

Virtualizzazione e integrazione di sistemi

Motivazioni del corso Scenari di Integrazione

Vittorio Ghini

a.a. 2023/24

Prima lezione, parte 2

La figura del Systems Integrator (1)

- Il termine **sistemista** è **obsoleto**. Aveva senso quando occorreva gestire batterie di singoli computer.
- Ora il "sistemista" deve progettare e realizzare le infrastrutture software per gestire, a livelli architetturali diversi, una molteplicità di entità differenziate che interagiscono continuamente.
- Il termine più corretto per individuare **l'attività del sistemista moderno** è **"Systems Integration and Testing"**.
- La figura professionale viene definita dall' EUCIP nel seguente documento:
<http://www.eucip.it/profili/profili-professionali/SystemsIntegrationandTestingEngineerVersion2.4.pdf>.
- I numerosi task della figura professionale originano da questa responsabilità di massima:
"The Systems Integration and Testing Engineer works within organizations (either as an employee or as an external provider) to ensure that software systems and components are successful integrated across hardware systems and meet specified requirements."
- Tra le competenze richieste si distingue la conoscenza degli standard software de jure e de facto
"Requirements include a specific knowledge on how interfaces between software modules are built."

La figura del Systems Integrator (2)

- In alcuni casi si distingue la figura dell' installatore/configuratore/manutentore di sistemi rispetto alla figura del designer di sistemi complessi.
- La figura di **Systems Integration Architect** si occupa principalmente della progettare delle infrastrutture software per gestire, a livelli architetturali diversi, una molteplicità di entità differenziate che interagiscono continuamente.
- La realizzazione/installazione/manutenzione è lasciata al Systems Integrator.

La figura del Systems Integrator (3)

- Dato che il systems integrator deve
 - progettare e realizzare sistemi complessi,
 - costituiti da multiple componenti
 - che interagiscono via rete
 - mediante interfacce standard
- egli necessita di competenze che spaziano in molti ambiti dell'informatica, e in particolare di conoscenze che riguardano:
 - Protocolli di rete.
 - Sicurezza nei sistemi di rete.
 - Programmazione e scripting.
 - IDM (Identity Management Systems, Sistemi di Gestione dell'Identità).
 - Sistemi di Provisioning.
 - Architetture a MicroServizi e sviluppo,
 - Gestione centralizzata di servizi distribuiti.
 - Sistemi per Virtualizzazione.
 - Amministrazione di specifici sistemi operativi per desktop e server.
 - Amministrazione di sistemi cloud privati.

Ambito del corso - (di che ci occupiamo)

Il filo conduttore del corso è

la progettazione, il dispiegamento e la manutenzione

di sistemi software

tipicamente multi-utente, distribuiti e virtualizzati.

Non verranno approfonditi gli aspetti relativi alla progettazione della parte hardware.

Non verranno considerati aspetti di migrazione di processi, mentre saranno accennati aspetti di migrazione di macchine virtuali, anche "a caldo".

Verranno solo accennati aspetti giuridici relativi alla privacy e al GDPR.

Non si dettaglierà la configurazione interna dei DBMS.

Non si dettaglierà la configurazione dei Web servers.

Non si tratterà la coordinazione di sistemi distribuiti.

Motivazioni del corso

Scenari di Integrazione

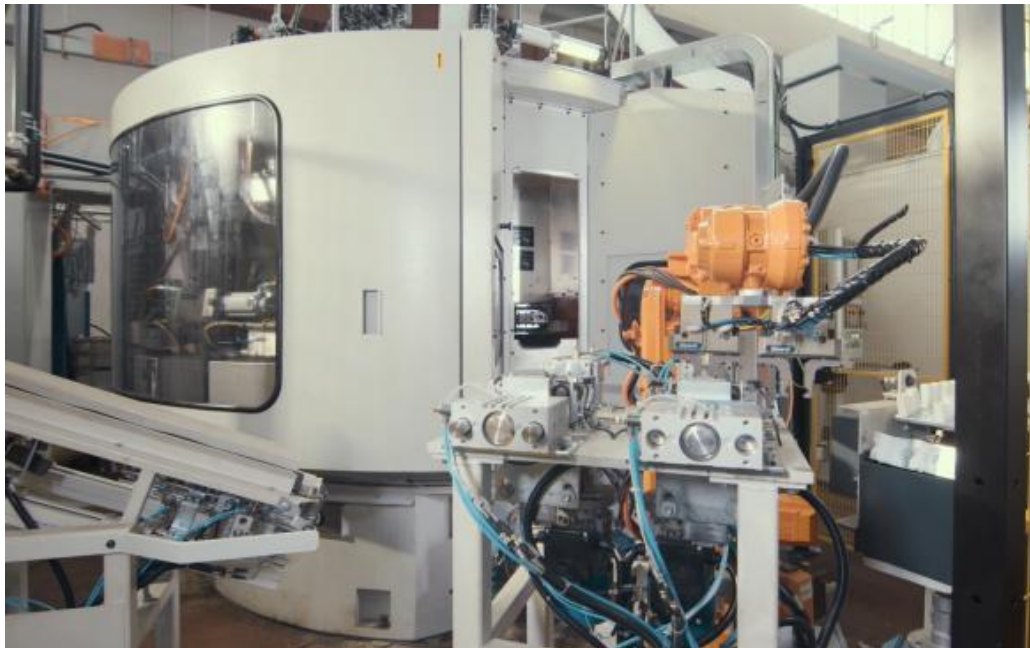
Progettare soluzioni aderendo agli Standard (de jure e de facto)

1. Integrazione col mondo dell'automazione
2. Applicazioni monolitiche vs. Sistemi a micro-servizi
3. Sistemi a micro-servizi in cloud
4. Sicurezza
5. Identity and Access Management (IAM) ---> Directory Service
6. Dispiegamento (deployment) scalabile mediante virtualizzazione e container
7. Scenari di Rete e ostacoli (firewall)

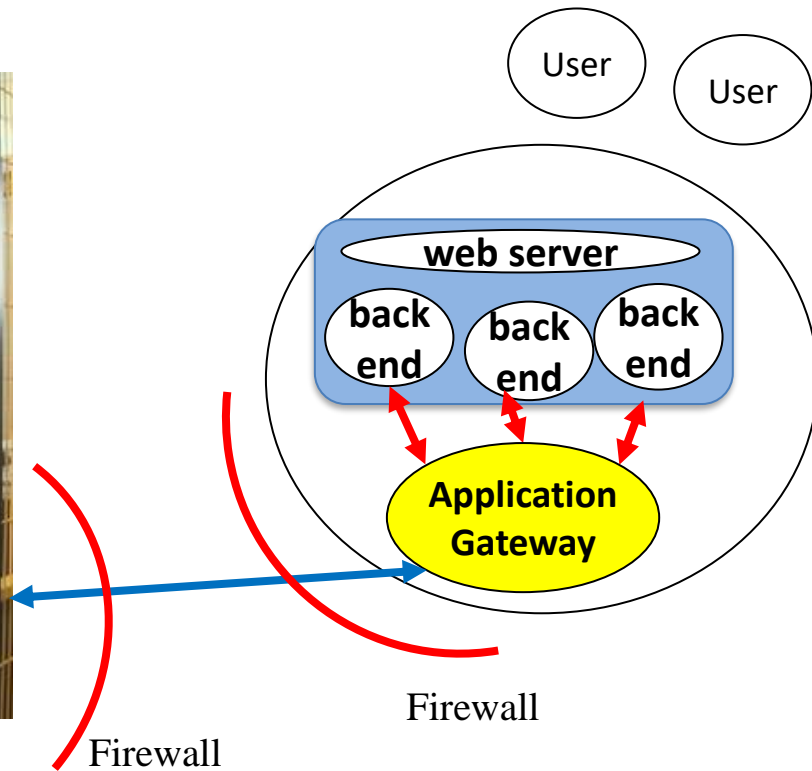
1) Integrazione col mondo
dell'automazione

L'Integrazione col mondo dell'automazione

- Le macchine automatiche, anche di grandi dimensioni, sono controllate da sistemi che non sono dei computer ma dei PLC.
- L'integrazione con il resto del mondo (industria 4.0) concerne spesso l'esportazione real-time di dati di sensori della macchina, su cui poi si opera con vari sistemi di analisi, big data, intelligenza artificiale. Anche flussi da videocamere sono spesso usati.
- L'integrazione richiede la conoscenza di protocolli di comunicazione specifici per l'ambiente dell'automazione ed i meccanismi di sicurezza.



Macchina Transfer , Giuliani Industries, Faenza (RA)



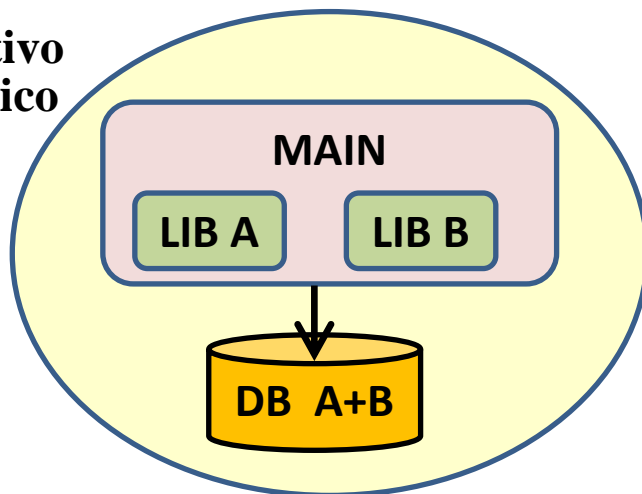
2) Sistemi a micro-servizi

Applicativi "monolitici" e "a micro-servizi"

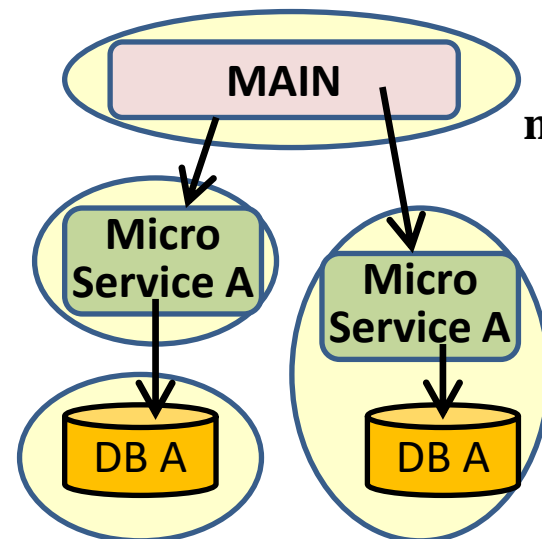
- Avete sempre visto una applicazione costituita di un solo pezzo o al massimo 2.
 - l'applicazione in java, oppure il servizio web,
 - oppure il servizio web + il tier del DB, indissolubili.
- In realtà gli applicativi moderni nel mondo reale sono costituite da
 - molteplici componenti separati e customizzabili, detti **micro-servizi**,
 - in esecuzione su uno stesso host oppure su più host distribuiti in rete,
 - componenti che **interagiscono tra loro scambiandosi messaggi**,
 - mediante interfacce software e protocolli standard,
 - permettendo di riutilizzare il software.

Il concetto resta valido se consideriamo un servizio web al posto dell'applicativo principale main

applicativo
monolitico



applicativo
a
micro-servizi



Esempio di "micro-servizi" in cloud Microsoft Azure

NB: non sono tutti prodotti Microsoft, alcuni sono open-source installabili in VM su cloud

- **VM** Macchina virtuale nuda o con sistema operativo a scelta
 - Configurabile per potenza di CPU, quantità memoria RAM e spazio disco
- **Disco virtuale** accessibile da tutte le VM
- **Archive Storage**, archiviazione basso costo per accessi rari
- Istanza di DB relazionale (es: **MySQL**)
- Istanza di DB documentale (es: **mongoDB**)
- Istanza di **DB con più tipi di API** di accesso (tipo documentale e relazionale) (es: Microsoft Azure **CosmosDB**)
- **Distributore di carico web e istanziatore a run-time di nuove VM**, per sopperire a picchi di carico, configurabile con opportune regole (**kubernetes**)
- **VPN gateway**
- **DNS**
- **Content Delivery Network**
- **Stream Analytics**
- **Machine Learning**
- **Active Directory**
- **Multi-factor Authentication**
- **Microsoft Media Service**
- **Microsoft Azure IoT Hub**, gateway centralizzato per gestione di messaggi da sensori vari, con protocolli AMQP, MQTT, HTTPS, ...
- **RabbitMq**, bus di messaggi AMQP MQTT, nasce per locale,, dispiegabile su VM
- **cloudamqp**, bus messaggi tipo RabbitMQ, dispiegato in cloud e fruibile anche da esterno

Sistemi "a micro-servizi" - Vantaggi

- Il fatto che l'applicativo sia strutturato in tanti componenti disaccoppiati (loosely coupled) e distribuiti su diversi host o datacenter, porta a considerare l'applicazione come un sistema.
- La scomposizione dell'applicativo in tanti micro-servizi comporta dei vantaggi:
 - **Riusabilità:** le componenti trasformate in micro servizi possono essere utilizzate da diversi applicativi .
 - **Scalabilità:** vengono replicati e bilanciati i soli micro servizi che necessitano
 - **Deployment:** può essere realizzato sia su una stessa macchina che su macchine differenti, riconfigurando in tal caso i servizi di comunicazione per utilizzare gli opportuni indirizzi, considerando la presenza di firewall, e verificando se la sicurezza
 - **Cloud :** I microservizi sono il modo ideale per affrontare lo sviluppo di applicativi orientati nativamente al cloud computing poichè i provider cloud offrono quasi esclusivamente micro-servizi.

Sistemi "a micro-servizi" - Problematiche

- D'altro canto, la forte necessità di comunicare richiede di :
 - costruire micro-servizi che espongano solo interfacce software e protocolli di comunicazione diffusi e condivisi (standard de facto).
 - progettare le applicazioni stesse affinché utilizzino solo questi standard,
 - assicurare che la rete di comunicazione permetta il passaggio di questi messaggi o, in alternativa, in fase di progettazione e dispiegamento inserire opportuni infrastrutture e servizi per superare firewall e NAT.

Progettare le comunicazioni

- Parte fondamentale della progettazione di un sistema "a micro-servizi" è costituito dalla progettazione delle comunicazioni tra i componenti, individuando i protocolli di comunicazione standard più adatti allo scenario di riferimento, tra quelli per i quali esistono implementazioni mature e largamente diffuse.

Sistemi "a micro-servizi"

Esempio comunicazioni locali (1)

Anche un'applicazione su singolo host può dover comunicare tra tanti micro-servizi. Vediamo un esempio:

- Una **applicazione a base web**, i cui **utenti** remoti **visualizzano, tramite un browser**, sia le statistiche attuali sui pezzi **prodotti da una macchina automatica**.
- Il servizio web è realizzato con nginx, NodeJs, Puma e opera, racchiuso in container, su un server Linux che sta nello stesso stabilimento della macchina.
- Gli **utenti sono serviti mediante connessioni persistenti HTML5** con un'istanza del back end web per ciascun utente.
- Un software della **macchina automatica** (non è un computer, è un PLC) **invia le info su ogni pezzo prodotto al server mediante il protocollo standard, OPC-UA**.
- Il server contiene due DB, un relazionale, MySql, ed un documentale, mongoDb.
- Il server **ha un servizio separato (OPC client)** che **riceve questi dati, li inserisce nel DB documentale**, poi aggiorna alcune statistiche e le salva nel DB relazionale.
- **Dopo ogni l'inserimento, l'OPC client informa tutte le istanze dei back-end web**, che in quel momento stanno servendo degli utenti mediante delle connessioni persistenti HTML5, **di aggiornare i grafici** che stanno visualizzando poiché sono appena giunti nuovi grafici.
- **Un'altro software monitora la raggiungibilità in rete** di un magazzino e, in caso di indisponibilità, **deve avvisare gli utenti** connessi al servizio web, con una notifica.



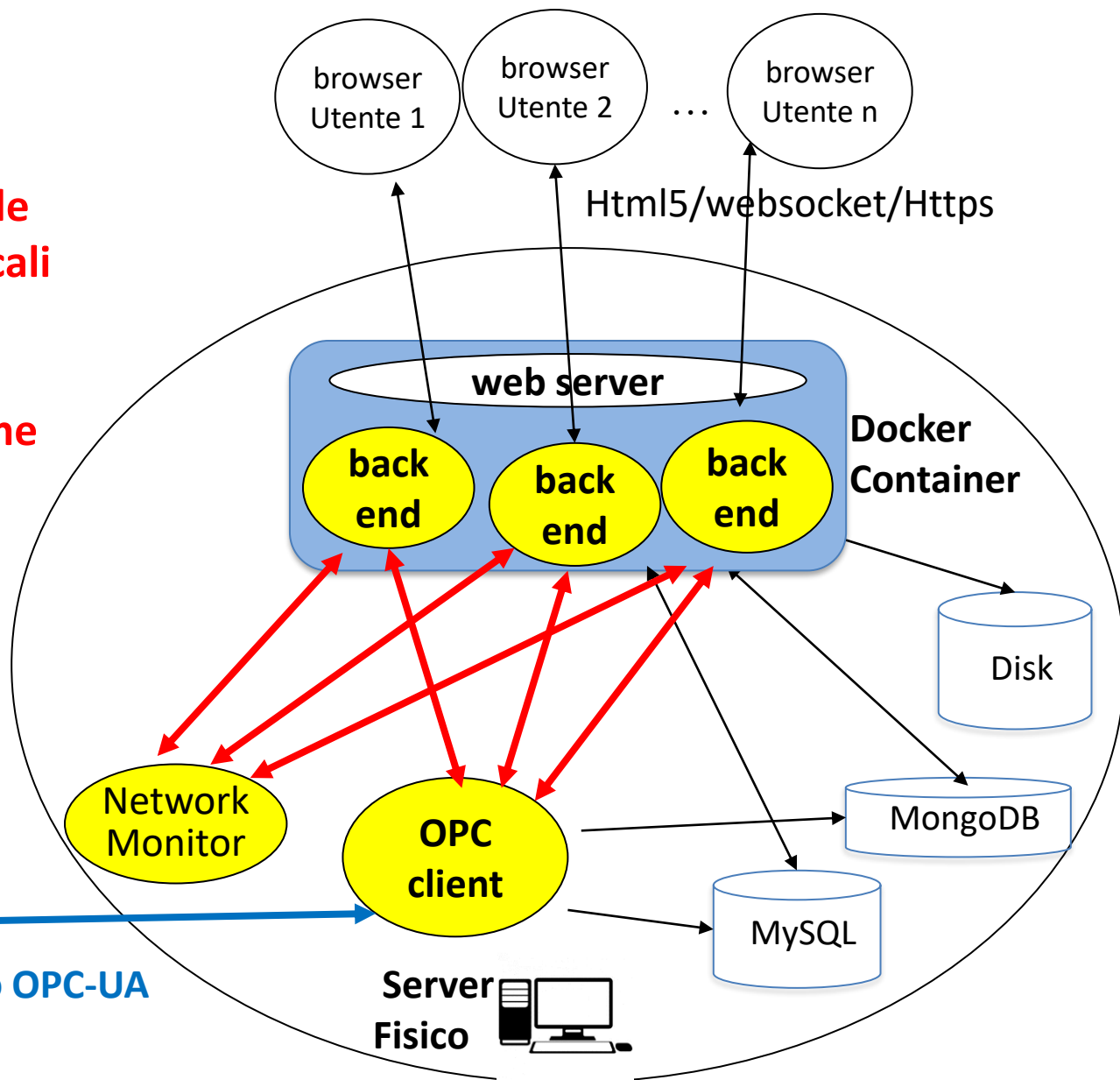
Sistemi "a micro-servizi"

Esempio comunicazioni locali (2)

- Problema

come realizzare le
comunicazioni locali
(rosse)

"molti a molti"
quando a run-time
non sappiamo
chi sono
e quanti sono
questi molti?



Sistemi "a micro-servizi"

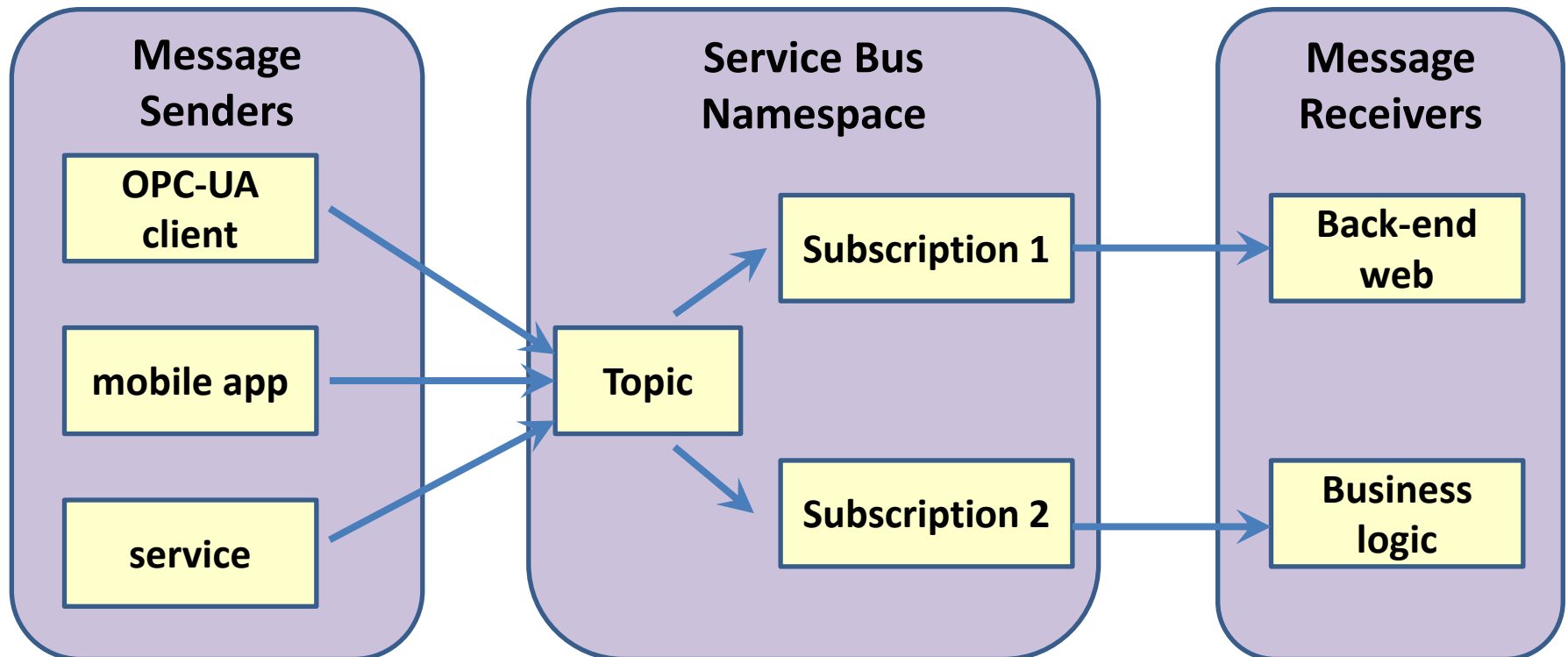
Esempio comunicazioni locali (3)

Bus di messaggi di tipo publish/subscribe orientato ai topic

Ogni entità receiver si registra e dichiara a quali topic è interessata.

Chi vuole spedire pubblica un messaggio indicando un subject del messaggio.

Il messaggio è consegnato a quelli che hanno indicato quel subject tra i topic di loro interesse.



Sistemi "a micro-servizi"

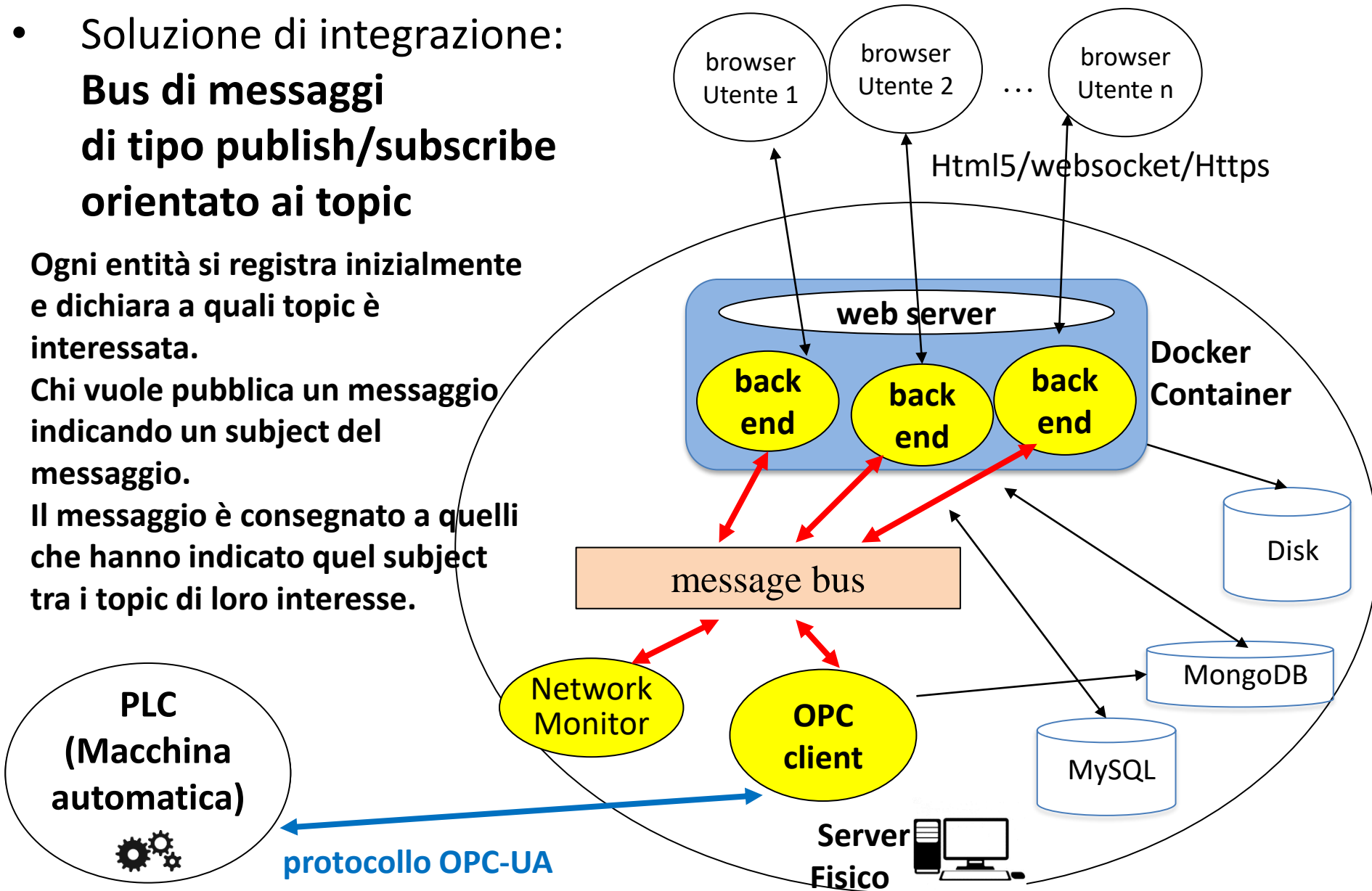
Esempio comunicazioni locali (4)

- Soluzione di integrazione:
**Bus di messaggi
di tipo publish/subscribe
orientato ai topic**

Ogni entità si registra inizialmente e dichiara a quali topic è interessata.

Chi vuole pubblica un messaggio indicando un subject del messaggio.

Il messaggio è consegnato a quelli che hanno indicato quel subject tra i topic di loro interesse.



Sistemi "a micro-servizi"

Bus di Messaggi (o Message Broker)

- Vengono chiamati in modi diversi, tra cui Message Bus e Message Broker.
- Esistono diversi protocolli usati per lo scambio dei messaggi.
- Tra i principali ricordiamo (ma li vedremo più avanti)
 - AMQP
 - MQTT
- Questi protocolli solitamente :
 - definiscono il formato base dei messaggi trasportati, che include un body.
 - permettono di scegliere le modalità di consegna dei messaggi (affidabile, non affidabile, con ricevuta di consegna, con timeout, ...).
 - mantengono i messaggi in code in accordo con le politiche di consegna.
 - Mettono a disposizione API per diversi linguaggi, anche a base web, e per diversi sistemi operativi
- Le applicazioni solitamente costruiscono un loro strato software che definisce il formato interno dei messaggi.
 - Spesso, per semplicità si sceglie di trasportare un body avente formato Json.

Sistemi "a micro-servizi"

Implementazioni Esistenti di Bus di Messaggi

- Esistono diverse implementazioni di questi bus di messaggi. Tra i più diffusi:
- **RabbitMQ,**
 - open source, esegue su vari sistemi operativi
 - implementa sia AMQP che MQTT,
 - deve essere installato sulle macchine in cui opera.
 - può essere utilizzato anche per far comunicare entità su macchine fisiche diverse
 - quindi può operare anche su macchine virtuali in cloud
 - implementa livelli sicurezza.
 - Esiste un servizio pre-installato in cloud a cui si può accedere mediante API (**cloudamqp**)
- **Microsoft Azure Service Bus Messaging**
 - opera nei datacenter di Microsoft Azure
 - implementa sia AMQP che MQTT,
 - mette a disposizione API per entità che lavorano su host fuori dal cloud
 - implementa livelli di sicurezza

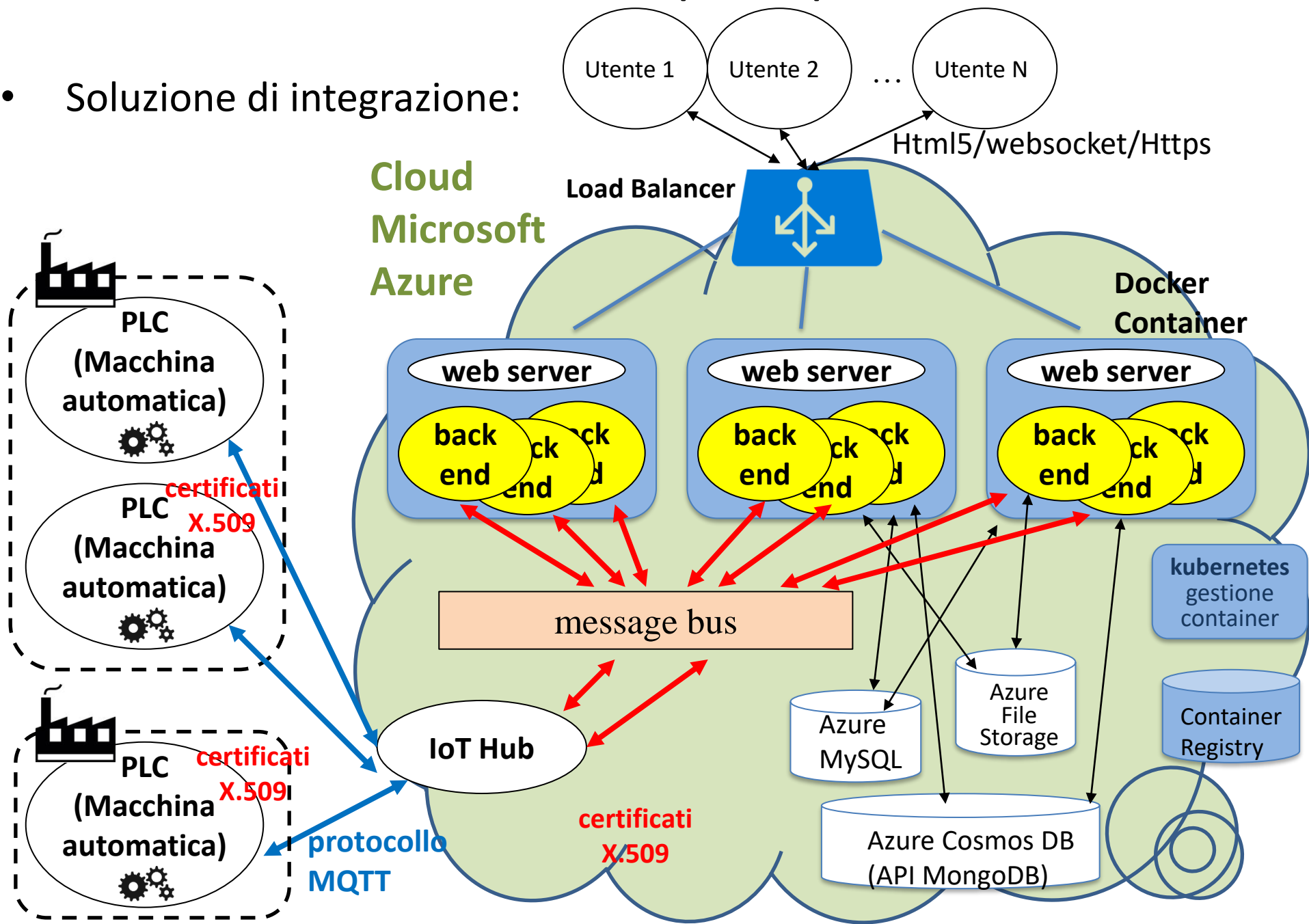
3) Sistemi a micro-servizi
dispiegati anche in cloud

Sistemi "a micro-servizi" dispiegati in cloud

- Un sistema può appoggiarsi, parzialmente o completamente, su micro-servizi operanti su infrastrutture cloud.
- In realtà, **non è usuale che un sistema operi esclusivamente in cloud**, poiché molto spesso i **dati raw su cui operare vengono generati da sorgenti di varia origine** (utenti, wearable objects, sensori, macchinari, auto, etc etc) distribuite geograficamente e collezionati poi in cloud.
- Quindi, solitamente **almeno la parte di acquisizione dei dati coinvolge l'esecuzione di procedure in dispositivi esterni al cloud**.
- I **sistemi cloud** più evoluti **facilitano questa fase di acquisizione dati, mettendo a disposizione API e librerie da eseguire su questi dispositivi esterni**.
- Le successive fasi di processing vengono invece tipicamente svolte su cloud.
- Consideriamo lo stesso esempio del **macchinario che produce pezzi in uno stabilimento** e che invia statistiche ad un **server** che stavolta non è nello stabilimento stesso ma che si trova in un datacenter in cloud.
- Ipoteizziamo anche che **ci siano molte macchine automatiche in stabilimenti diversi** e che tutte mandino le loro statistiche **allo stesso server in cloud**.
- Gli stabilimenti e le **macchine** potrebbero appartenere a **proprietari diversi**, mentre il servizio in cloud potrebbe essere gestito dal produttore delle macchine, sicché occorre **distinguere i permessi di accesso degli utenti alle info collocate in cloud** delle diverse macchine (anagrafica di utenti e macchine).

Sistema "a micro-servizi" principalmente in cloud

- Soluzione di integrazione:



4) Sicurezza nei sistemi informatici

Sicurezza dei sistemi informatici

- **La sicurezza è un criterio base di progettazione** trasversale a tutti i sistemi.
 - Non è un aspetto accessorio a cui pensare in un secondo momento.
- Molto spesso, un requisito di progettazione è la **centralizzazione in un servizio unico, delle funzionalità di autenticazione e autorizzazione all'uso di risorse.**
 - La **Centralizzazione**/coordinamento dei sistemi di autenticazione ed autorizzazione è essa stessa un **fattore di sicurezza poiché limita i punti di attacco e favorisce il controllo.**
 - I **Servizi di Directory** si occupano anche di autenticazione e autorizzazione

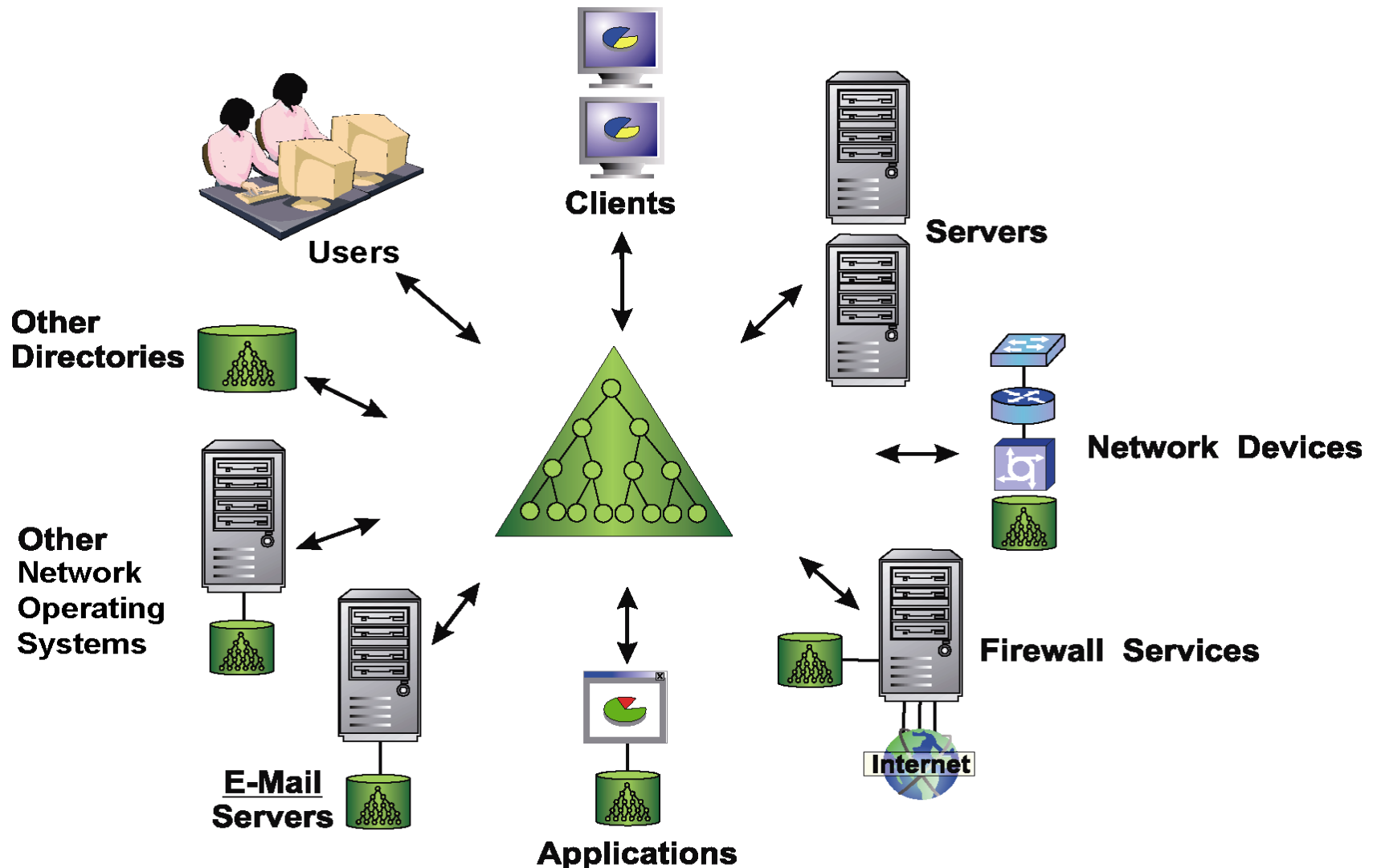
Esistono infiniti scenari in cui la sicurezza determina la progettazione dei sistemi e richiede la centralizzazione delle procedure di autenticazione ed autorizzazione.

- **Sicurezza nei mezzi trasmissivi**
 - canali wireless es: server Radius
 - comunicazioni applicative es: https, ssl/tls
 - reti private virtuali es: VPN, esporre porte di protocollo con ssl
- **Autenticazione delle entità**
 - certificati X.509
- **Autenticazione degli utenti**
 - Autenticazione multi-fattore
- **Autorizzazione all'accesso alle risorse**
 - Infrastrutture, Servizi di Directory, Domini, DNS, interazione tra sistemi

5) Directory Service

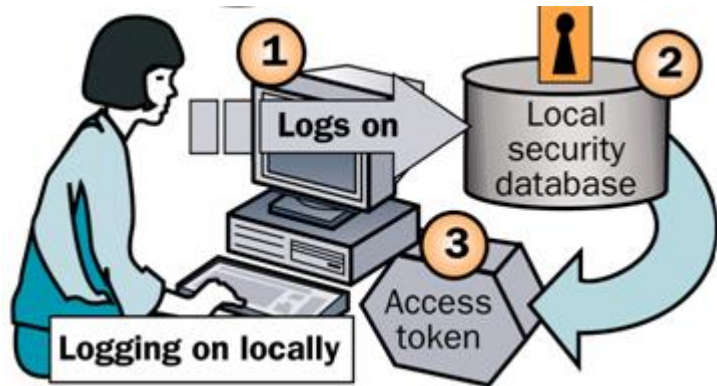
Directory Service

Un sistema centralizzato di gestione/fruizione di informazioni per utenti, reti, servizi, applicazioni in un dominio.



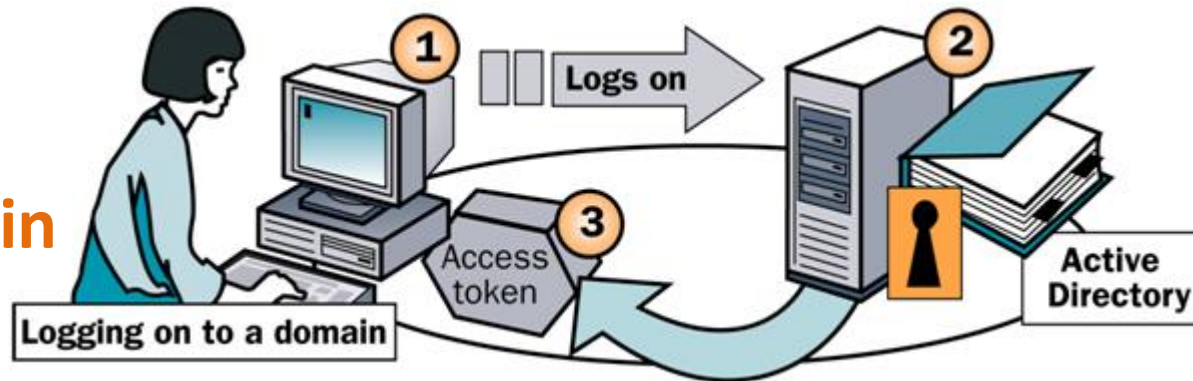
Scenario di Integrazione con auth/autn centralizzati

Processo di Autenticazione



local

domain



☐ Access token

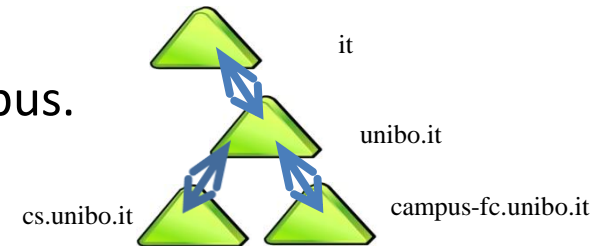
- Provides user identity and security setting
- Enables a user to gain access to resources and perform system tasks

- User provides user name and password.
- The o.s. compares the logon information with the user information that is stored in the appropriate database.
- If the information matches and the user account is enabled, then an access token is created for the user.
- If the logon information does not match or the user account is not validated, access to the domain or local computer is denied.

Scenario di Integrazione con auth/autn centralizzati

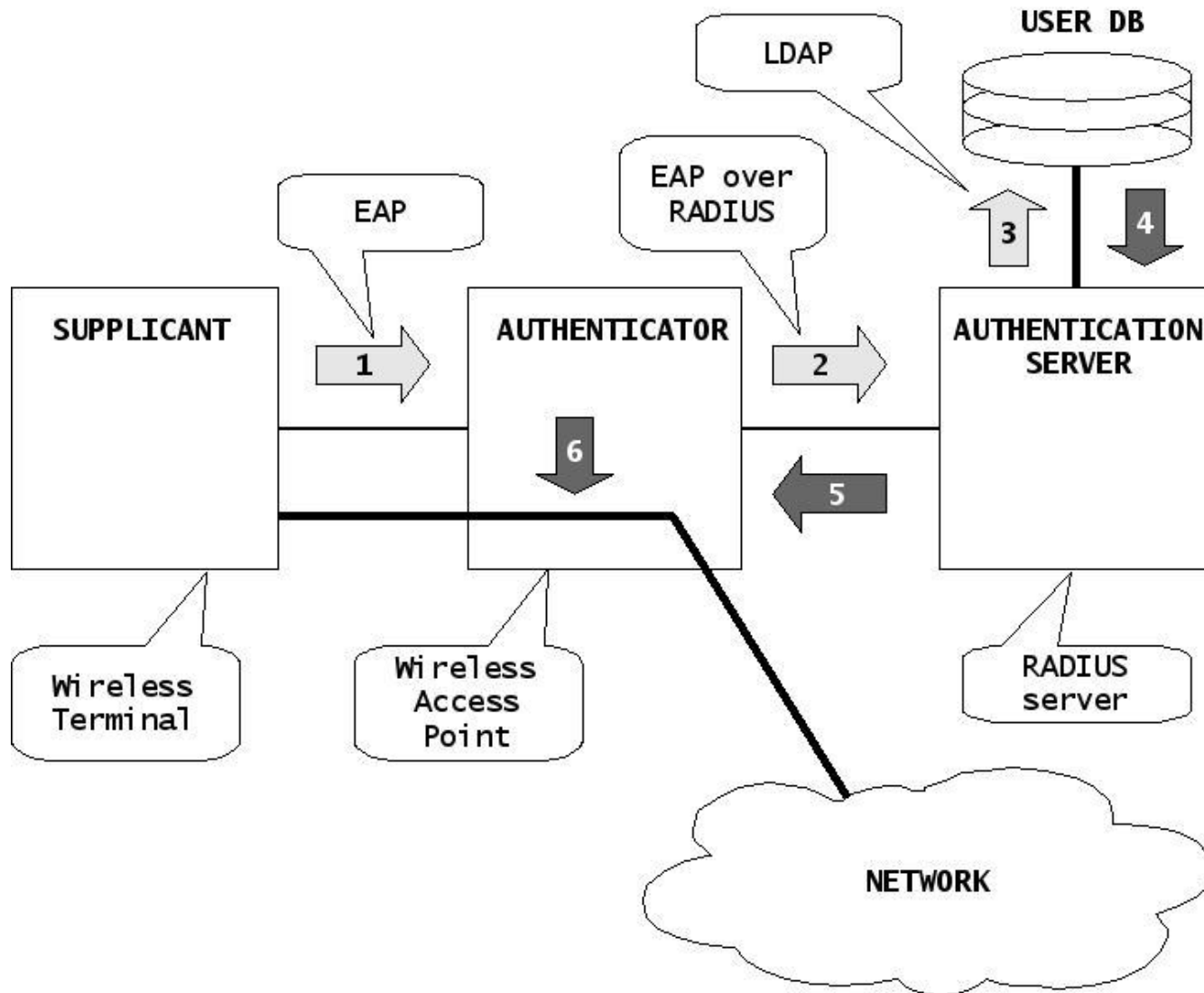
Identity and Access Management = Directory Service.

- **Laboratori studenti e computer del personale amministrativo di Unibo**
 - Unico sistema centralizzato di autenticazione autorizzazione
 - Utilizza il sistema **Microsoft Active Directory**
 - Dispiegato su sistemi windows server 2008
 - Distribuisce responsabilità e funzionalità sui 5 campus.
 - Distingue i domini interni unibo.it e studio.unibo.it
- PC studenti appartengono al dominio studio di unibo.it (join)
- Studenti si autenticano presso server delocalizzati nei campus.
- Possono usare servizi condivisi (file system remoti) forniti da sistemi Windows o Linux mediante protocolli standard (SMB,CIFS, Samba).
- Politiche di sicurezza centralizzate.
- PC windows fanno join immediato
- PC Linux utilizzano applicativi/protocolli di accesso alle risorse quali LDAP e Samba.
- **Possibile anche il contrario: Server a base Linux (applicativi della suite Samba, informalmente chiamato Linux Active Directory) e client Windows.**



5b) Autenticazione in canali wireless
basata su Directory Service

Autenticazione per accesso a AP WiFi e retrostante rete basata su **Server Radius** (WPA2, 802.1x e Active Directory)



Autenticazione per accesso a AP WiFi e retrostante rete basata su **Server Radius** (WPA2, 802.1x e Active Directory)

Lo standard **WPA2** per la sicurezza nelle reti wireless prevede di utilizzare 802.1x per gestire l'autenticazione. Ottima sicurezza e grande flessibilità, poiché permette di gestire svariati metodi di autenticazione.

IEEE 802.1x è uno standard per l'autenticazione e l'autorizzazione in rete, sviluppato in origine per reti wired, basato sul protocollo EAP (Extensible Authentication Protocol) per l'autenticazione. IEEE 802.1x prevede tre entità :

- **Authenticator**: chiede l'autenticazione prima di offrire il servizio (è l'access point).
- **Supplicant**: vuole accedere al servizio e dev'essere autenticato (è il wireless terminal).
- **Authentication server**: verifica le credenziali del supplicant a nome dell'autenticator (nel nostro caso il server RADIUS).

L'autenticazione avviene tramite il protocollo EAP, che permette di negoziare diversi metodi di autenticazione, i principali sono EAP-MD5, EAP-TLS, EAP-TTLS e PEAP.

Durante il processo di autenticazione l'AP (authenticator) ha un ruolo passivo, si occupa solo di prendere i messaggi provenienti dal WT (supplicant), contenuti nel protocollo EAP, e metterli nel protocollo RADIUS per inoltrarli al server (authentication server) e viceversa

Autenticazione per accesso a AP WiFi e retrostante rete

esistono alternative

- . PPPoE, 802.1x e Active Directory

5c) Autorizzazione all'uso di applicazioni (single-sign-on)

Single-Sign-On

autorizzazione all'uso di più applicazioni con una sola richiesta di credenziali utente

DEFINIZIONE: **Single Sign On** è un generico meccanismo che concede ad un

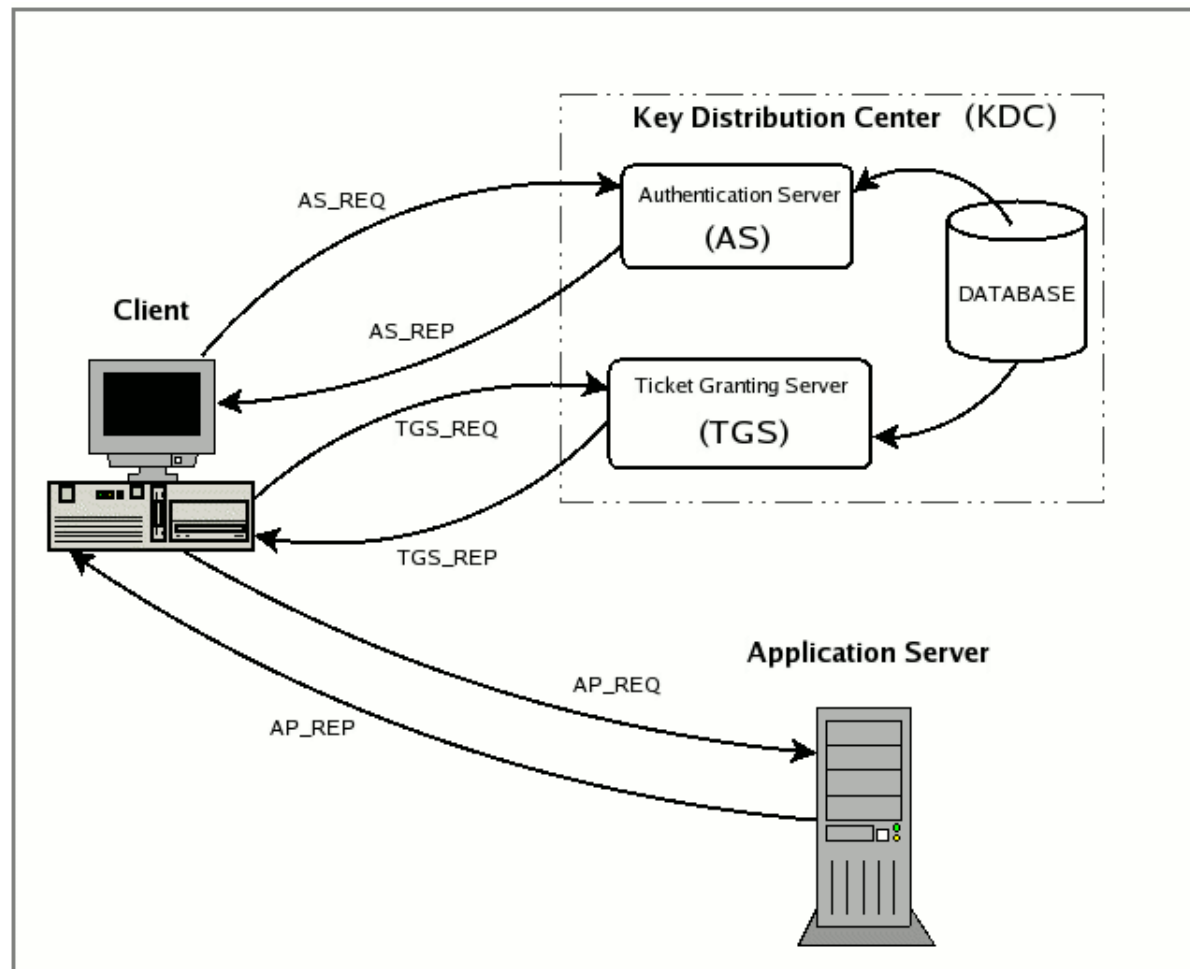
- utente l'autorizzazione ad utilizzare più applicazione mediante una sola richiesta di credenziali.

L'implementazione del SSO può essere fatta in tanti modi diversi a seconda di:

1. Dove si trova il computer su cui opera l'utente (**nello stesso dominio** in cui operano i server delle applicazioni **oppure fuori da quel dominio**).
2. Quale sistema operativo opera sul computer dell'utente (ambiente Microsoft o no).
3. Le **applicazioni** da utilizzare sono **a base web o no** (nel caso web infatti esistono vari meccanismi standard di sessione (cookie e altro) per mantenere in modo standard alcune informazioni di autenticazione quali token di accesso ed altro.
 - Tra i sistemi di SSO per servizi web il più usato è **Shibboleth**

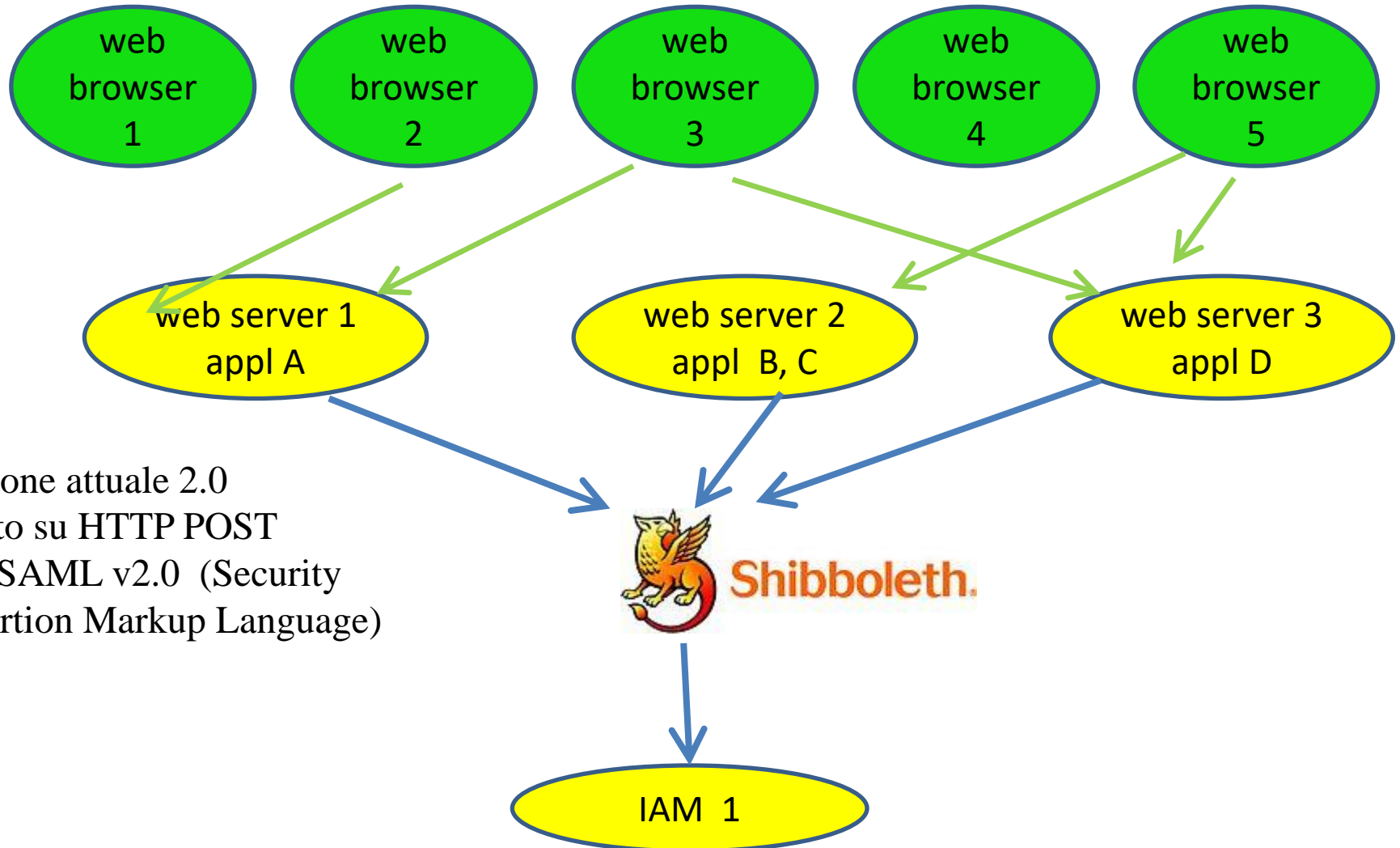
Single-Sign-On all'interno di in un dominio

Autorizzazione all'uso di più applicazioni con una sola richiesta di credenziali utente
(basato su kerberos e marche temporali)



Single-Sign-On per applicazioni web, anche all'esterno di un dominio

Shibboleth



versione attuale 2.0
basato su HTTP POST
e su SAML v2.0 (Security
Assertion Markup Language)

6) Dispiegamento (deployment)
di servizi
scalabili ed isolati
mediante

Virtualizzazione e Containerizzazione

Virtualizzazione

La **virtualizzazione** è la creazione di una rappresentazione virtuale (basata su software) e non fisica di qualcosa. Si può virtualizzare applicazioni, server, storage, reti.

Virtualizzazione del server: tecnologia mediante cui si esegue un sistema operativo ospitato isolandolo **all'interno di una macchina che non è fisica bensì ottenuta da uno strato software** denominato Virtual Machine Monitor o **Hypervisor**.

La virtualizzazione del server può essere realizzata in tanti modi diversi, distinti per:

- **il livello HW o SF su cui gli hypervisor si appoggiano** e quindi l'utilizzo o meno delle funzionalità offerte dall'eventuale presenza di un s.o. sottostante.
- **il fatto che gli hypervisor mettano a disposizione le stesse funzionalità dell'hardware oppure delle funzionalità diverse nel qual caso occorre modificare il s.o. ospitato.**
- **le modalità di dettaglio mediante cui si eseguono le istruzioni del s.o. ospitato.**

La virtualizzazione ha il **pregio** di

- **isolare una applicazione all'interno del s.o. installato sulla VM.**

La virtualizzazione classica ha **alcuni difetti**:

- **richiede l'esecuzione di tutto un s.o. nella sua VM** per isolare ed eseguire una specifica applicazione.
- **l'applicazione da eseguire comunque dipende** dalla specifica versione, configurazione, presenza di software installato **nel s.o. installato nella VM.**

Containerizzazione


La tecnologia dei container sposta il focus dalla virtualizzazione del server verso la **virtualizzazione della applicazione**, creando per l'applicazione un contesto di esecuzione virtualizzato che non è più tutto un server (hw+s.o.).

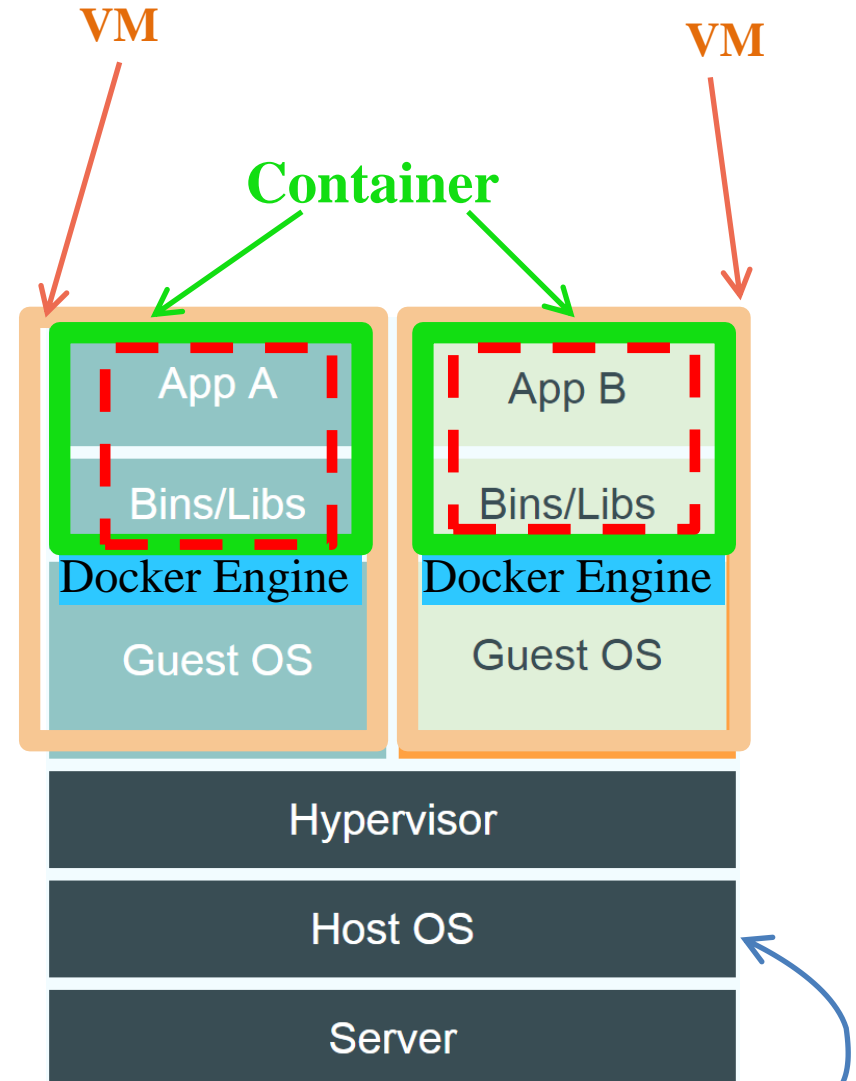
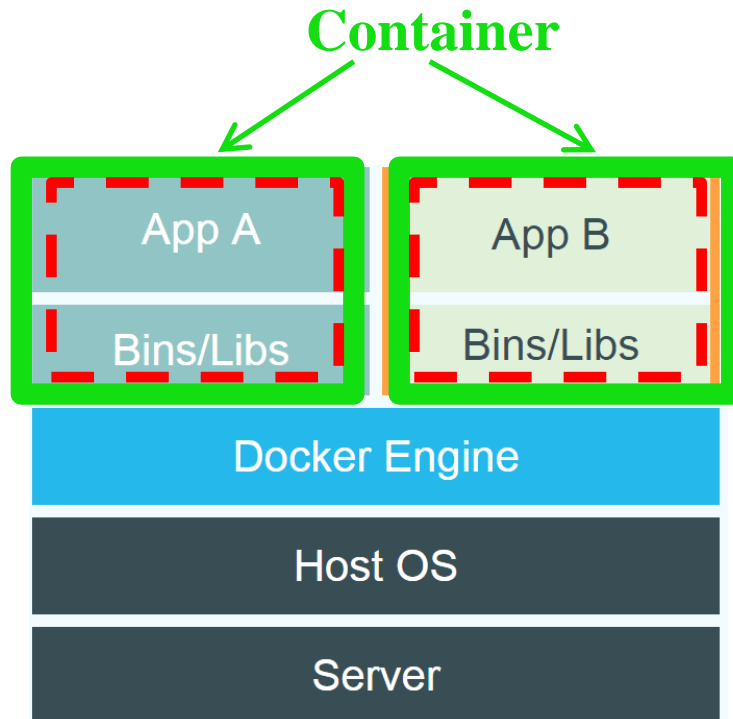
I Container: realizzano una specie di sottoinsieme delle risorse offerte da un s.o. e mettono a disposizione queste risorse alle applicazioni che vengono eseguite all'interno dei container.

Quindi i container isolano ulteriormente le applicazioni all'interno del s.o. ospitato, astraendo, in parte, dal s.o. e dalla macchina virtuale in cui gira il s.o., e rendendo l'applicazione dipendente dal container ma non dalla versione e configurazione del s.o. ospitato nella VM.

Si specifica "astraendo, **in parte**, dal s.o." perché il container sfrutta pesantemente il s.o. sottostante e riutilizza la maggior parte delle funzionalità del s.o. e della eventuale VM su cui gira il s.o. Questo riutilizzo ha lo scopo di minimizzare il peso del container rendendone più agile l'esecuzione.

Container vs. Virtual Machine

L'applicazione 
può essere inserita
in un **container** all'interno del s.o.
oppure sopra un s.o. all'interno di una **VM**



NB: alcuni hypervisor si sostituiscono al sistema operativo, quindi qui l' Host OS potrebbe non esserci

Cloud e categorie di servizi cloud

Il **Cloud** di un provider è essenzialmente un gruppo di datacenter in ciascuno dei quali vengono forniti dei micro-servizi.

- I micro-servizi più essenziali consistono di macchine virtuali per le quali è possibile richiedere certe tipologie di CPU e caratteristiche HW, ammontare di memoria, spazio disco etc etc,
- Su questi micro-servizi base vengono poi costruiti altri micro-servizi più complessi.
- Alcuni micro-servizi più raffinati permettono di gestire ridondanza dei dati e failure recovery verso altri datacenter.

I servizi cloud si distinguono per il **livello di componibilità che offrono** e per la **capacità di scalare, trasparentemente all'utente**. Distinguiamo servizi:

SaaS - Software as a service: un servizio applicativo che non vede né sistema operativo né host su cui lavora. Ad esempio, google documents, editi documenti senza sapere nulla.

PaaS - Platform as a service: fornisce delle API di sviluppo per comporre servizi più complessi, ma non sai su che sistema operativo lavori e nemmeno su che host. E' scalabile in modo trasparente. Ad esempio, il database CosmosDB.

IaaS - Infrastructure as a service: fornisce una infrastruttura su cui installare servizi: Ad esempio una macchina virtuale, una connessione di rete. La scalabilità deve essere effettuata o aumentando la potenza o aumentando il numero delle istanze.

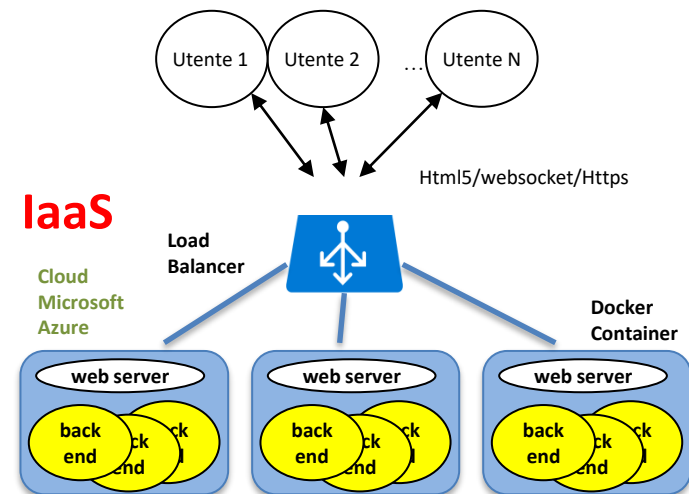
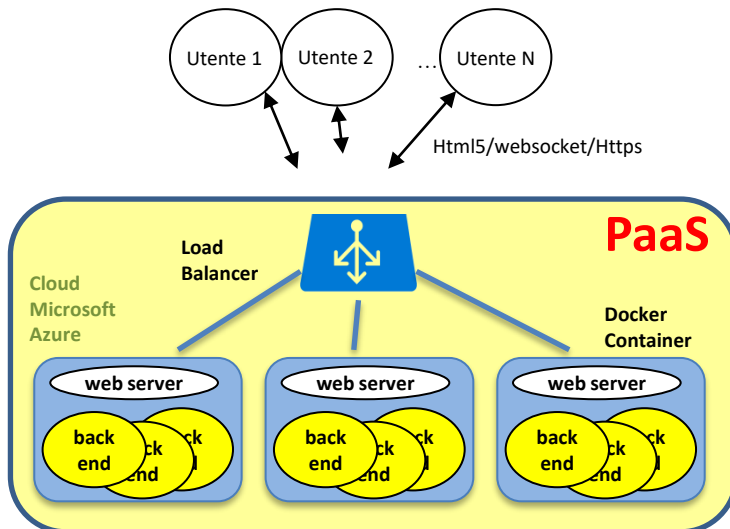
PaaS vs. IaaS

Evidenziamo la differenza tra PaaS e IaaS con un esempio:

Supponiamo che occorra costruire un servizio web, abbiamo due scelte:

PaaS: Affidarsi ad un servizio PaaS fornito da un provider, che ci garantisce di servire fino a 1000 utenti contemporaneamente, e che ci fa pagare un costo base più un tanto per ciascuna richiesta utente che ecceda le 2000 richieste al giorno. Noi dobbiamo scrivere le funzioni del server usando API di nginx ma non dobbiamo occuparci della scalabilità.

IaaS: Costruire il servizio web partendo dalle componenti di tipo IaaS, cioè istanziare le VM, collocare sulle VM i container contenenti nginx, istanziare il load balancer kubernetes che all'occorrenza faccia partire nuove VM e container con nginx .

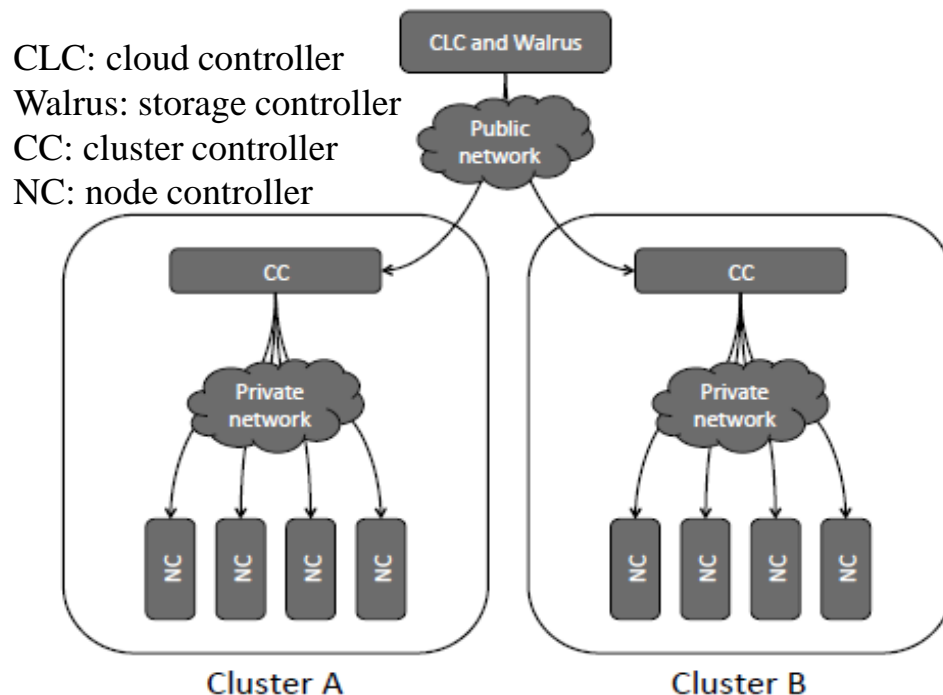


Piattaforme di gestione di cloud privati

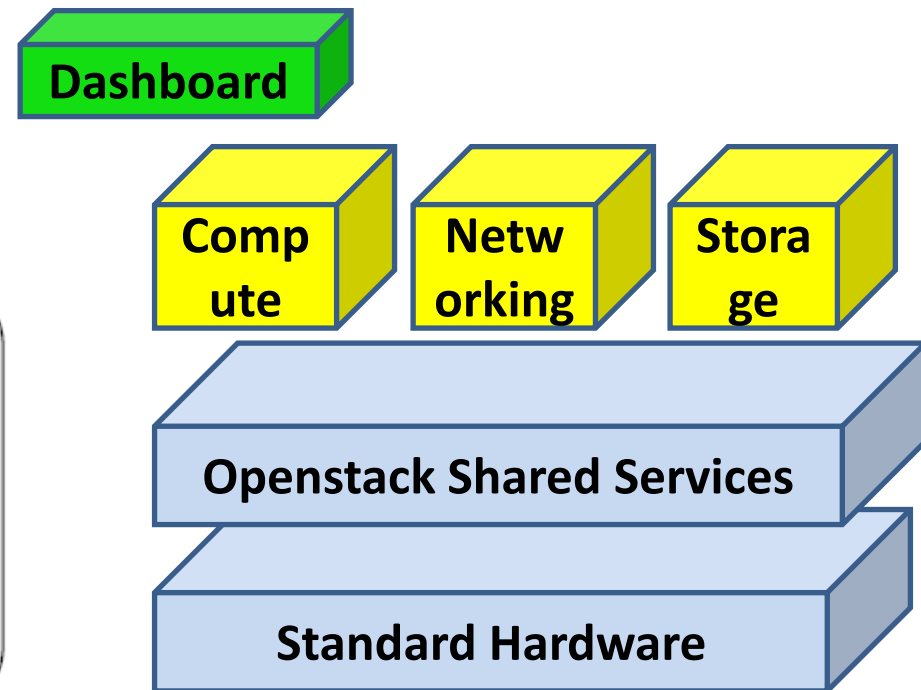
Un privato vuole costruire un proprio datacenter in cui realizzare un sistema cloud.

- Ciò è reso possibile da alcune architetture software che permettono di utilizzare le risorse di più macchine fisiche rendendole disponibili parzialmente su richiesta.
- Queste architetture software vengono dette Piattaforme di Gestione del Cloud (Cloud Management Platform).
- Tra queste architetture ricordiamo le principali: **OpenStack e OpenNebula**.

Visione Interna del Datacenter



Visione Esterna del Datacenter



Cloud

Hardware requirements

- A very large number of processors
 - Clustered in racks as blades
- In one major computer centre
 - May be replicated for business continuity
- With massive online storage
 - RAID for resilience
- And excellent communications links
 - For access

Economies of scale – both
purchasing and operation

Energy economies in location

Staffing economies in
location

Multitenancy: How does it work

- Cloud resources (hardware) shared dynamically between customers;
- Each customer application in its own virtual machine
 - Isolation for security, privacy
 - Allows scheduling with respect to shared resources
- Application in one VM multithreaded with user data / profiles etc in other VMs

Cloud trend

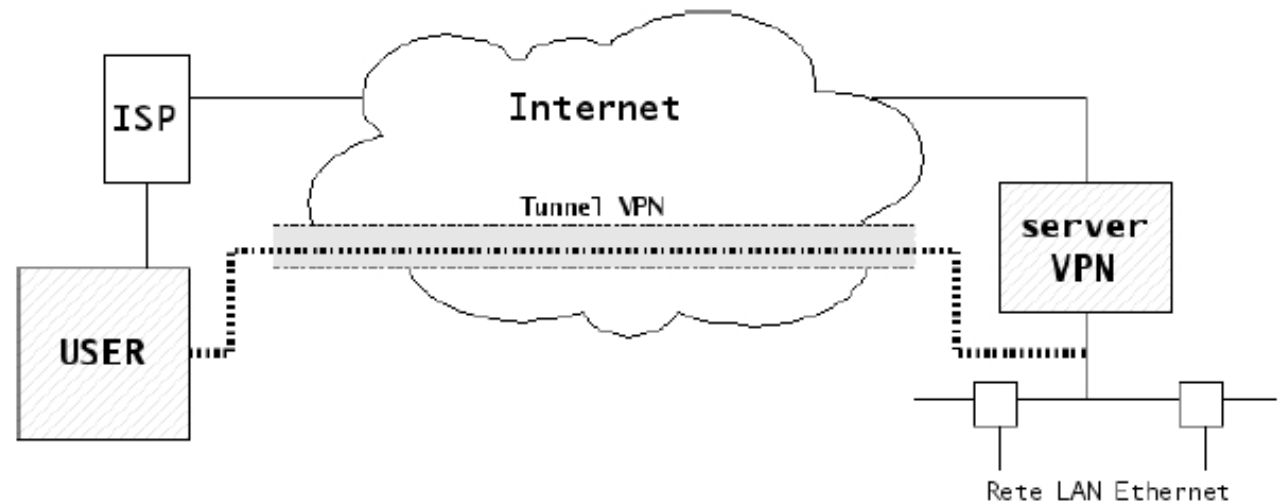
Il futuro



7) Progettazione Comunicazioni in Reti protette da NAT e Firewall

7a) Virtual Private Network - VPN

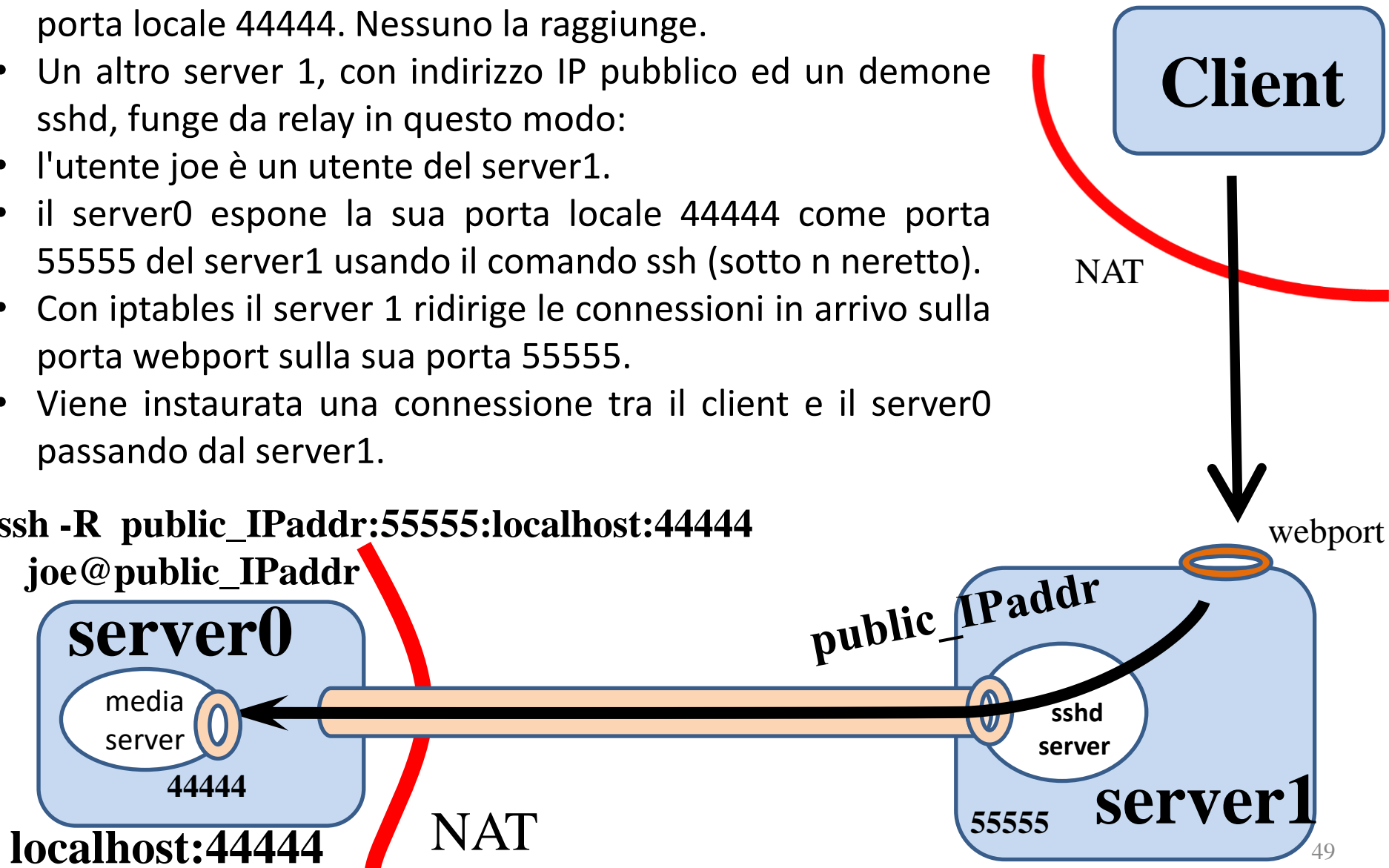
- Creare una rete privata (una sorta di LAN unica) tra host anche geograficamente molto distanti.
- **Site-to-site VPN:** sono VPN instaurate tra due reti, per esempio tra due sedi distaccate di una stessa azienda per le quali si vuole usare una rete privata unica.
- **Remote access VPN:** collegamento un singolo computer ad una rete privata tramite VPN.
- **Problema di sicurezza:** la compromissione di un computer esterno, collocato, in una rete di un dominio grazie ad una VPN, rischia di compromettere tutta la rete del dominio.



7b) Esposizione di porte remote mediante **ssh remote port forwarding**

- Un server 0 è dietro un NAT e espone un servizio sulla sua porta locale 44444. Nessuno la raggiunge.
- Un altro server 1, con indirizzo IP pubblico ed un demone sshd, funge da relay in questo modo:
- l'utente joe è un utente del server1.
- il server0 espone la sua porta locale 44444 come porta 55555 del server1 usando il comando ssh (sotto n neretto).
- Con iptables il server 1 ridirige le connessioni in arrivo sulla porta webport sulla sua porta 55555.
- Viene instaurata una connessione tra il client e il server0 passando dal server1.

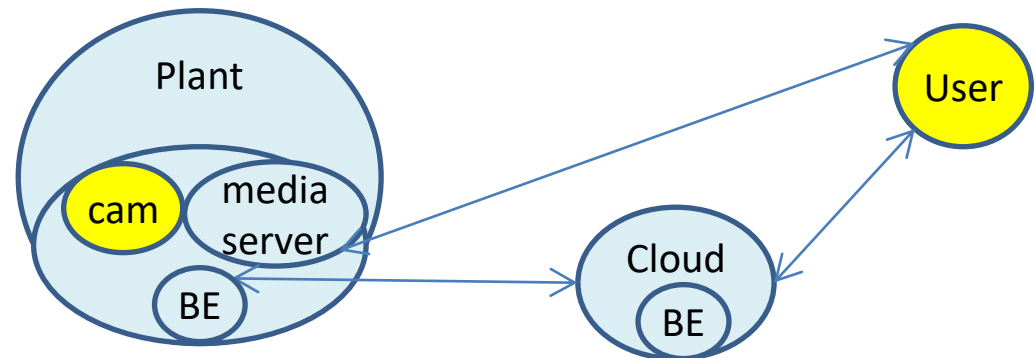
ssh -R public_IPaddr:55555:localhost:44444
joe@public_IPaddr



7c) Scenario di Integrazione

es: video controllo da remoto con superamento firewall

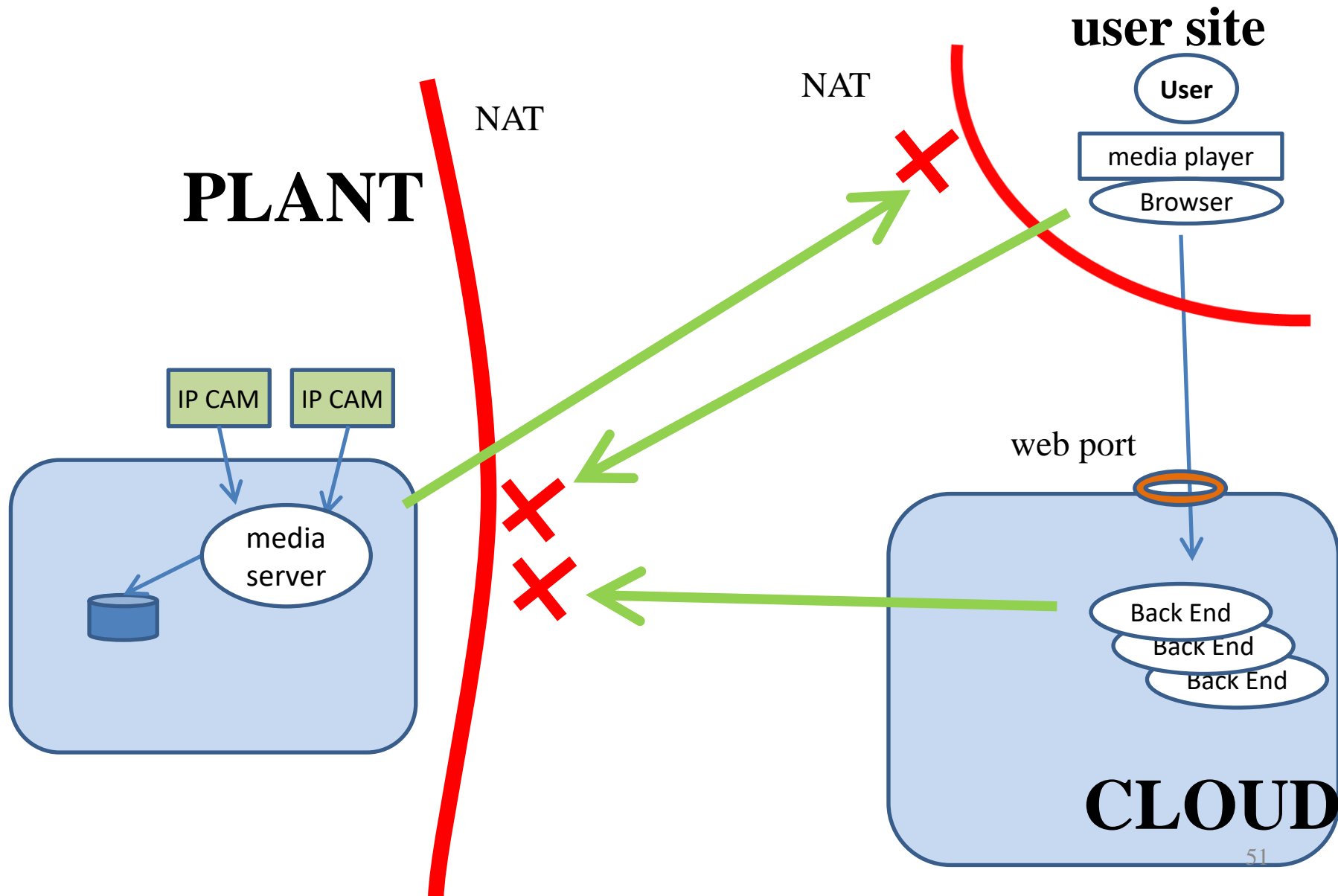
- Varie macchine automatiche, dotate di videocamera, in diversi stabilimenti.
- Utenti, sparsi nel mondo, con dispositivi vari, vogliono vedere la macchina.
- No vincoli sui dispositivi ---> video streaming a base http
- Necessario controllo sull'identità dell'utente ---> server centrale
- Possibile presenza NAT e firewall davanti ai plant e agli utenti (reti private)



- **Live Streaming,**
- crea video a run-time
- visualizzazione da remoto
- 4 attori software
 - **media server**, prende flussi audio video da videocamera e li codifica opportunamente.
 - Può essere la videocamera stessa , se ha caratteristiche avanzate.
 - Uno o più **utenti con un browser**, nel plant o fuori
 - Un servizio per signalling.
 - **Un servizio di distribuzione dei contenuti** che può essere
 - **centralizzato in cloud**, opera come relay dei flussi video del media server,
 - o **decentralizzato** attiva connessione diretta tra utente e media server.

Problema di NAT (anche di tipo simmetrico)

Impossibile collegamento diretto, occorre intermediario



Probabilità di Instaurare Connessione Diretta

NON si riesce ad instaurare una connessione diretta tra due end-system quando entrambi gli end-system sono protetti da NAT simmetrici.

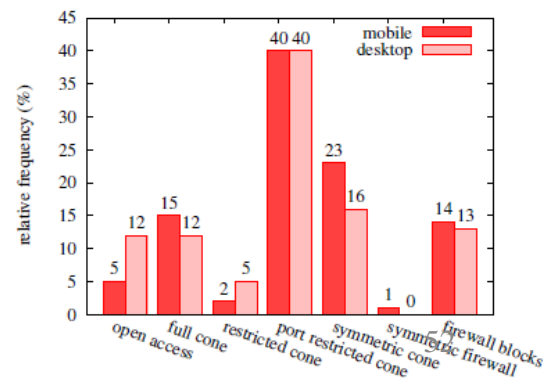
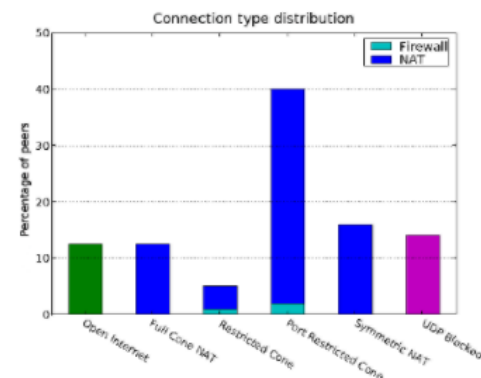
La probabilità di instaurare una connessione diretta dipende perciò dalla distribuzione e tipologia dei NAT presenti in rete.

Vari studi sperimentali e teorici stimano che:

- gli end-system protetti da NAT simmetrici siano un 16-23% del totale (per sistemi desktop o mobile rispettivamente).
- la probabilità perciò che entrambi gli end-system siano schermati da NAT simmetrici non è esagerata, ma va considerata.

La maggior presenza di port-restricted NAT è dovuto al fatto che i grandi provider di rete prediligono i port-restricted NAT/firewall a quelli simmetrici perché questi ultimi limitano il riutilizzo degli indirizzi e delle porte di protocollo dell'host che funge da NAT.

Va inoltre considerato che alcuni firewall bloccano il traffico UDP, quindi è bene utilizzare protocolli di trasporto a base HTTP, se possibile.



Riferimenti Bibliografici

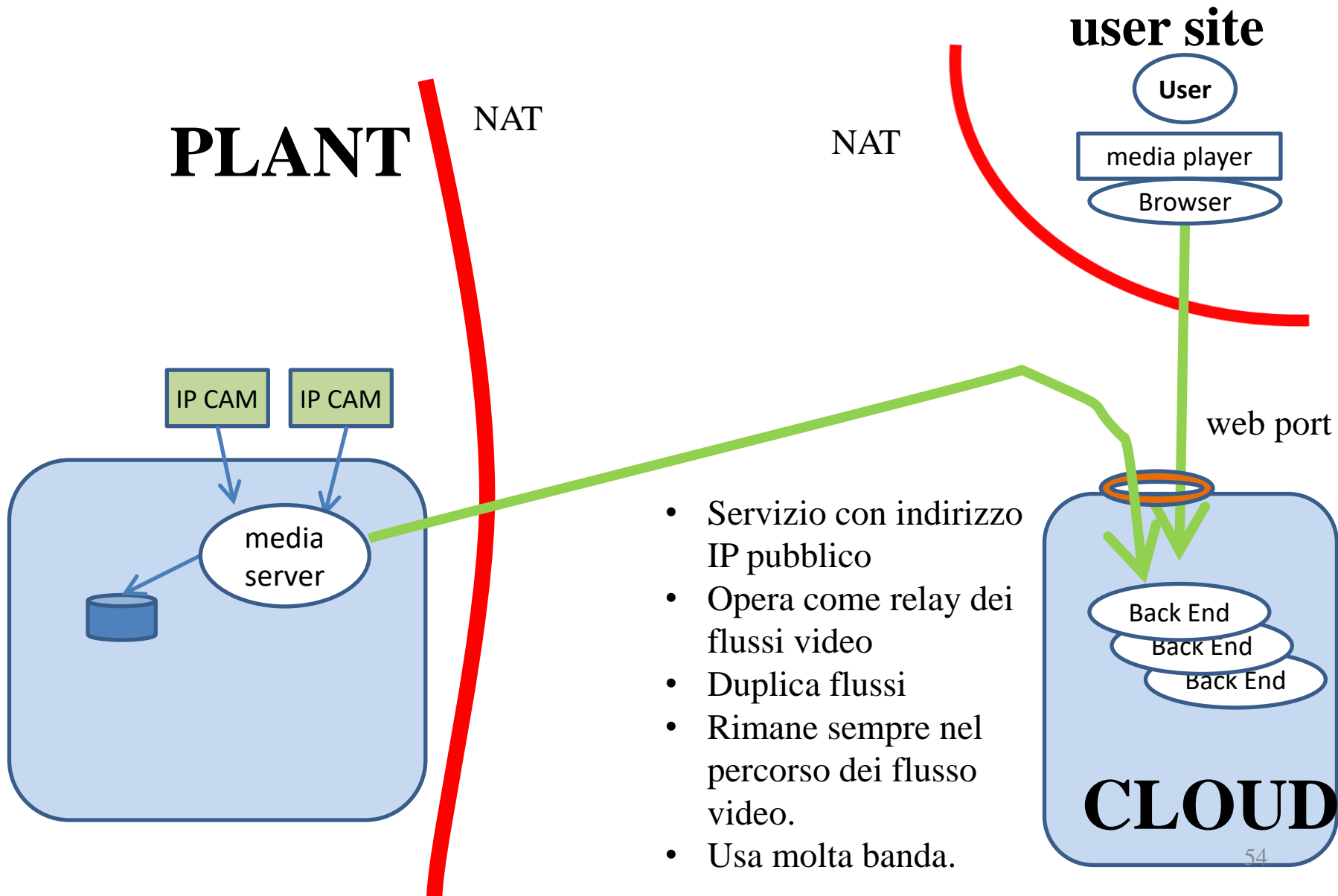
"Defining and Understanding Smartphone Churn over the Internet: a Measurement Study", V. Bilicki et al., 14-th IEEE International Conference on Peer-to-Peer Computing, 2014.

"A Measurement of NAT and Firewall Characteristics in Peer to Peer Systems", L D'Acunto, J Pouwelse, H Sips - Proc. 15-th ASCI Conference, 2009.

"Anatomy: A Look Inside Network Address Translators", Geoff Huston, The Internet Protocol Journal - Volume 7, Number 3, 2004, CISCO Press.

Superamento NAT

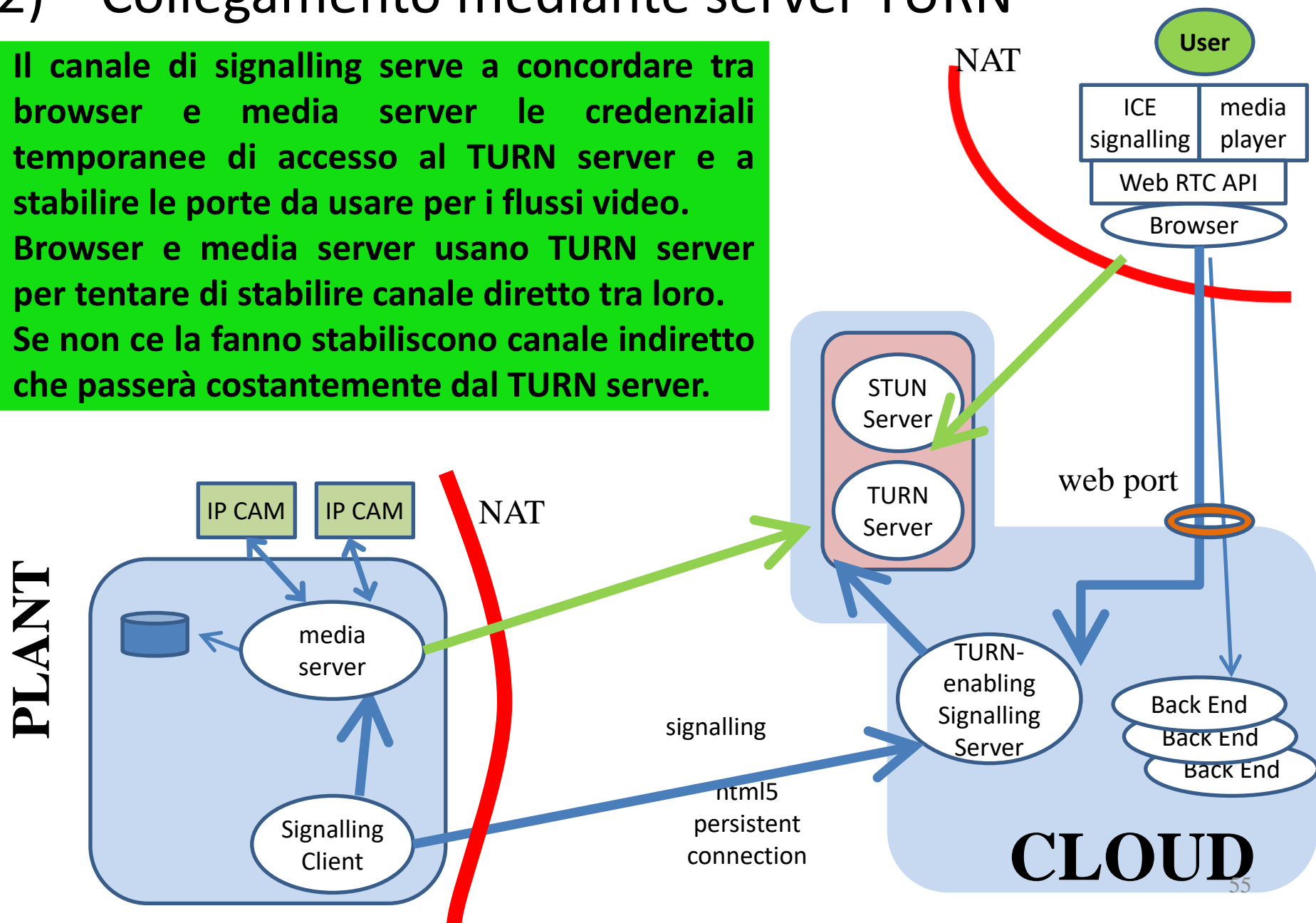
1) Collegamento mediante Intermediario in Cloud



Superamento NAT

2) Collegamento mediante server TURN

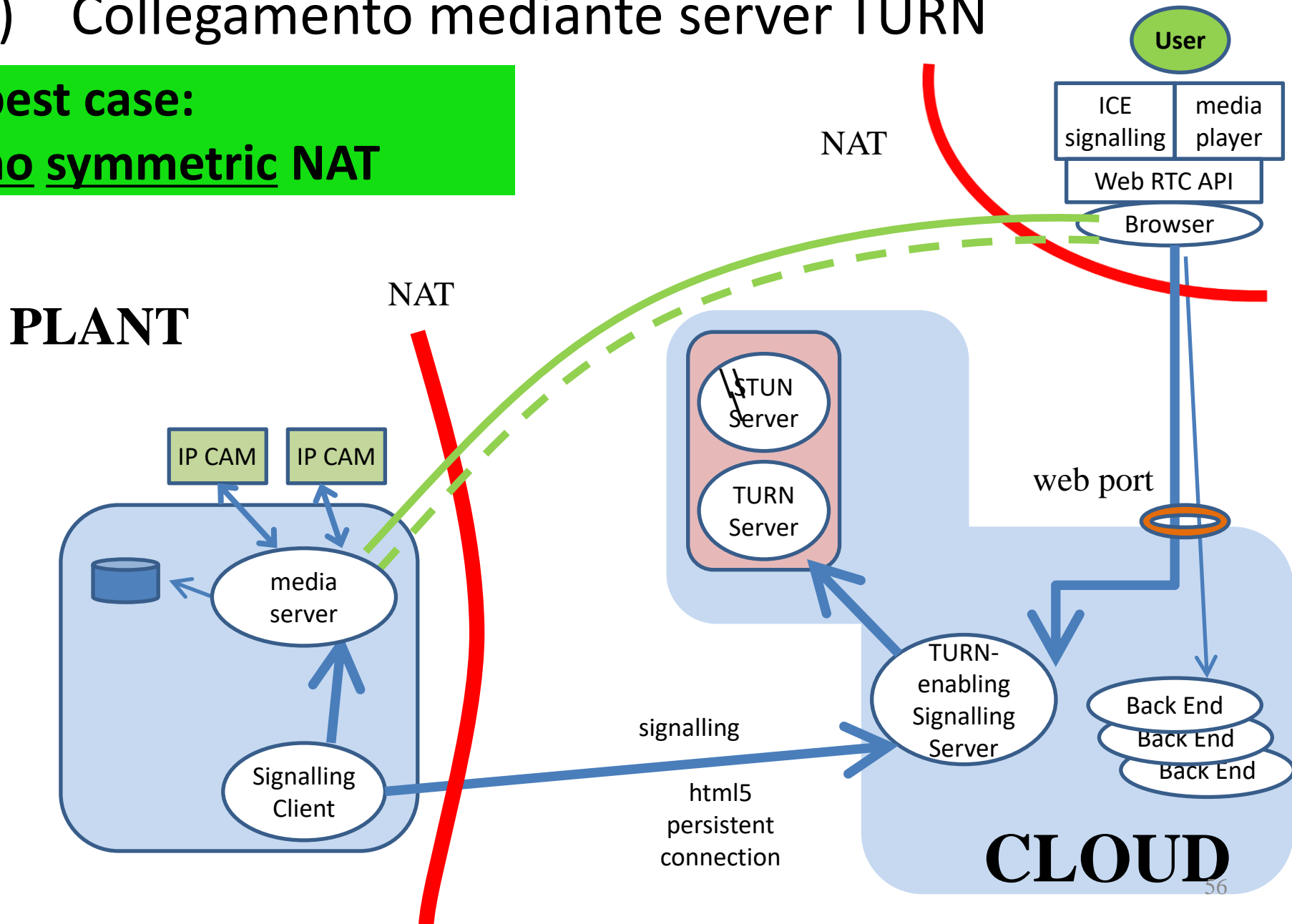
- Il canale di signalling serve a concordare tra browser e media server le credenziali temporanee di accesso al TURN server e a stabilire le porte da usare per i flussi video.
- Browser e media server usano TURN server per tentare di stabilire canale diretto tra loro.
- Se non ce la fanno stabiliscono canale indiretto che passerà costantemente dal TURN server.



Superamento NAT

2) Collegamento mediante server TURN

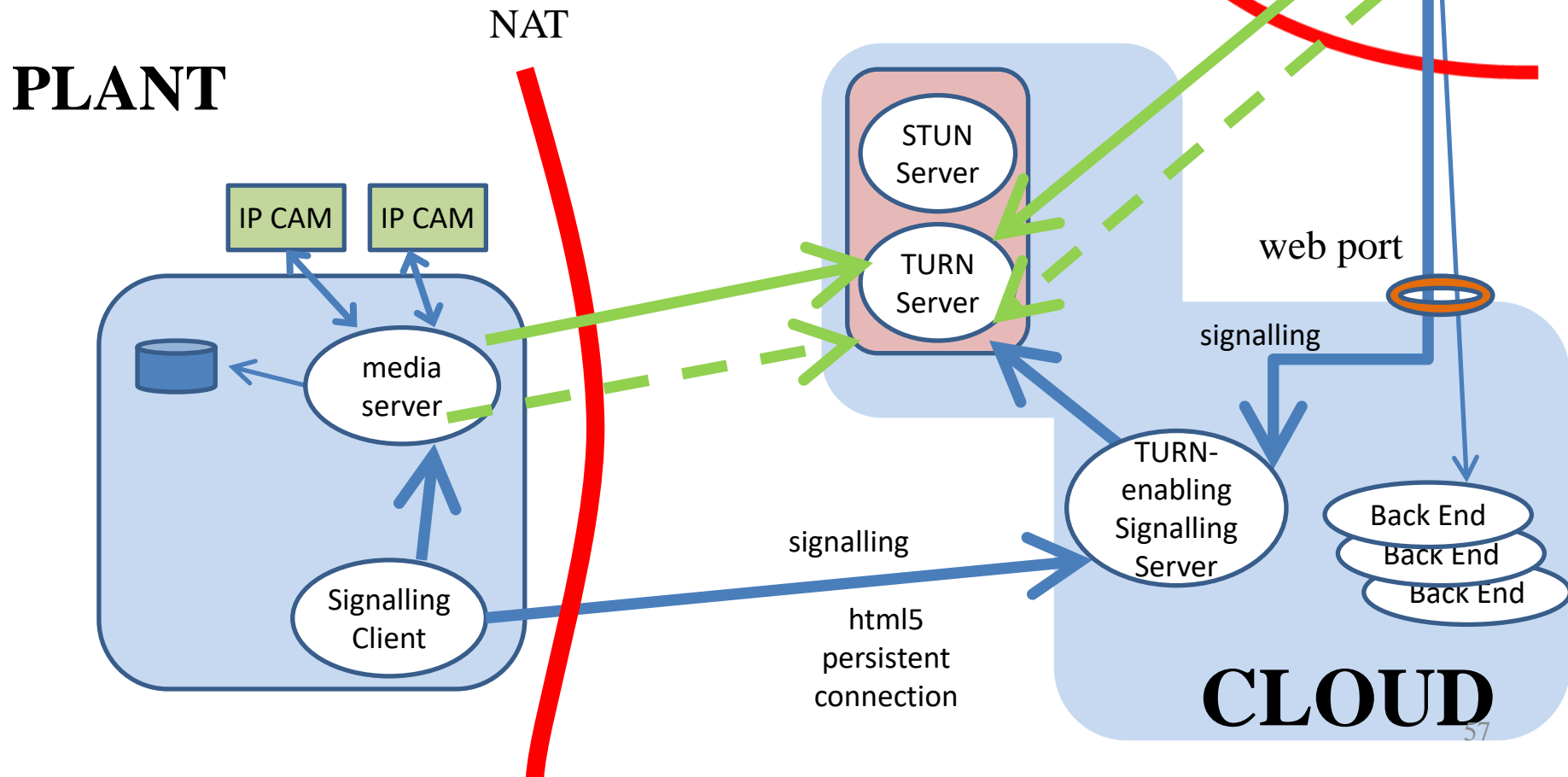
**best case:
no symmetric NAT**



Superamento NAT

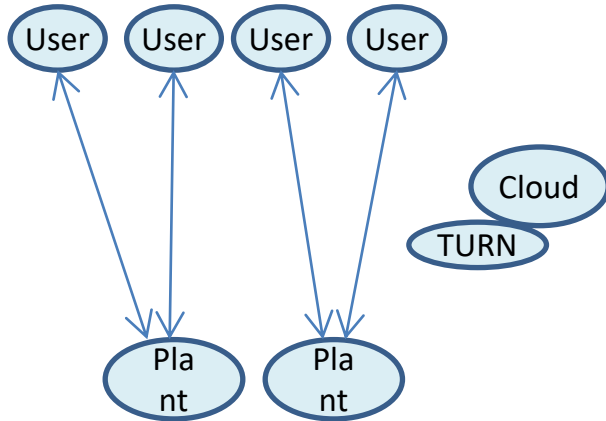
2) Collegamento mediante server TURN

**worst case:
symmetric NAT on both sides**

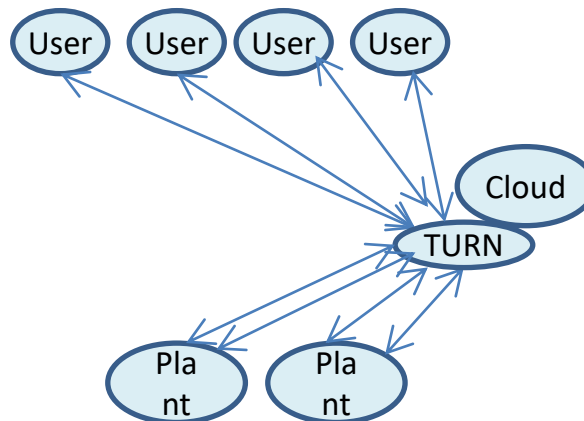


Comparazione Flussi Video

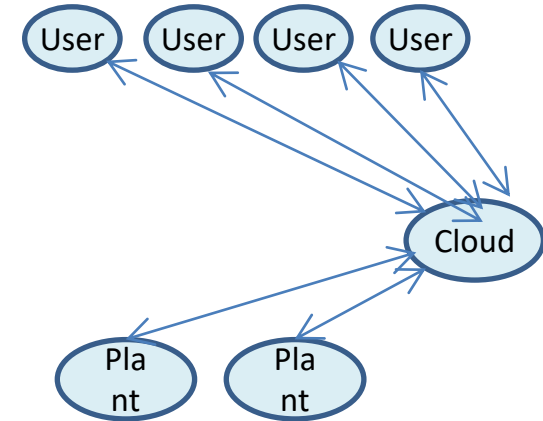
WebRTC & TURN best case



WebRTC & TURN worst case



Microsoft Azure Live Event



NOTA BENE:

- Il WORST CASE dell'architettura WebRTC&TURN è peggio dell'architettura Microsoft Live Event perché da ciascun media server locale escono tanti flussi video quanti sono gli user che usano le telecamere di quel plant, invece che un solo flusso.
- I flussi in uscita da cloud/TURN verso gli user, invece, rimangono uguali nei due casi.
- L'architettura WebRTC&TURN è preferibile solo nell'ipotesi che abbiamo pochi utenti (sarebbe meglio 1 solo) che vogliono accedere contemporaneamente alla stessa videocamera da una postazione fuori dal plant.
- L'architettura Microsoft Azure Live Event invece è pre-dimensionata (e si paga la prenotazione delle risorse di rete) per mandare il video di una videocamera fino a circa 100 utenti contemporaneamente (servizio base prevede infatti 600Mbps in uscita).

Analisi Costi Servizio Live Streaming di Azure

Il costo del servizio complessivo è dato da 5 componenti:

- | | |
|---|----------------|
| 1) Costo di Instaurazione del Channel (1 per ciascuna videocamera) | FREE |
| 2) Costo del Servizio di Streaming dal Channel, basico, no codifica, max 600 Mbps, tariffa a consumo, si pagano i minuti di servizio. | 0.835 €/hour |
| 3) Costo della codifica in Cloud. Non usato. Codifica in locale. | - - - - |
| 4) Costo della banda entrante in Azure | FREE |
| 5) Costo della banda uscente da Azure verso gli User per consumi entro i 10 TB/month | 0.074 € per GB |

Consideriamo il costo MENSILE per lo streaming di 1 videocamera.

Ipotizziamo che da Azure esca un flusso di 1.6 Mbps verso ciascun user.

Ipotizziamo che lo streaming tramite cloud venga usato, ogni giorno, da un solo user alla volta, per un totale di X ore giornaliere

Allora il costo mensile per X ore giornaliere sarà dato da:

$$\begin{aligned}\text{COSTOMENSILE}(X) &= X * [(30\text{gg} * 0.835\text{€/h}) + ((1.6\text{Mbps}/(8*1000)) * 3600\text{s} * 30\text{gg} * 0.074\text{€/GB})] = \\ &= X * [30 * 0.835\text{€/h} + (21.6\text{GB} * 0.074\text{€/GB})] = \\ &= X * (30 * 0.835\text{€} + 21.6 * 0.074\text{€})\end{aligned}$$

Per $X=1$ un'ora giornaliera di video Inspection su una videocamera, il costo sarà

$$\text{COSTOMENSILE}(1\text{ora al giorno}) = 26.6484 \text{ €}$$