrypted Traffic)" and "Serial Speed: 115200" (red text).
- Home LAN:** Blue lines connect the Fluidity Server/Client to the FttC/FttH OLT DSLAM, labeled "Home LAN (Encrypted Traffic)".
- Super-VDSL:** Purple lines connect the FttC/FttH OLT DSLAM, labeled "Super-VDSL with Fixed IP (Encrypted Traffic)".
- Optical:** A yellow line connects the FttH OLT DSLAM to the Fluidity Client, labeled "Optical with Fixed IP (Encrypted Traffic)".



# BASIC EXAMPLE — ENCRYPTION ENCAPSULATION

## - SERVER-SIDE

Ping 192.168.56.1  
Ping 192.168.57.1

14	11.548806103	192.168.55.1	192.168.56.1	ICMP	74 Echo (ping) request
15	11.549691781	192.168.56.1	192.168.55.1	ICMP	74 Echo (ping) reply
16	12.112550762	192.168.55.1	192.168.55.255	NBNS	92 Name query NB.WPAD.66
17	12.112550762	192.168.55.1	192.168.55.255	NBNS	92 Name query NB.WPAD.66
18	12.112550762	192.168.55.1	192.168.55.255	NBNS	92 Name query NB.WPAD.66
19	12.112550762	192.168.55.1	192.168.55.255	NBNS	92 Name query NB.WPAD.66
20	12.112550762	192.168.55.1	192.168.55.255	NBNS	92 Name query NB.WPAD.66
21	12.112550762	192.168.55.1	192.168.55.255	NBNS	92 Name query NB.WPAD.66
22	12.112550762	192.168.55.1	192.168.55.255	NBNS	92 Name query NB.WPAD.66
23	12.112550762	192.168.55.1	192.168.55.255	NBNS	92 Name query NB.WPAD.66
24	12.112550762	192.168.55.1	192.168.55.255	NBNS	92 Name query NB.WPAD.66
25	12.112550762	192.168.55.1	192.168.55.255	NBNS	92 Name query NB.WPAD.66
26	12.112550762	192.168.55.1	192.168.55.255	NBNS	92 Name query NB.WPAD.66
27	12.112550762	192.168.55.1	192.168.55.255	NBNS	92 Name query NB.WPAD.66
28	12.112550762	192.168.55.1	192.168.55.255	NBNS	92 Name query NB.WPAD.66
29	12.112550762	192.168.55.1	192.168.55.255	NBNS	92 Name query NB.WPAD.66
30	12.112550762	192.168.55.1	192.168.55.255	NBNS	92 Name query NB.WPAD.66
31	12.112550762	192.168.55.1	192.168.55.255	NBNS	92 Name query NB.WPAD.66
32	12.112550762	192.168.55.1	192.168.55.255	NBNS	92 Name query NB.WPAD.66
33	12.112550762	192.168.55.1	192.168.55.255	NBNS	92 Name query NB.WPAD.66
34	12.112550762	192.168.55.1	192.168.55.255	NBNS	92 Name query NB.WPAD.66
35	12.112550762	192.168.55.1	192.168.55.255	NBNS	92 Name query NB.WPAD.66
36	12.112550762	192.168.55.1	192.168.55.255	NBNS	92 Name query NB.WPAD.66

23	15.570909569	Vmware_2b:6e:ea	Vmware_6a:21:42	ARP	60 192.168.55.1 is at 06
24	16.022180243	192.168.55.1	192.168.57.1	ICMP	74 Echo (ping) request
25	16.023090663	192.168.57.1	192.168.55.1	ICMP	74 Echo (ping) reply
26	16.416856789	192.168.55.1	192.168.55.255	NBNS	92 Name query NB.WPAD.66
27	17.018902197	192.168.55.1	192.168.57.1	ICMP	74 Echo (ping) request
28	17.019835062	192.168.57.1	192.168.55.1	ICMP	74 Echo (ping) reply
29	17.160242096	192.168.55.1	192.168.55.255	NBNS	92 Name query NB.WPAD.66
30	17.910098132	192.168.55.1	192.168.55.255	NBNS	92 Name query NB.WPAD.66
31	18.019409860	192.168.55.1	192.168.57.1	ICMP	74 Echo (ping) request
32	18.020403360	192.168.57.1	192.168.55.1	ICMP	74 Echo (ping) reply
33	19.019570166	192.168.55.1	192.168.57.1	ICMP	74 Echo (ping) request
34	19.020440161	192.168.57.1	192.168.55.1	ICMP	74 Echo (ping) reply
35	21.448746158	192.168.55.1	192.168.55.255	NBNS	92 Name query NB.WPAD.66
36	22.402510058	192.168.55.1	192.168.55.255	NBNS	92 Name query NB.WPAD.66

Internal Interface

352	58.450678663	192.168.52.135	192.168.52.136	TLSv1.2	159 Application Data
353	58.451046372	192.168.52.136	192.168.52.135	TCP	66 46154 → 1441 [ACK]
354	58.451741968	192.168.52.136	192.168.52.135	TLSv1.2	159 Application Data
355	58.451759712	192.168.52.135	192.168.52.136	TCP	66 1441 → 46154 [ACK]
356	59.449988979	192.168.52.135	192.168.52.136	TLSv1.2	159 Application Data
357	59.450653693	192.168.52.136	192.168.52.135	TLSv1.2	159 Application Data
358	59.450697622	192.168.52.135	192.168.52.136	TCP	66 1441 → 46154 [ACK]
359	60.449589311	192.168.52.135	192.168.52.136	TLSv1.2	159 Application Data
360	60.450109576	192.168.52.136	192.168.52.135	TLSv1.2	159 Application Data
361	60.450127700	192.168.52.135	192.168.52.136	TCP	66 1441 → 46154 [ACK]
362	61.450529566	192.168.52.135	192.168.52.136	TLSv1.2	159 Application Data
363	61.451068439	192.168.52.136	192.168.52.135	TLSv1.2	159 Application Data
364	61.451122630	192.168.52.135	192.168.52.136	TCP	66 1441 → 46154 [ACK]
365	61.451122630	192.168.52.135	192.168.52.136	TLSv1.2	159 Application Data
366	61.451122630	192.168.52.135	192.168.52.136	TCP	66 1441 → 46154 [ACK]
367	61.451122630	192.168.52.135	192.168.52.136	TLSv1.2	159 Application Data
368	66.945131105	192.168.52.135	192.168.52.137	TCP	66 1442 → 42842 [ACK]

352	58.450678663	192.168.52.135	192.168.52.136	TLSv1.2	159 Application Data
353	58.451046372	192.168.52.136	192.168.52.135	TCP	66 46154 → 1441 [ACK]
354	58.451741968	192.168.52.136	192.168.52.135	TLSv1.2	159 Application Data
355	58.451759712	192.168.52.135	192.168.52.136	TCP	66 1441 → 46154 [ACK]

FL Server PC\_1

.Fluidity Server

# BASIC EXAMPLE — ENCRYPTION ENCAPSULATION — CLIENT SIDE

322	46.808989585	192.168.52.135	192.168.52.136	TLSv1.2	159	Application Data
323	46.809092519	192.168.52.136	192.168.52.135	TCP	66	46154 → 1441 [ACK]
324	46.809866531	192.168.52.136	192.168.52.135	TLSv1.2	159	Application Data
325	46.810026692	192.168.52.135	192.168.52.136	TCP	66	1441 → 46154 [ACK]
326	47.807618682	192.168.52.135	192.168.52.136	TLSv1.2	159	Application Data
327	47.808046101	192.168.52.136	192.168.52.135	TCP	66	1441 → 46154 [ACK]
328	47.808274978	192.168.52.135	192.168.52.136	TLSv1.2	159	Application Data
329	48.806582446	192.168.52.135	192.168.52.136	TLSv1.2	159	Application Data
330	48.806932948	192.168.52.136	192.168.52.135	TCP	66	1441 → 46154 [ACK]
331	48.807111533	192.168.52.135	192.168.52.136	TLSv1.2	159	Application Data
332	49.806884503	192.168.52.135	192.168.52.136	TLSv1.2	159	Application Data
333	49.807205399	192.168.52.136	192.168.52.135	TCP	66	1441 → 46154 [ACK]
334	49.807438446	192.168.52.135	192.168.52.136	TLSv1.2	159	Application Data
353	54.550046224	192.168.52.135	192.168.52.136	TLSv1.2	159	Application Data

.Fluidity Client 1

Internal Interface

Ping Reply

4	0.000422455	192.168.56.1	192.168.55.1	ICMP	74	Echo (ping) reply
5	0.989596175	192.168.55.1	192.168.56.1	ICMP	74	Echo (ping) request
6	0.989823153	192.168.56.1	192.168.55.1	ICMP	74	Echo (ping) reply
7	1.990001514	192.168.55.1	192.168.56.1	ICMP	74	Echo (ping) request

1	0.000000000	192.168.55.1	192.168.56.1	ICMP	74	Echo (ping) request
2	0.000298784	Vmware_5e:70:a9	Broadcast	ARP	60	who has 192.168.56.2
3	0.000298784	Vmware_5e:70:a9	Vmware_5e:70:a9	ARP	42	192.168.56.254 is at
4	0.000298784	192.168.55.1	192.168.56.1	ICMP	74	Echo (ping) reply
5	0.000298784	192.168.56.1	192.168.55.1	ICMP	74	Echo (ping) request
6	0.000298784	192.168.55.1	192.168.56.1	ICMP	74	Echo (ping) reply
7	0.000298784	192.168.56.1	192.168.55.1	ICMP	74	Echo (ping) request
8	0.000298784	192.168.55.1	192.168.56.1	ICMP	74	Echo (ping) reply
9	0.000298784	192.168.56.1	192.168.55.1	ICMP	74	Echo (ping) request
10	2.989125115	192.168.56.1	192.168.55.1	ICMP	74	Echo (ping) reply
11	5.064772348	Vmware_e1:d2:f5	Vmware_5e:70:a9	ARP	42	who has 192.168.56.1
12	5.065006883	Vmware_5e:70:a9	Vmware_e1:d2:f5	ARP	60	192.168.56.1 is at 0

.FL 1.2 PC\_2

# .FLUIDITY SECURITY FEATURES IN SHORT...

Key folders are encrypted with ecryptFS. Its security specs are:

- Encryption method: AES
- Key length: 256bits
- With folder name encryption enabled.

.Fluidity client management and connection session management is achieved through an SSH administrative channel. SSH is utilized with the following security specs:

- Encryption method: RSA
- Key length: 2048 bits.

Data are exchanged between .Fluidity devices with SOCAT SSL connections. The Client – Server SSL public/private certificate pair uses the following features:

- Encryption method: RSA
- Key length: 2048 bits
- Password protected SSL certificates

# .FLUIDITY SECURITY FEATURES IN SHORT...

There is a safeguard mechanism that allows SSH and SSL certificate creation only when system entropy exceeds a 1000 bits.

There is also a safeguard mechanism that assigns to each .Fluidity (SOCAT SSL) connection a random port number.

Connection to a .Fluidity client is only allowed with an SSH private key. This feature prevents SSH brute-force attacks.

Key policies in SSH server `/etc/ssh/sshd_config` are changed, in order to fine-tune .Fluidity client security.

Firewalling is performed with the Uncomplicated Firewall (UFW), an easy to use front-end interface to Linux IP-tables.

# .FLUIDITY SECURITY FEATURES IN SHORT...

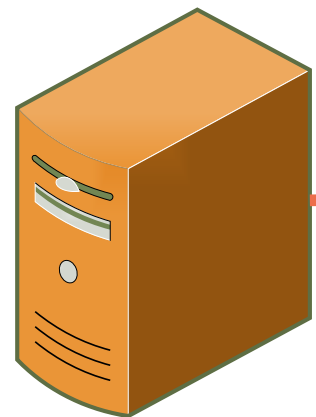
A safeguard mechanism exists that performs SSL client-server certificate validation by comparing their generated MD5 hashing values.

A safeguard mechanism exists that performs SSL client-server certificate validation by comparing their generated SHA256 hashing values.

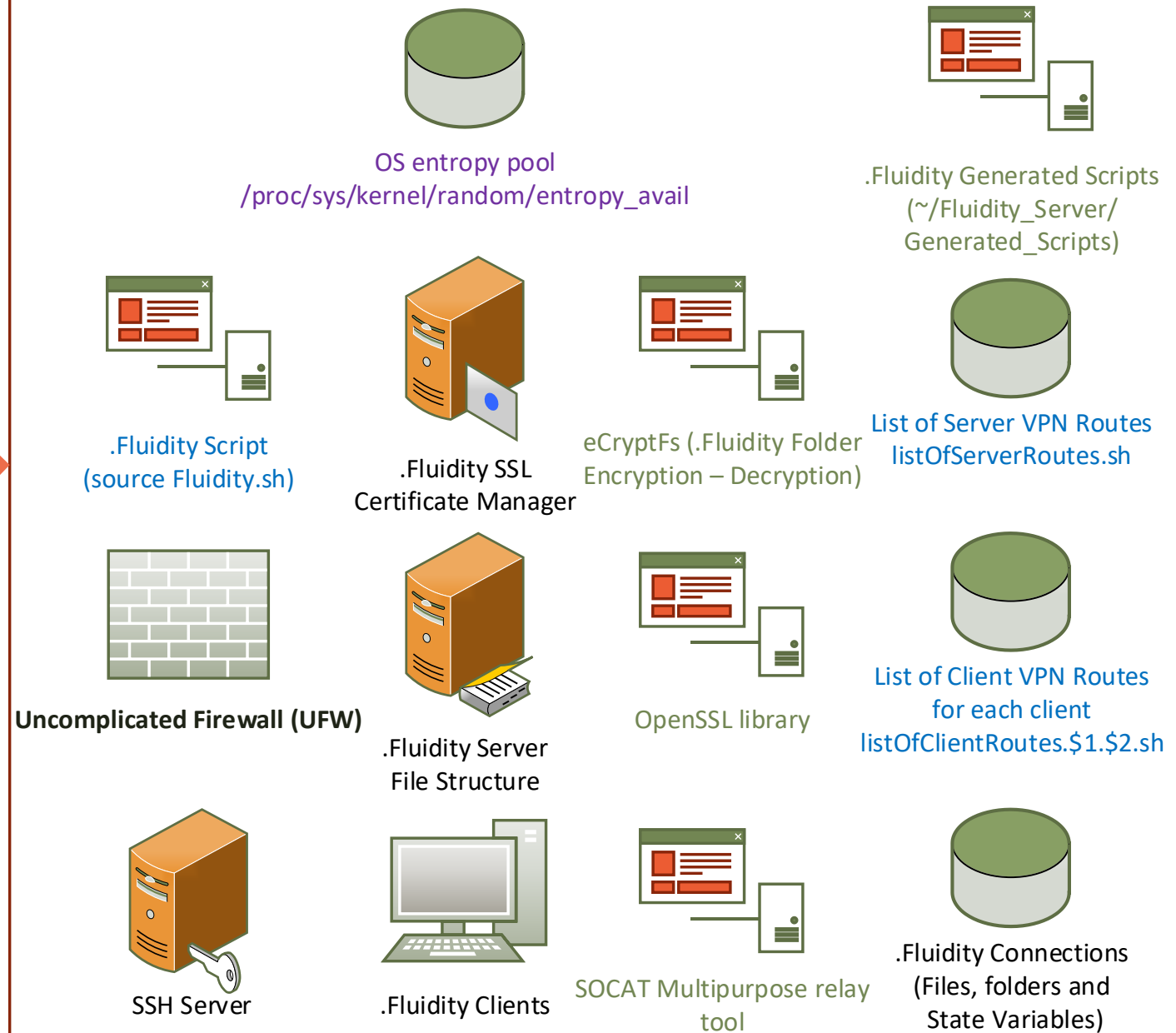
Client data, in the client connection folder, are automatically re-encrypted when the respective .Fluidity connection is detected inactive.

There is an obfuscation mechanism that ensures that the SSL certificate password is encrypted when htop command is executed on a .Fluidity client.

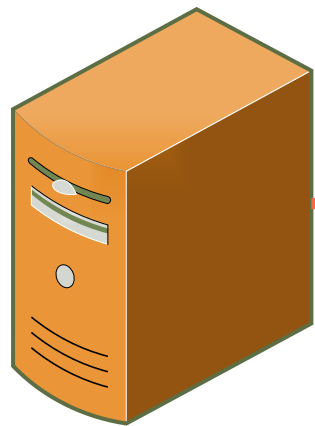
.Fluidity Server: Main systems and sub-systems (not in complete detail):



.Fluidity Server

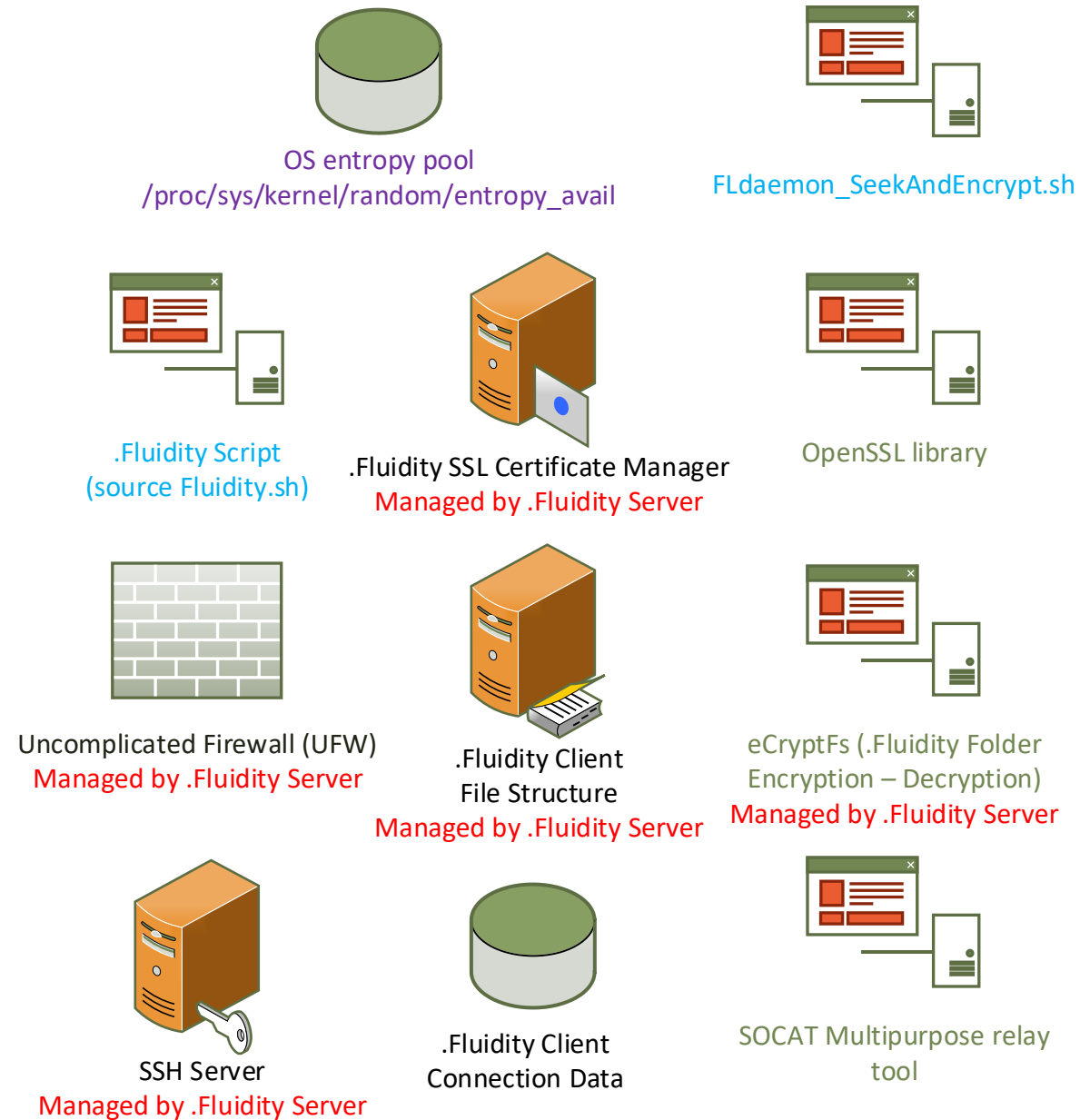


[ .Fluidity: Bring your telecoms infrastructure to the future ]



.Fluidity Client

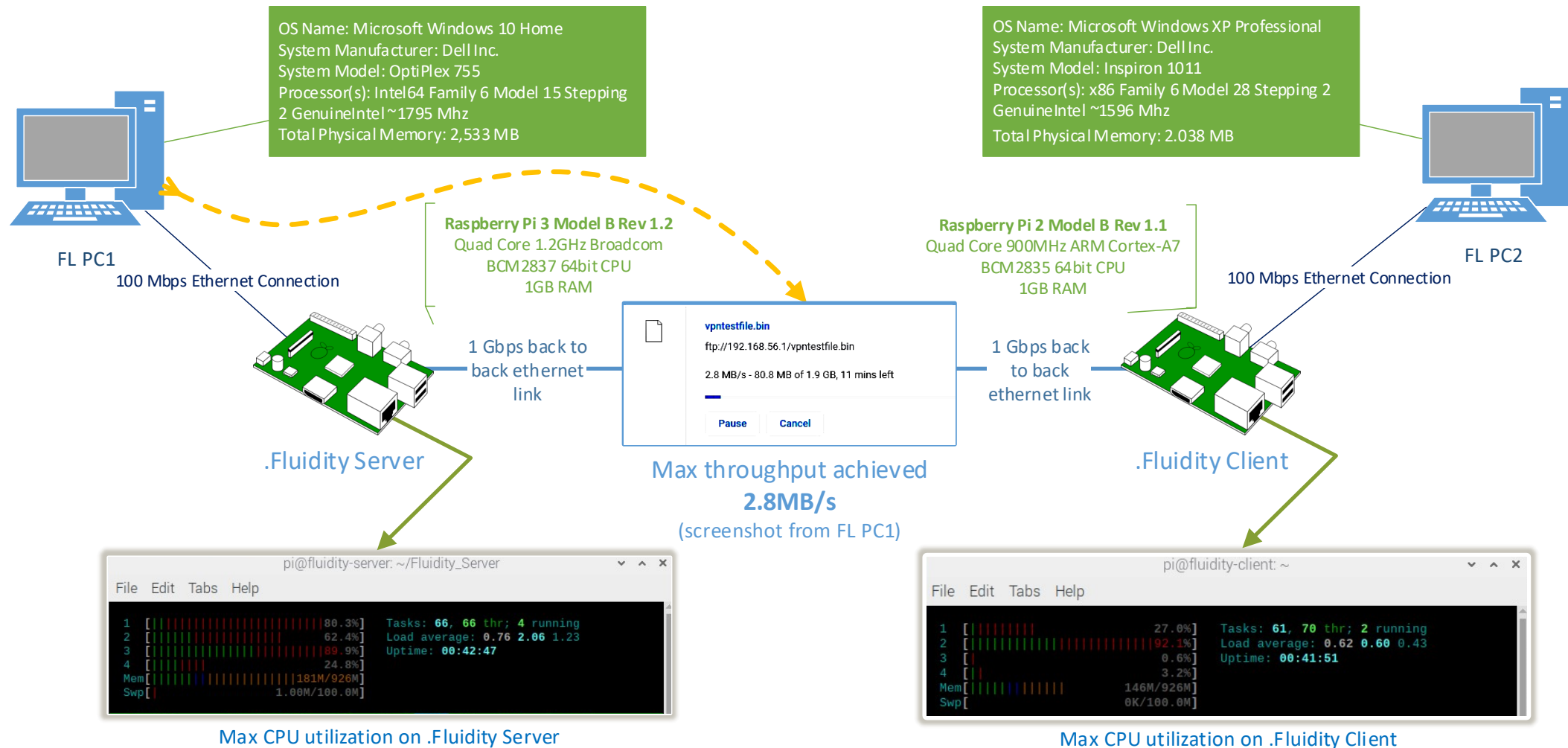
.Fluidity Client: Main systems and sub-systems (not in complete detail):



[ .Fluidity: Bring your telecoms infrastructure to the future ]

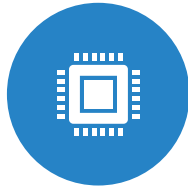


# LAB IMPLEMENTATION— .FLUIDITY MAX THROUGHPUT EXPERIMENT FOR A SINGLE CONNECTION





# WHY SHOULD PEOPLE USE IT?



Low cost alternative to much more expensive systems (Cisco 800s loaded with crypto licenses).



Open-Source. You know what you get (review the code yourself).



Fully customizable. Add your own code, if you wish.



Uses Debian LINUX Free operating system. Linux source code is open source and gets regularly reviewed by an active community.





Protect your competitive advantage by encrypting your data.



Use cheap public connections to implement a high-end VPN network.

# WHO WE ARE



 [ckaponaris@ote.gr](mailto:ckaponaris@ote.gr)  
 [LinkedIn](#)

Charalampos Kaponaris is a Telecoms Electronics Engineer, is currently employed by the OTE Group of Companies (HTO) and holds the position of the Customer Experience Technician.

Charalampos believes in the transformative power that Open Source Software brings to modern society.

Based on the challenges that modern telecommunications organizations are currently facing, Charalampos drew inspiration for this project.

These can be summarized as follows:

- The convergence of the legacy technologies to the new unified telecommunications IP infrastructure.
- The need for innovation in a continuously changing technological environment.

But most of all:

- safeguarding the telecommunications privacy and data security in systems that service critical subsystems of the corporate infrastructure.



 [vkoutlas@ote.gr](mailto:vkoutlas@ote.gr)

Vasileios Koutlas holds a degree in Electrical Engineering and adheres to the idea of Ethical and White Hat Hacking.

He has broad experience in the use of Linux OS and participated in many projects, mainly related to IP and VoIP networks.

Being highly sensitive in matters concerning data security and data confidentiality, he enthusiastically accepted his participation in this project.

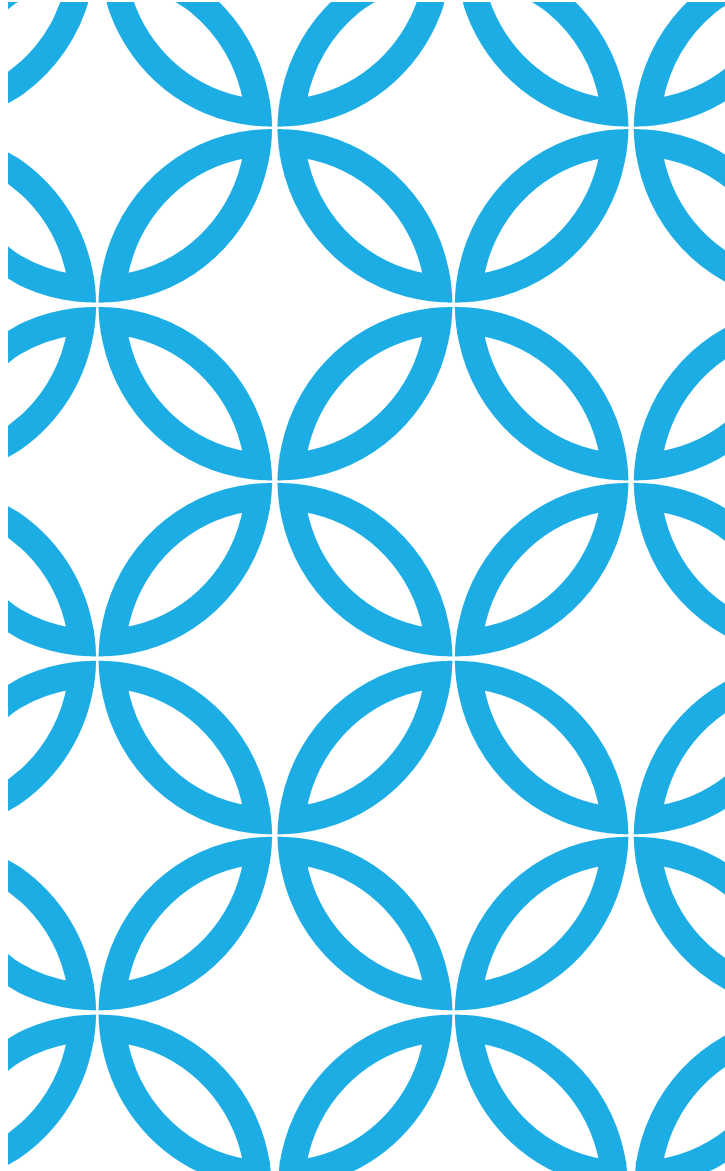
He believes that his involvement in the Open Source Community may result in the development of safer methods for data transfer.

He has been working in OTE Group of Companies (HTO) occupying various positions in the organization, currently occupying the position of CX Technicians Supervisor of Filed Technical Operations in Xanthi Greece.

FIND US ON  
GITHUB



<https://github.com/ckapolonaris/.Fluidity>



THANK YOU FOR YOUR  
ATTENTION

---