

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Introduzione

Franco CALLEGATI

Dipartimento di Informatica: Scienza e Ingegneria

A.A. 2022-2023



Comunicazione

- Dal latino : cum-munis
 - Cum = insieme
 - Munis = compito, dovere
- Significato prevalente oggi
 - Condivisione e scambio di informazione
- Cosa serve per comunicare?

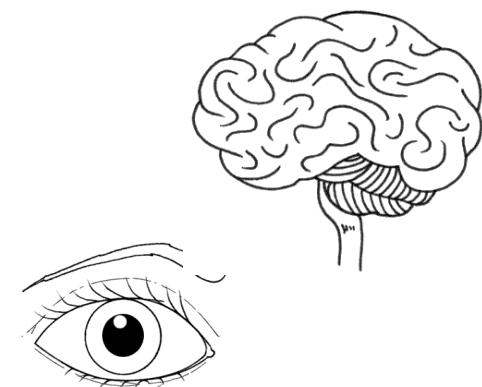
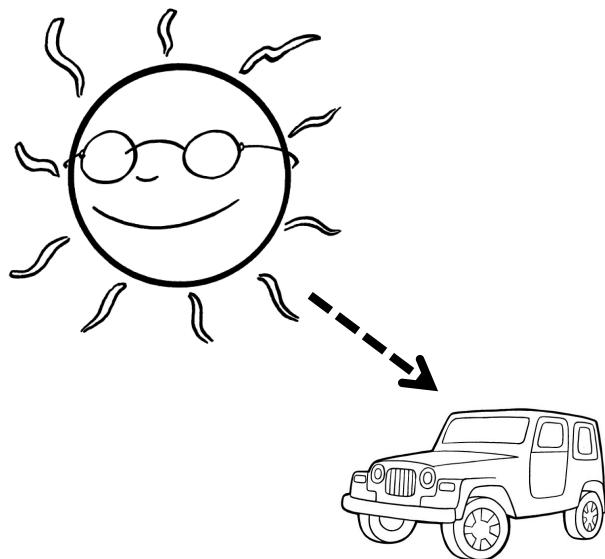
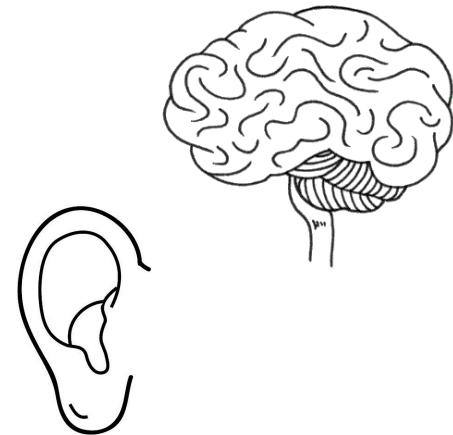
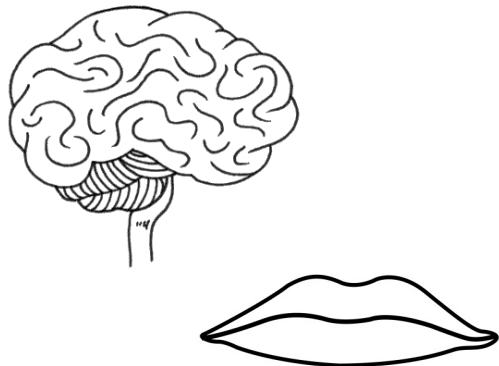
Codificatore/
Trasmettitore

Trasporto

Ricevitore/
decodificatore



Esempi



Limiti?

- Le comunicazioni umane naturali richiedono prossimità e immediatezza
- Come estendere la comunicazione
 - Tempo
 - Registrare l'informazione per un uso successivo
 - Trasmissione orale
 - Scrittura, pittura, fotografia
 - Memorizzazione elettronica
 - Spazio
 - Portare lontano dalla sorgente l'informazione



Telecomunicazione?

- Tele ($\text{\textit{T\v{e}\lambda\v{e}}}$)
 - Greco antico : tele = lontano
- Telecomunicazione = comunicare lontano
- Il problema tecnico
 - **Trasportare** l'informazione nello spazio
 - Quantità
 - Qualità



Le torri di guardia genovesi in Corsica

- Tante torri in posizioni strategiche sulle coste
 - Ogni torre deve avere in vista le due adiacenti



Fantasia: Il signore degli anelli

- Con il fuoco di Amon Din il regno di Gondor chiede aiuto al regno Rohans
 - Stima della velocità di propagazione del messaggio in base alla descrizione del libro: 50 km/h
 - <http://scienceblogs.com/dotphysics/2010/07/30/how-fast-is-the-beacon-of-gond/>

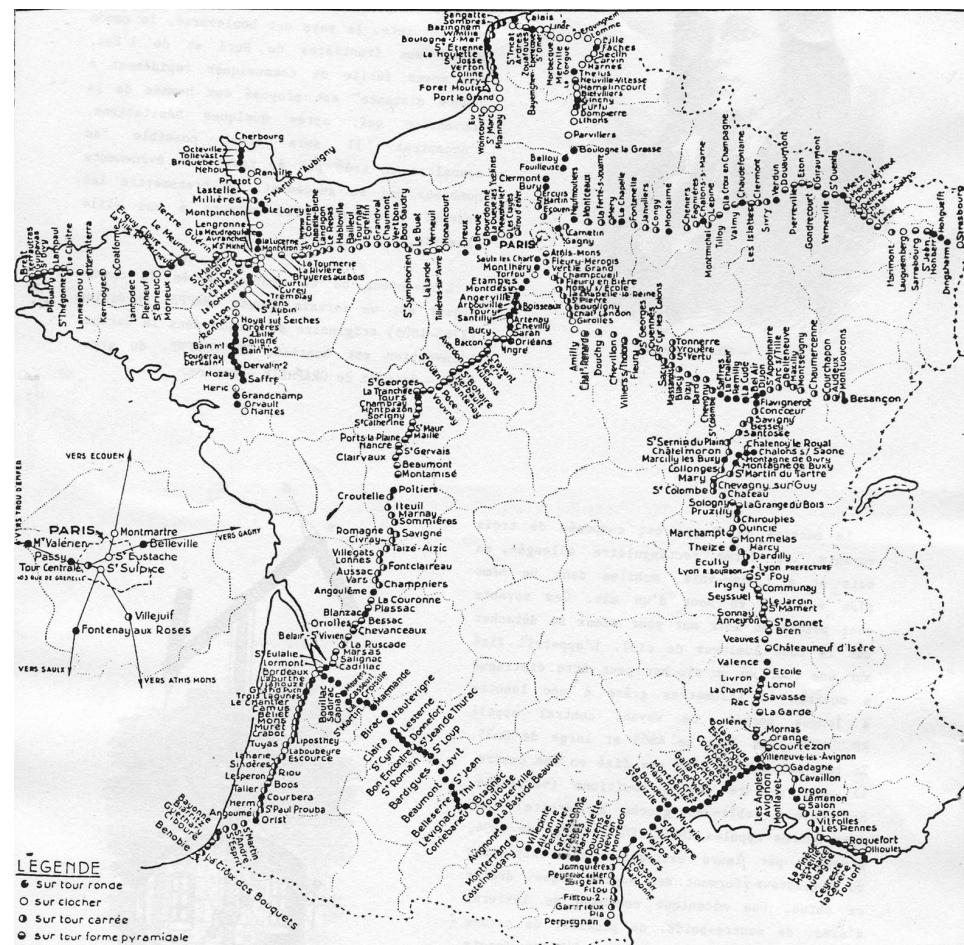


1792

Telegrafo Claude Chappe

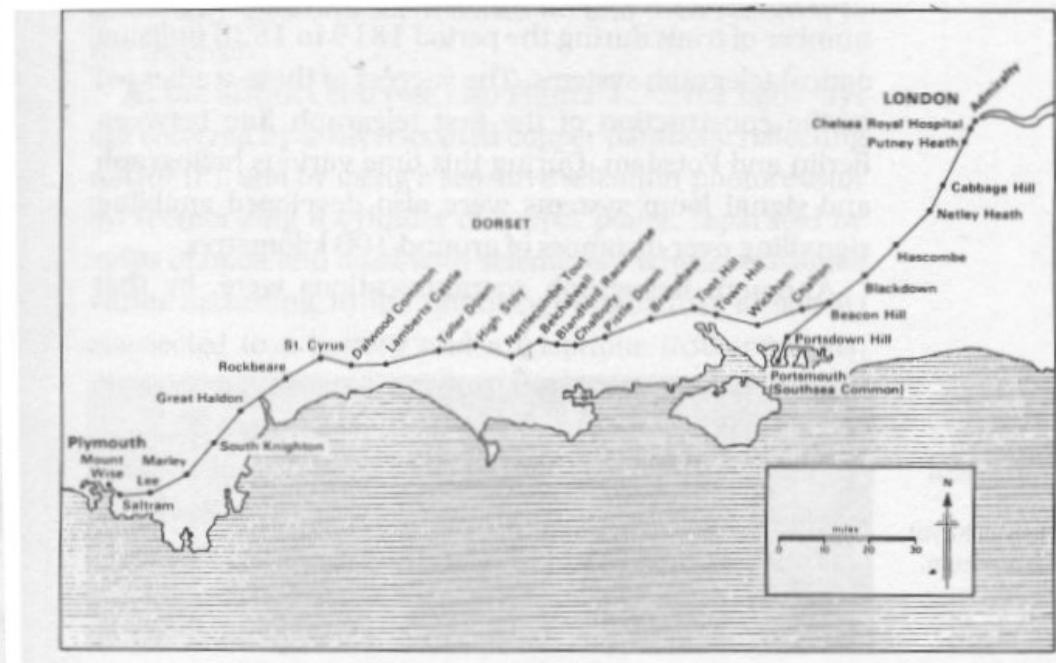
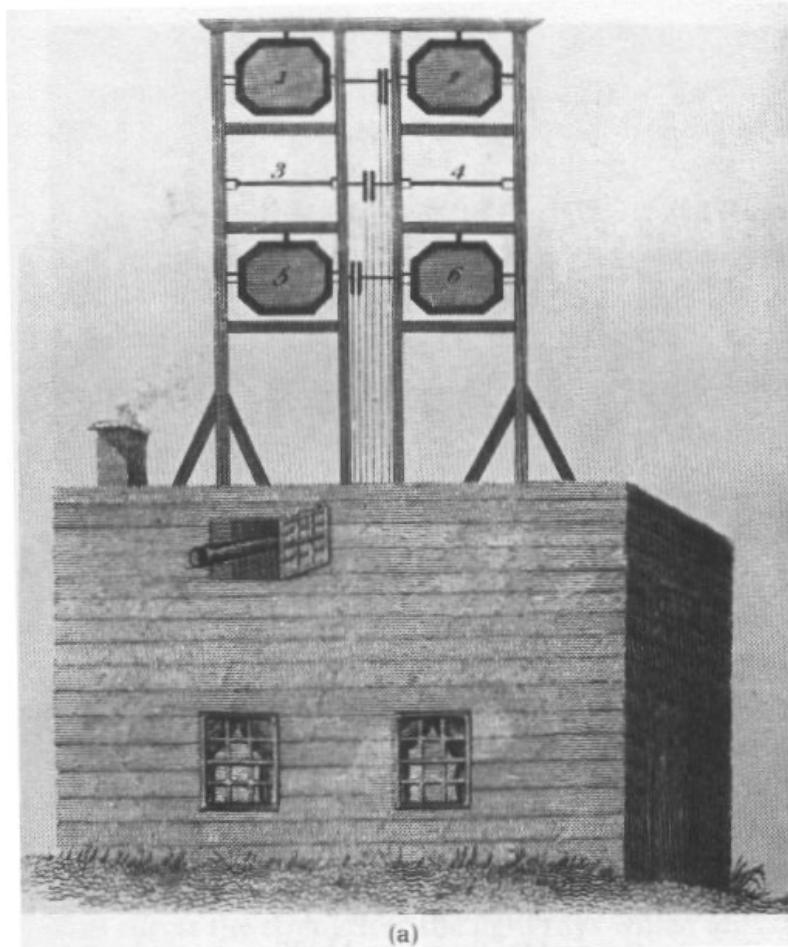


Speed : 500 km/h



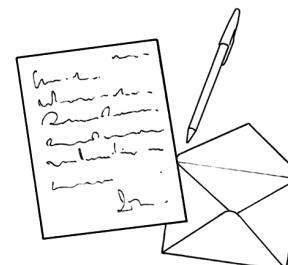
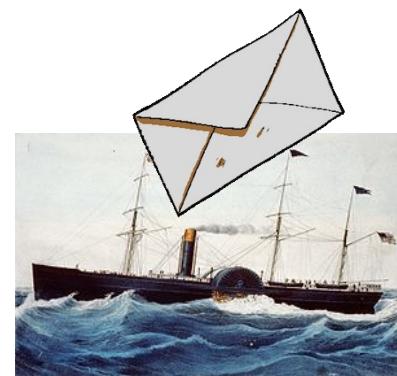
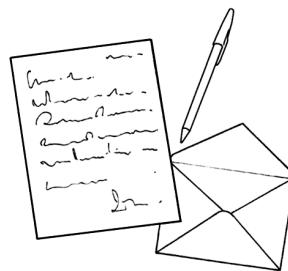
1795 Telegrafo Shutter

15 min to transmit from Plymouth to London



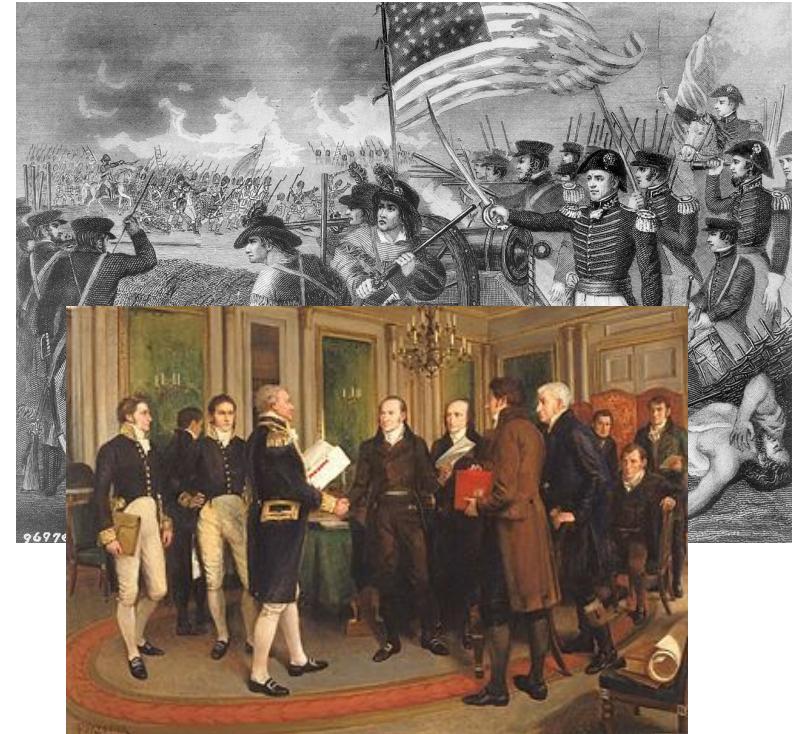
Un piccolo esempio

- Nel 1850 la Regina Vittoria vuole comunicare con Franklin Pierce presidente degli USA
 - Liverpool – New York in circa 10 giorni
 - Round Trip Time circa 1 mese



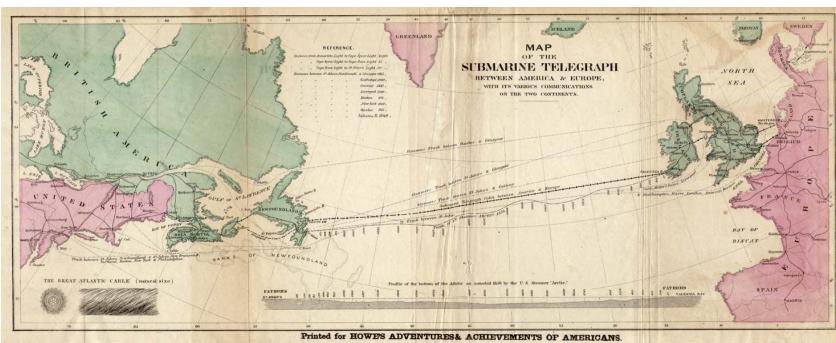
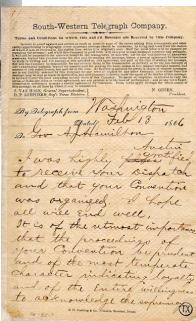
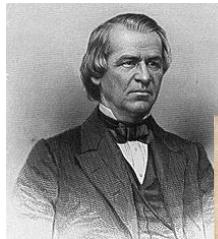
Un significativo esempio

- Battaglia di New Orleans
 - Dal 8/1/1815 al 18/1/1815
 - Inghilterra contro USA
 - 2500 morti circa
- Ma ... c'era già la pace
 - Trattato di Ghent firmato il 24/12/1814
 - Reso noto alle truppe inglesi il 18/1/1815
 - Ratificato dal governo americano in Febbraio



Ancora un piccolo esempio

- Nel 1865 la Regina Vittoria vuole comunicare con Johnson, Andrew presidente degli USA
 - Invia un telegramma sul cavo transatlantico
 - Round Trip Time poche ore





ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Il canale e la rete

Franco CALLEGATI



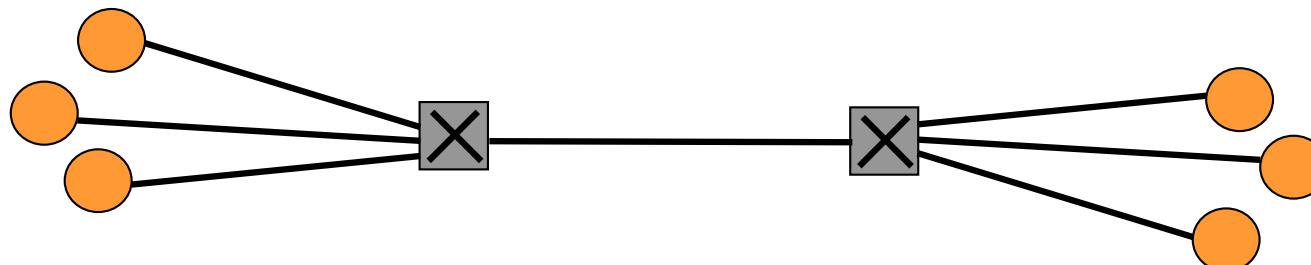
Canale

- Le telecomunicazioni utilizzano *canali di comunicazione*
- Canale = entità logica o fisica che permette il trasporto dei singoli flussi informativi fra punti (nodi) remoti nello spazio
 - **Monodirezionale**
 - L' informazione può essere trasferita in una sola direzione
 - **Bidirezionale**
 - L' informazione può essere trasferita in entrambe le direzioni
 - **Punto-punto**
 - Un nodo è collegato con un singolo nodo
 - **Punto-multipunto**
 - Un nodo può comunicare con tanti altri
 - **Broadcast**: un nodo trasmette allo stesso tempo a tutti i nodi della rete
 - **Multicast**: un nodo trasmette allo stesso tempo ad un sottoinsieme dei nodi



Perché la rete

- Due o più utenti utilizzano un mezzo fisico per realizzare un canale per comunicare
 - Configurazione statica e prefissata
 - Il telefono delle giovani marmotte
- Una **grande popolazione** di utenti vuole **condividere** un insieme di canali per comunicare a **richiesta**
 - È necessario un sistema complesso che permetta il “**riuso**” dei canali: la **rete di telecomunicazioni**
 - Deve essere definito un insieme di regole per il suo utilizzo



Componenti della rete

- **Terminali**

- Fungono da interfaccia con l' utente finale
- Codificano l' informazione in modo consono ad essere trasferita in rete

- **Mezzi trasmissivi o collegamenti**

- Permettono il trasferimento di uno o più flussi di informazione fra punti remoti nello spazio

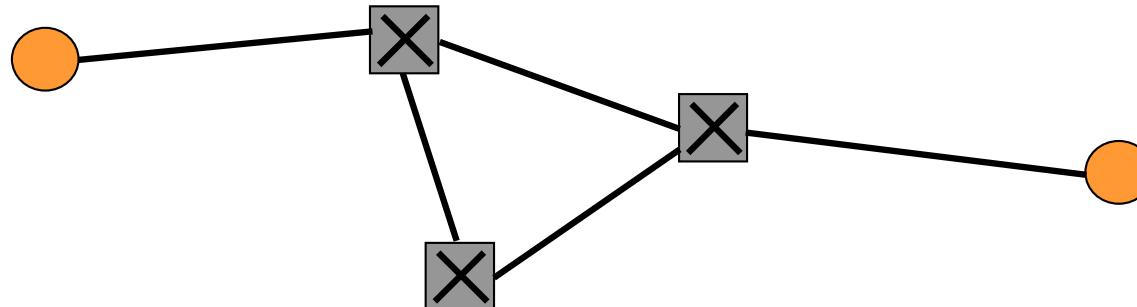
- **Nodi di commutazione** 

- Utilizzano i mezzi trasmissivi al fine di creare canali di comunicazione sulla base delle richieste degli utenti



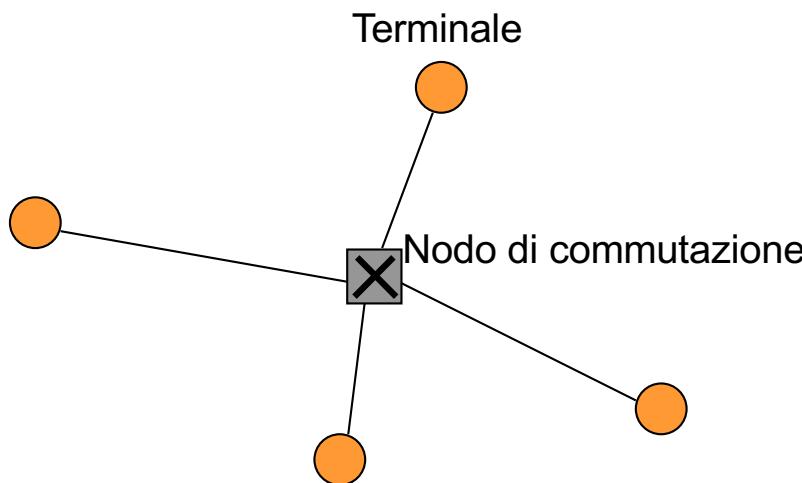
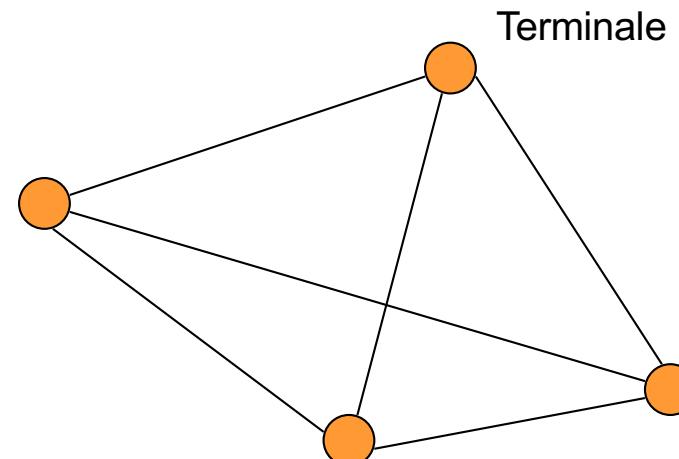
Topologie di rete

- Descrizione geometrica di una rete
- Rami (archi)
 - Linee di collegamento fra due nodi della rete
- Nodi
 - Punti che si trovano agli estremi dei collegamenti
- La rete è descrivibile tramite un *grafo*



Maglia completa e stella

- Un collegamento per ogni coppia di nodi
- N nodi implicano $N(N-1)/2$ collegamenti
 - Grande resistenza ai guasti
 - Complessità e costo

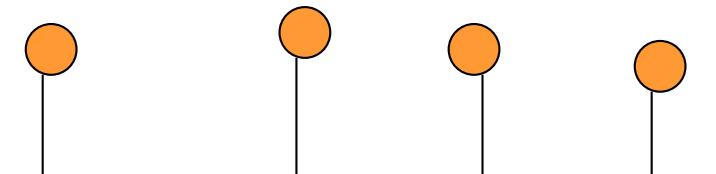
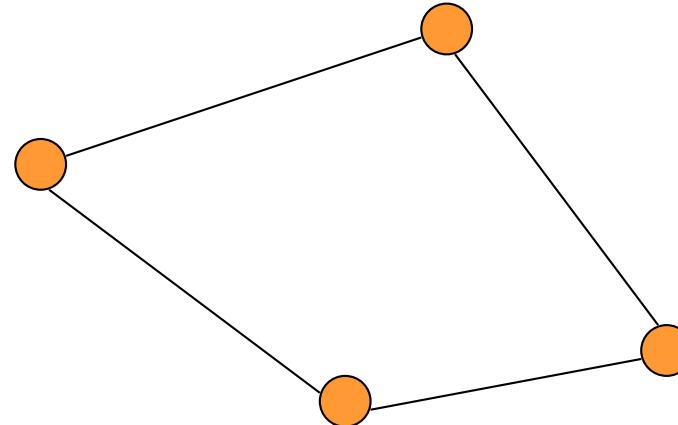


- N collegamenti
- Centro stella (attivo o passivo) deve smistare le informazioni
 - Minor costo
 - Minore resistenza ai guasti



Anello e bus

- Anello
 - Anelli monodirezionali
 - Se un collegamento si interrompe la rete si guasta
 - Anelli bidirezionali
 - Maggiore complessità per maggiore resistenza ai guasti



- Bus Attivo o passivo
 - Tipicamente semplice ed economico
 - Poco resistente ai guasti

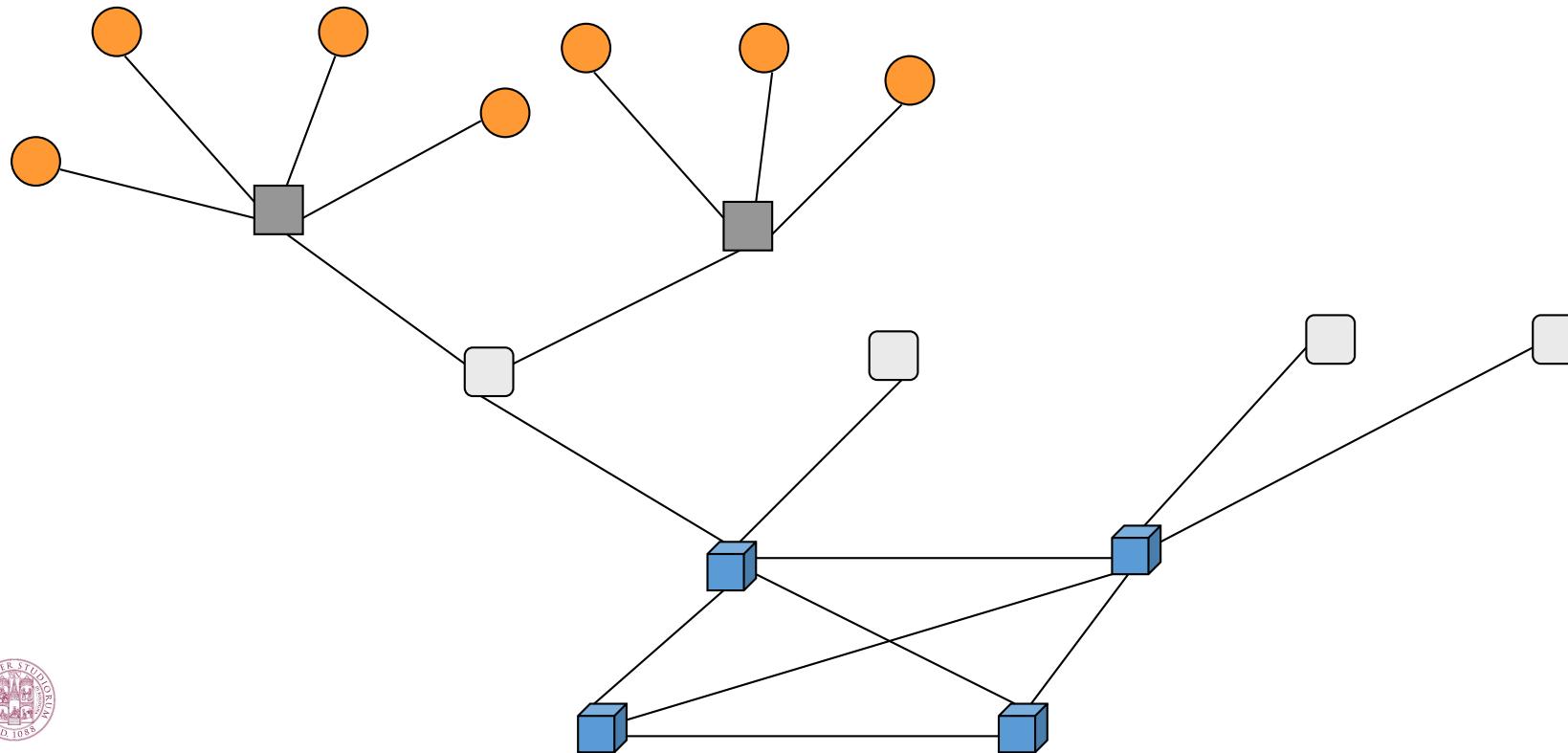
Bus bidirezionale

- Il mezzo di trasmissione è condiviso
 - È necessario definire un opportuno protocollo di accesso (MAC)

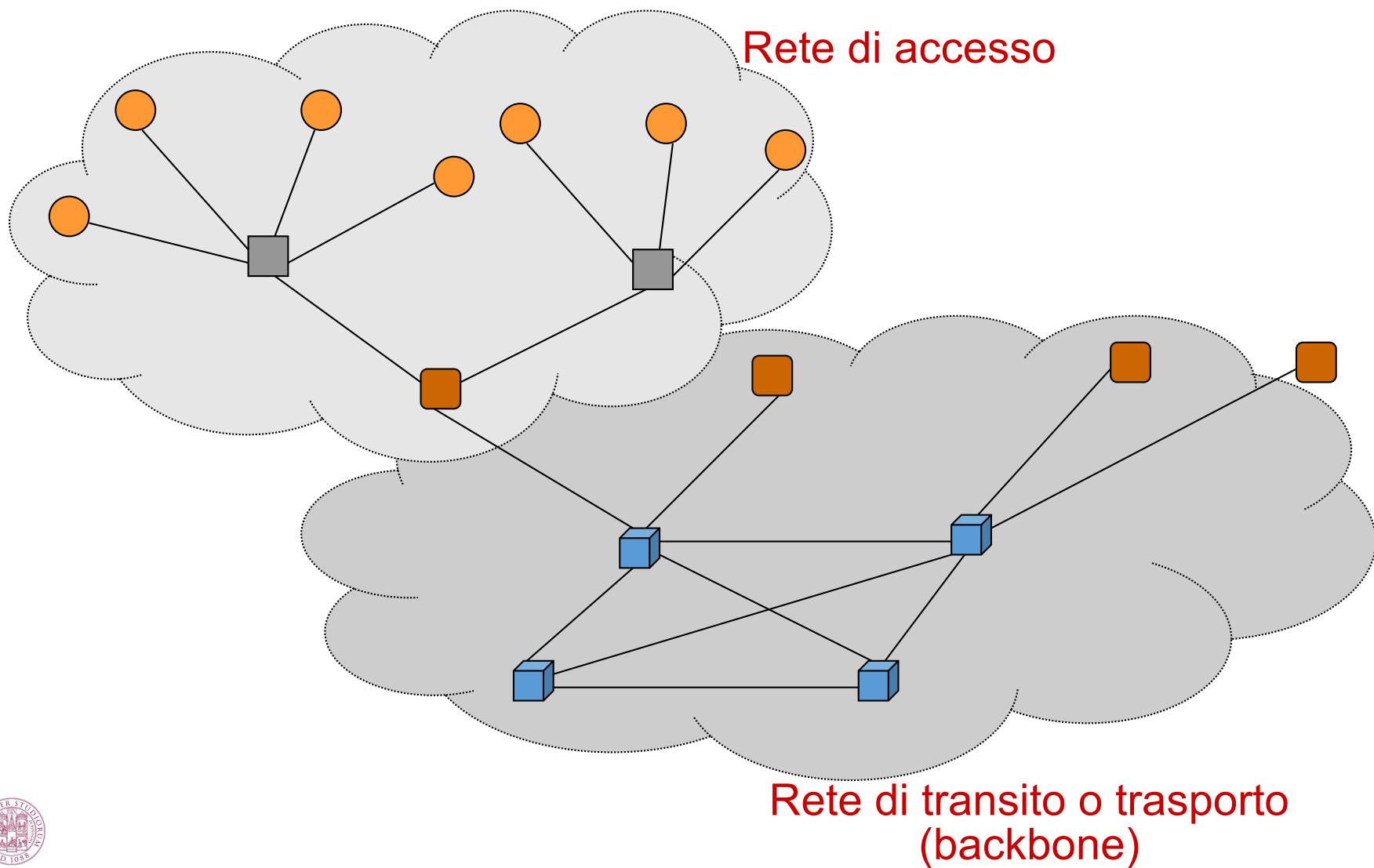


Rete gerarchica

- Organizzata su più livelli
 - Terminali connessi ai nodi periferici
 - Nodi periferici connessi tramite nodi intermedi
 - Interconnessione a lunga distanza con nodi di transito tipicamente interconnessi a maglia completa



Rete di accesso e Rete di transito





ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Informazione, segnali, digitalizzazione

Franco CALLEGATI

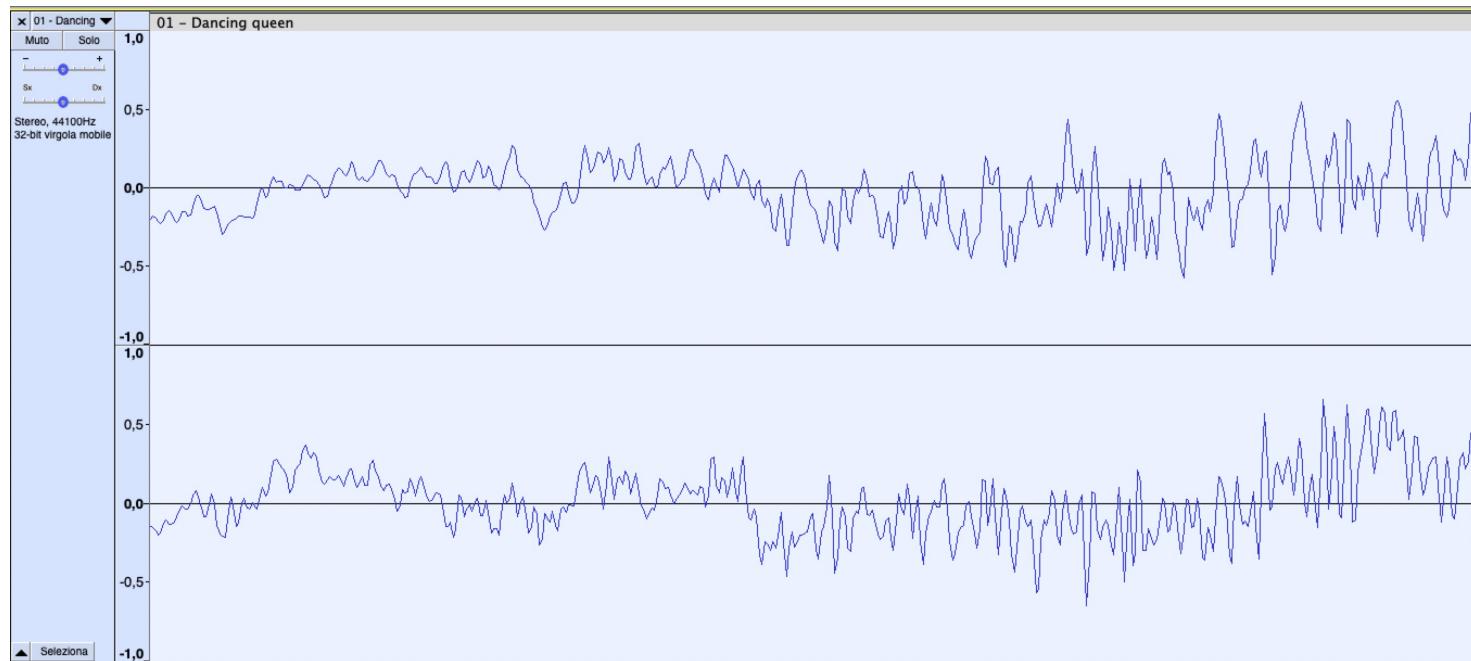
Dipartimento di Informatica: Scienza e Ingegneria

A.A. 2022-2023



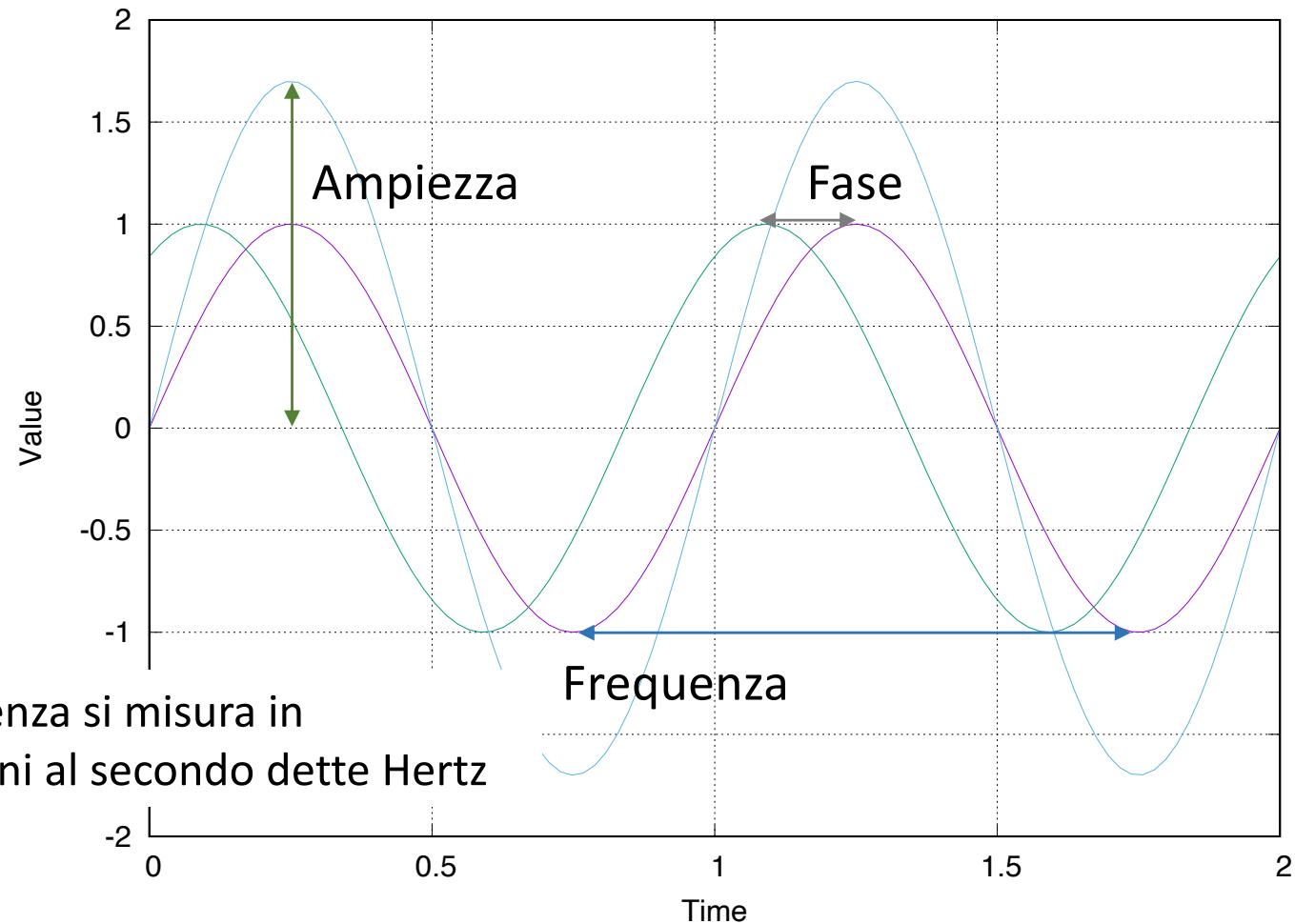
Ascoltiamo un po' di musica

- Questa figura mostra un grafico corrispondente all'ampiezza del segnale elettrico che pilota gli altoparlanti durante la riproduzione di un brano musicale
 - Il grafico mostra la variazione nel tempo
 - Le variazioni rappresentano un'informazione che corrisponde al brano musicale e permette di riprodurlo
 - È quello che normalmente viene chiamato un **segnale analogico**



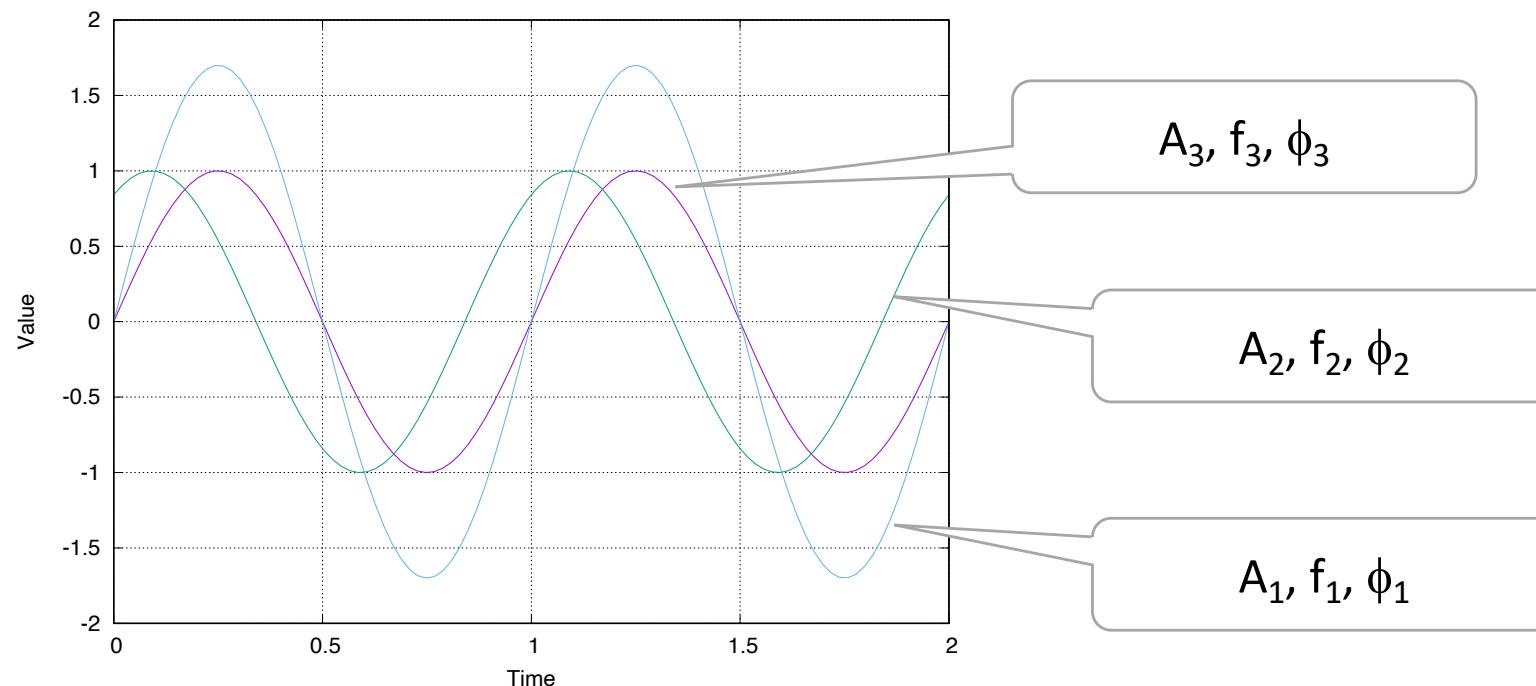
Un segnale molto semplice: la sinusoide

$$A \sin(2\pi ft + \phi)$$



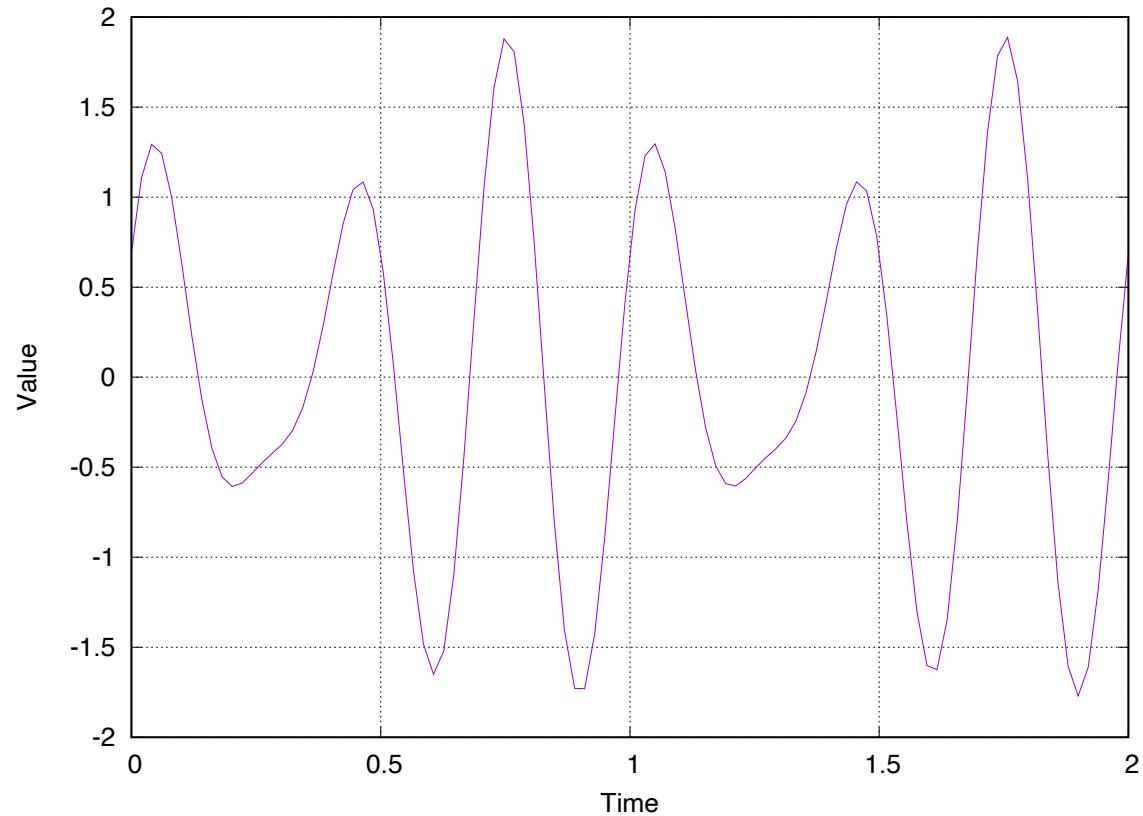
Descrivere il segnale

- La singola sinusoide ha un andamento temporale noto
- Le tre grandezze fondamentali sono sufficienti per caratterizzarla



Un caso più complesso

$$1.2 \sin(2\pi 3t) + 0.7 \sin(2\pi 4t + 1.4)$$

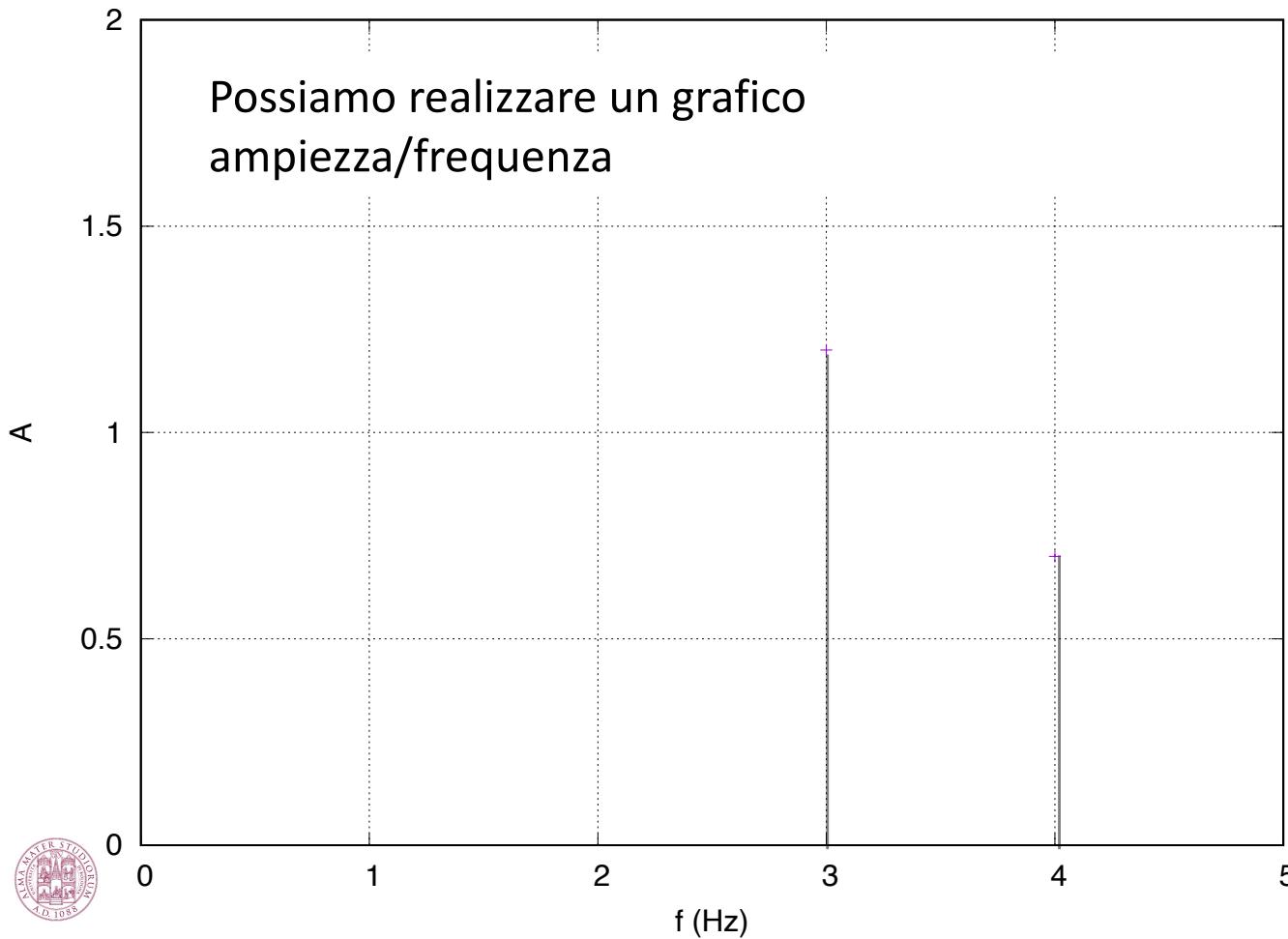


- L'andamento di questo segnale nel tempo è piuttosto difficile da descrivere



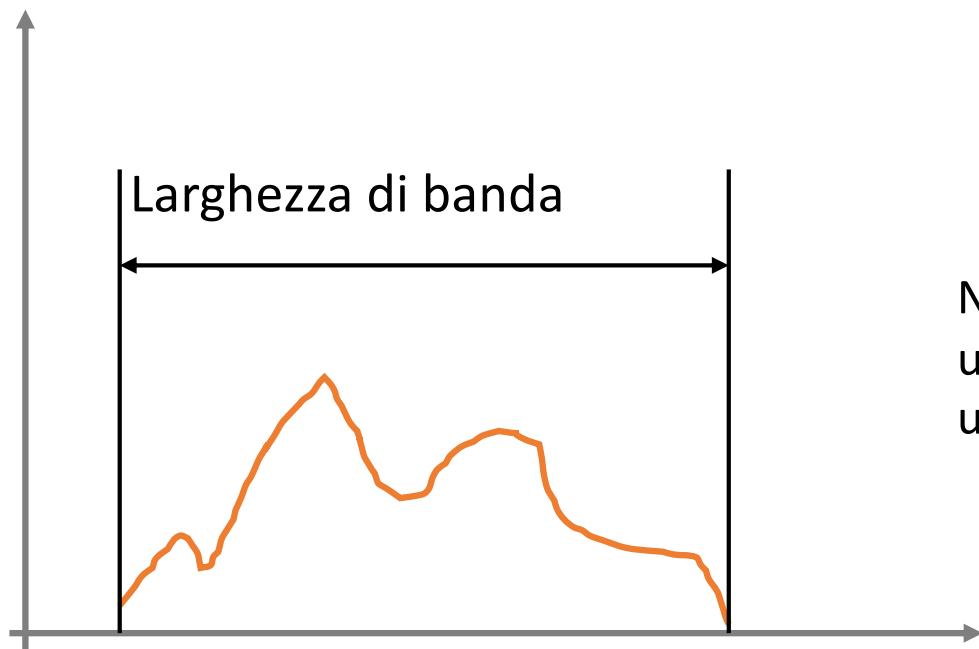
Ma

- Se ci focalizziamo su ampiezza, fase e frequenza descrivere il segnale è molto più semplice



La voce e i suoni

- L'apparato vocale umano può generare suoni con frequenze approssimativamente fra 0 e 8000 Hz
- L'orecchio umano può percepire suoni approssimativamente a frequenze fra 20 e 15000 Hz

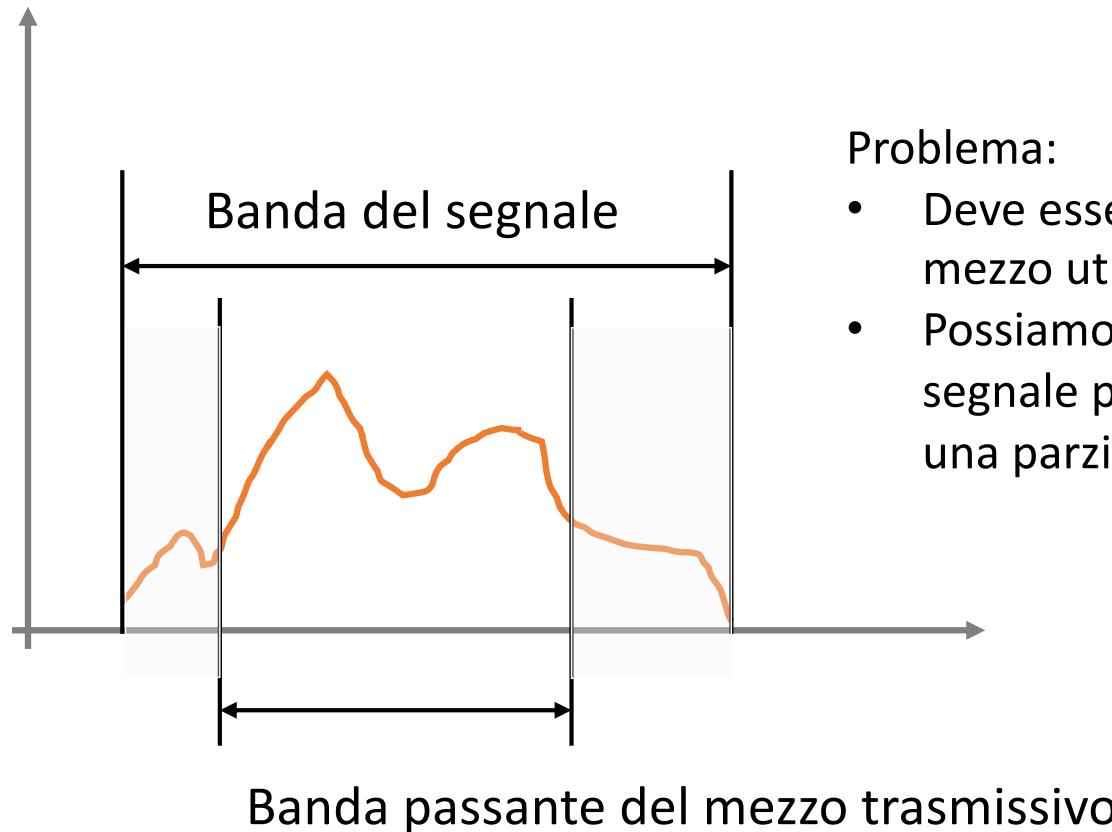


Normalmente la larghezza di banda è una misura del grado di complessità di un segnale



Trasmissione dei segnali

- Le tecnologie per la trasmissione dei segnali hanno normalmente dei limiti



Problema:

- Deve esserci una coerenza fra segnale e mezzo utilizzato per trasmetterlo
- Possiamo chiederci se l'usabilità del segnale possa essere compatibile con una parziale perdita di frequenze?



Telefono

- Nella telefonia analogica tradizionale il segnale sonoro prodotto dall'utente viene filtrato e limitato nella «banda fonica» che va da 300 a 3400 Hz
- Una buona parte delle frequenze della voce umana è fuori dalla banda fonica
- Le frequenze fondamentali per rendere il palato comprensibile (nelle lingue occidentali almeno) concentrata intorno ai 2 KHz
- Il sistema di trasmissione ha una banda passante inferiore a quella del segnale ma sufficiente a permetterne l'utilizzo



La transizione al digitale

- Conversione Analogico/Digitale (ADC)
 - Il segnale analogico viene rappresentato in forma binaria, utilizzando le cifre 0,1 dette **bit**
- Conversione Digitale/Analogico (DAC)
 - Una sequenza di bit viene riconvertita in una segnale analogico funzione del tempo

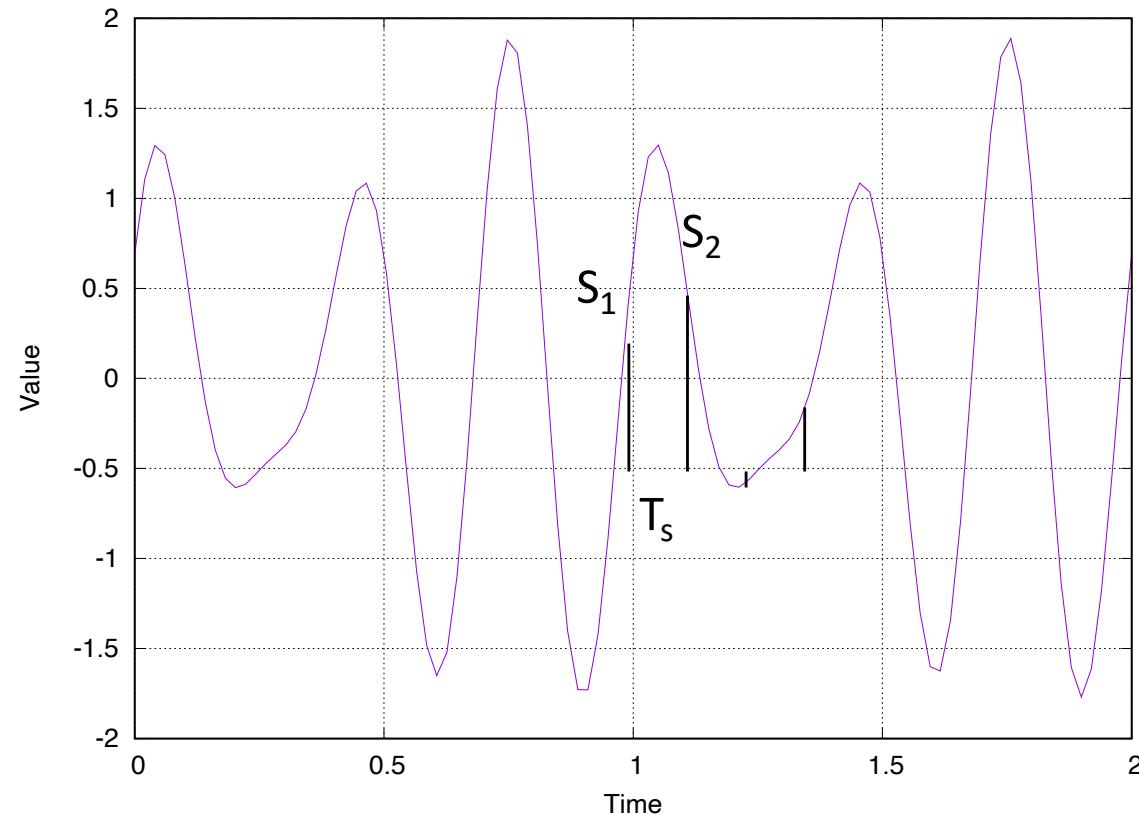


La conversione analogico/digitale

- È caratterizzata da due fasi distinte
 - Campionamento
 - Il segnale analogico viene misurato in predeterminati istanti di tempo
 - Viene prodotta una serie temporali di numeri corrispondenti alle misure effettuate
 - Quantizzazione
 - I numerici risultanti dal campionamento sono rappresentati in forma binaria utilizzando un numero di cifre predeterminato



Campionamento

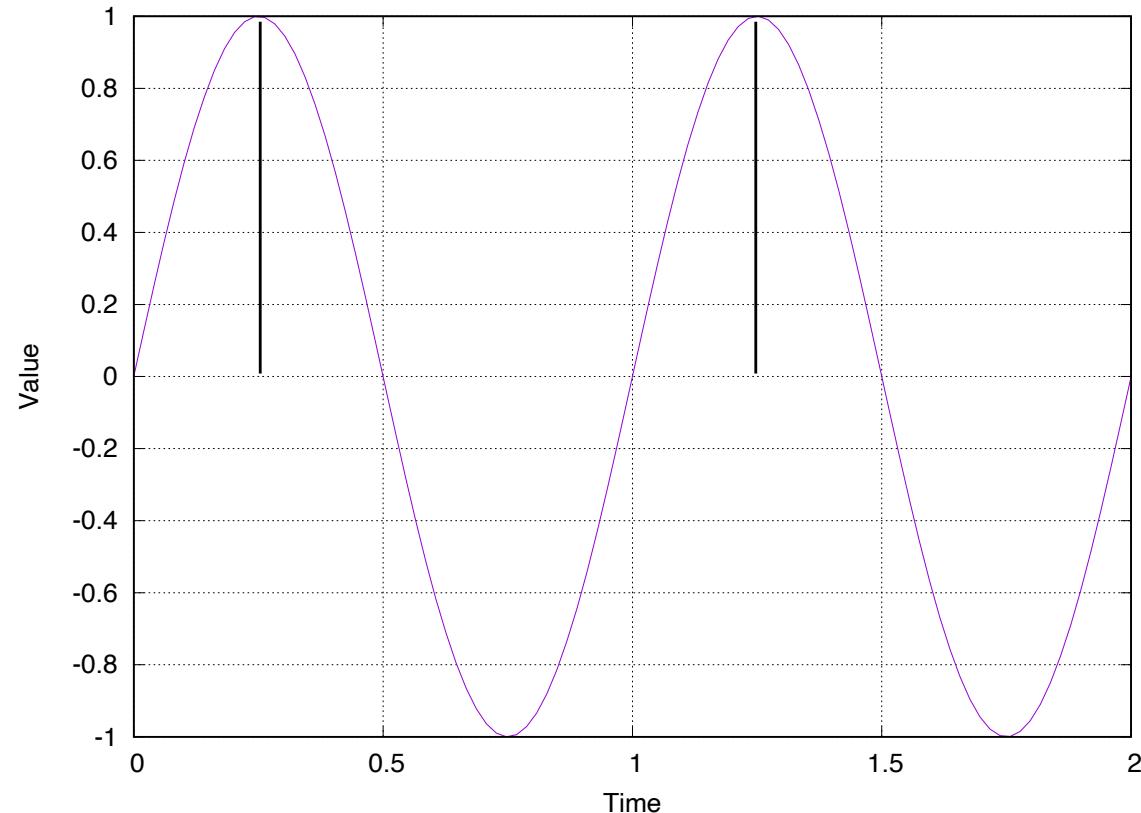


T_s intervallo di campionamento

Viene creata la sequenza numerica corrispondente al segnale analogico
 S_1, S_2, \dots, S_n



L'intervallo di campionamento

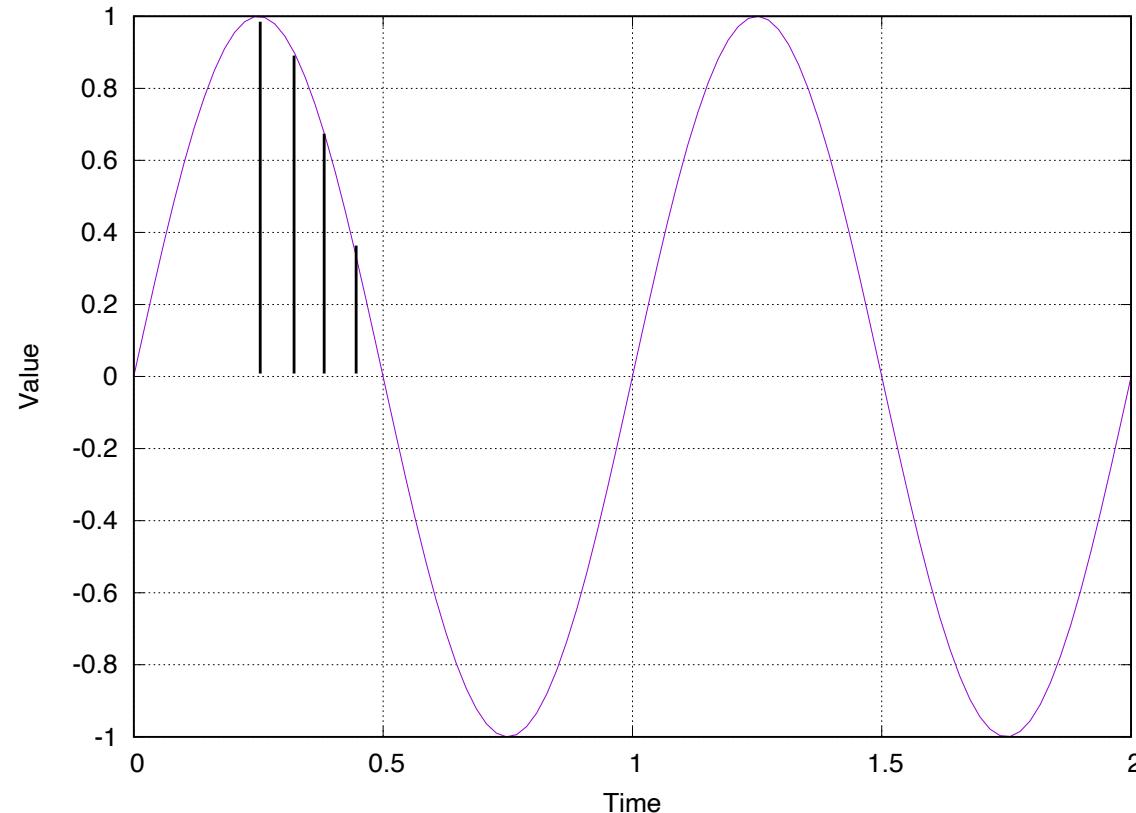


Sotto-campionamento (Sub-sampling)

- T_s grande
- Come mostra l'esempio si possono perdere facilmente importanti caratteristiche del segnale analogico



L'intervallo di campionamento



Sovra campionamento (Over-sampling)

- T_s piccolo
- La serie numerica include tantissimi valori, che posso risultare sovrabbondanti per rappresentare il segnale



Il teorema di Shannon-Nyquist

- Fornisce un criterio per determinare il giusto intervallo di campionamento
- In particolare permette di determinare il minimo intervallo di campionamento che permette di ricostruire il segnale analogico in modo integrale

$$f_s \geq 2f_M$$



Quantizzazione

- I campioni sono rappresentati in forma binaria

Sequenza numerica

$S_1, S_2, S_3, \dots, S_n$

10, 12, 15, ..., 8

Rappresentazione binaria



10 -> 1010

12 -> 1100

15 -> 1111

8 -> 0100

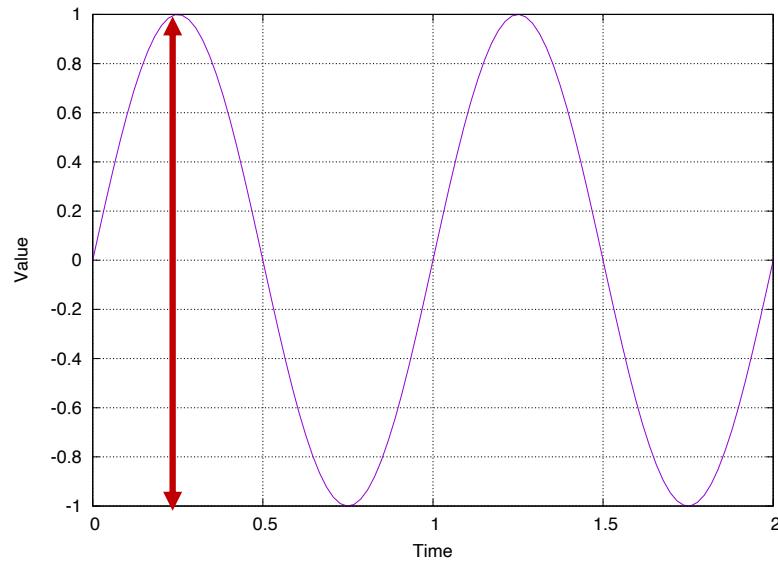


Sequenza di bit

1010 1100 1111 0100

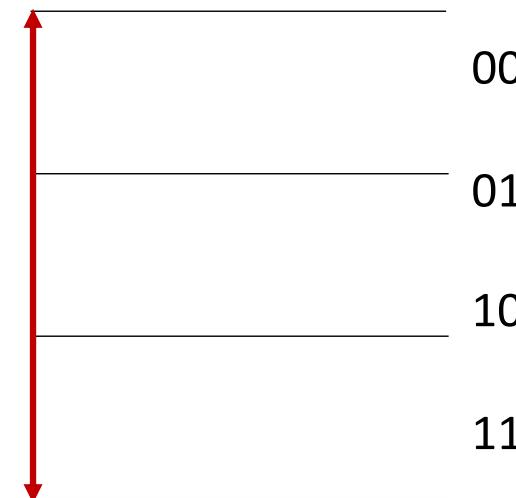


Precisione della quantizzazione

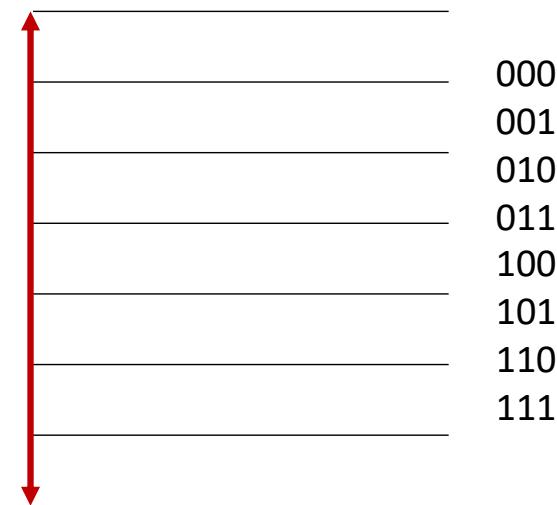


Un maggior numero di bit permette
una maggior precisione nella
rappresentazione dei campioni

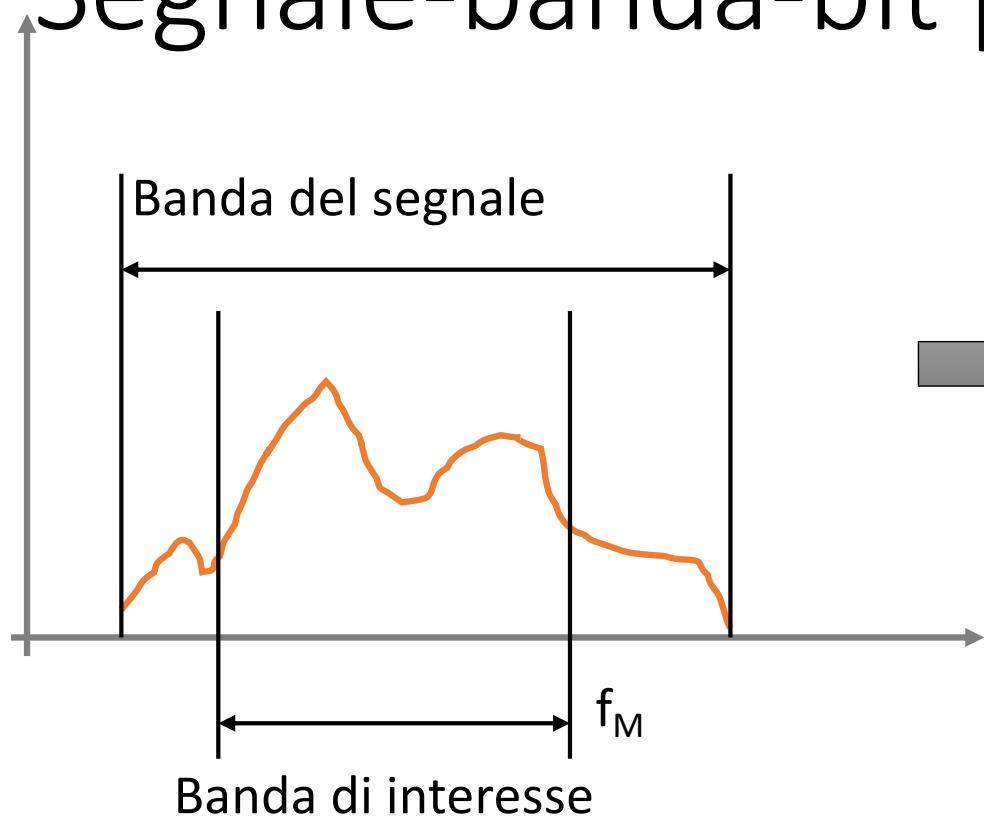
Con due 2 bit
4 numeri



Con 3 bit 8 numeri



Segnale-banda-bit per secondo



$$f_s \geq 2f_M$$

$$T_s = \frac{1}{f_s}$$

Bit al secondo

Numero di bit
Per campione

Numero di campioni
al secondo



Opportunità del digitale

- **Integrazione**

- Formato dell'informazione unificato -> una sola tecnologia di telecomunicazione

- **Computazione**

- I segnali diventano bit che possono essere elaborati dai calcolatori elettronici
 - Modifica: aggiungi, estrai, combina
 - Compressione
 - Ridotto numero di bit a parità di qualità percepita
 - Cifratura
 - Riservatezza, autenticazione





ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Reti e servizi

Franco CALLEGATI



Servizio?

- L' offerta dei servizi di telecomunicazione è molto aumentata nell' ultimo decennio
- Una vasta popolazione di utenti "umani" fa uso "intensivo" di servizi di telecomunicazione
- Esistono caratteristiche comuni e differenze
 - Tipologia di interazione nella comunicazione
 - Modalità con cui fluiscono le informazioni
 - Tipologia di informazioni
- Vediamo alcuni esempi tra i più diffusi e quindi familiari agli utenti



ITU : tassonomia dei servizi

- **Servizi interattivi:** esiste interazione fra sorgente e destinazione
 - Conversazione
 - Scambio informativo in tempo reale: telefonia, condivisione di file system
 - Messaggistica
 - Scambio informativo in tempo differito con memorizzazione: posta elettronica, SMS
 - Consultazione
 - Scambio informativo con flusso controllato dall' utente: WWW, teledidattica
- **Servizi distributivi:** la sorgente diffonde informazioni in modo indipendente ad un numero impreciso di destinazioni
 - Senza controllo di presentazione
 - L' utente di destinazione non controlla l' ordine con cui ricevere le informazione: radio/tele-diffusione
 - Con controllo di presentazione
 - L' utente di destinazione può controllare l' ordine con cui ricevere le informazione: televideo





Flusso informativo

- **Punto-punto**
 - Traferimento informativo uno a uno
 - Telefono, file transfer, posta elettronica
- **Punto multipunto (multicast)**
 - Traferimento informativo da uno a tanti
 - Mailing list, SMS a gruppi
- **Diffusivo (broadcast)**
 - Traferimento informativo da uno a tutti
 - Radio-tele diffusione
- **Monodirezionale**
 - Traferimento informativo in una sola direzione
 - Radio/tele-diffusione
 - Streaming
- **Bidirezionale simmetrico**
 - Uguale capacità per ogni direzione
 - Telefonia
- **Bidirezionale asimmetrico**
 - Diversa capacità per ogni direzione
 - ADSL

Servizi multimediali

- Servizio **monomediale**
 - Trasporta informazioni di un solo tipo
 - Trasferisce più tipologie di informazione sotto forma di un solo segnale
 - Televisione: immagine e suoni in un unico segnale analogico opportunamente costruito
- Servizio **multimediale**
 - Trasporta informazioni di almeno due tipologie diverse
 - Le diverse tipologie di informazioni sono trasportate dalla medesima rete ma con modalità distinte
 - Diversi protocolli
 - Differenti qualità di servizio (conformi alla tipologia di informazione)
 - Videoconferenza: voce e video bidirezionale

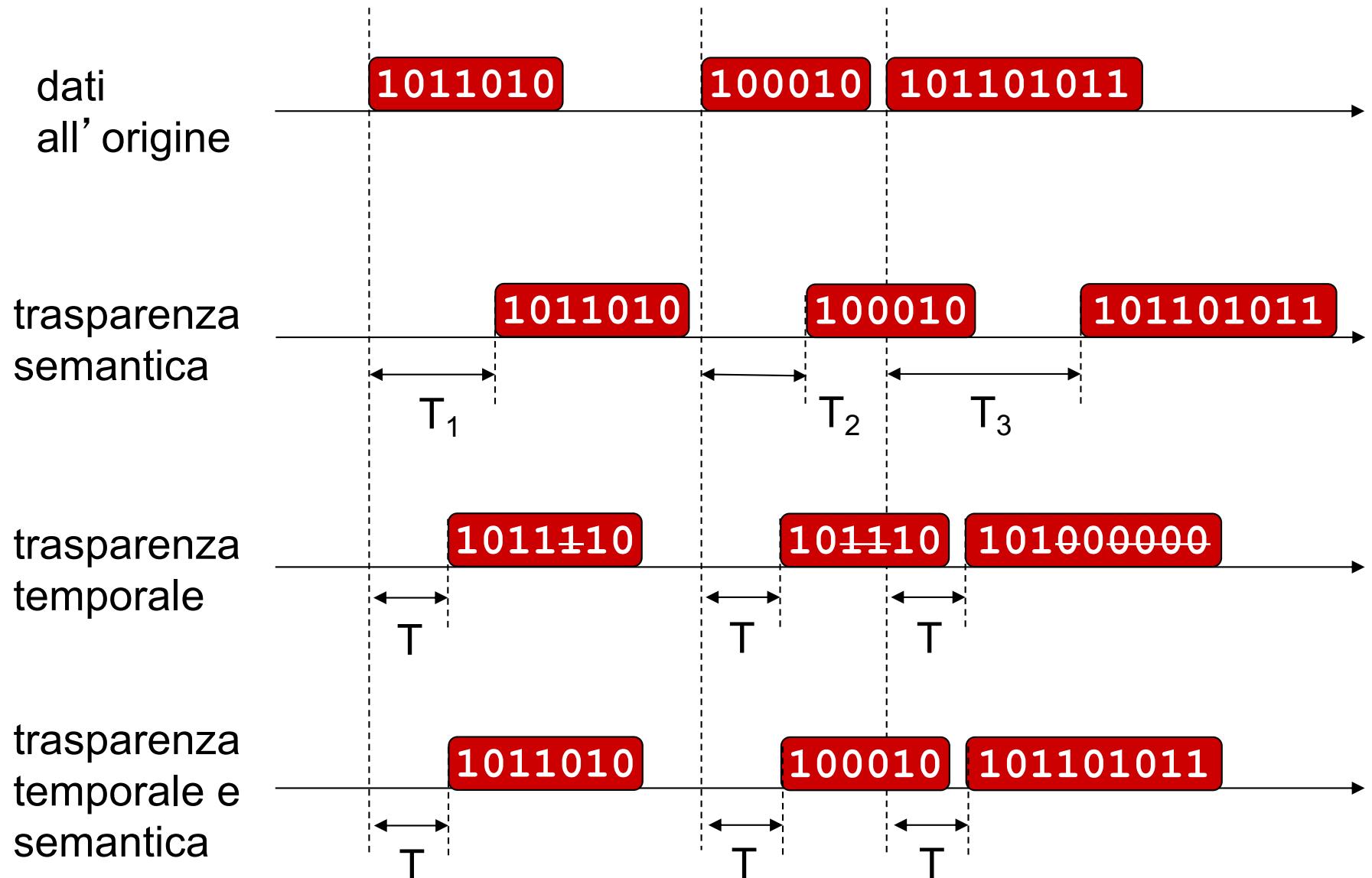


La qualità di servizio

- Problema di **trasparenza**
 - La rete può modificare l' informazione pertinente ad un dialogo
- Quality of service o QoS = qualità della comunicazione percepita dall' utente del servizio
 - È funzione delle caratteristiche di trasparenza della rete
- Trasparenza semantica
 - riguarda l' integrità delle informazioni trasportate
 - richiede di attuare procedure di recupero da situazioni di errore che possono insorgere nella rete
- Trasparenza temporale
 - riguarda la variabilità dei ritardi di transito
 - Un minimo ritardo di transito è sempre presente a causa del ritardo di propagazione



Trasparenza semantica e temporale



Indicatori di QoS

- Forniscono una misura “sintetica” della QoS
 - Probabilità di errore o perdita delle informazioni
 - Ritardo nella consegna delle informazioni
 - Variazioni del ritardo nella consegna delle informazioni (jitter)
 - Uniformità delle prestazioni (fairness)
- Applicazioni **non real-time**
 - Trasparenza semantica → bassa probabilità di errore
- Applicazioni **real-time**
 - Trasparenza temporale → basso ritardo e jitter
 - I servizi che richiedono la trasparenza temporale per la corretta interpretazione dell’informazione sono detti **isocroni**
 - Tipicamente quelli che si ottengono per conversione A/D, ad esempio il servizio telefonico
 - Per certi servizi è anche importante il valore di picco del ritardo di transito
 - Tale valore deve essere tanto più basso quanto maggiori sono le esigenze di interattività della comunicazione



Requisiti di QoS per diversi servizi

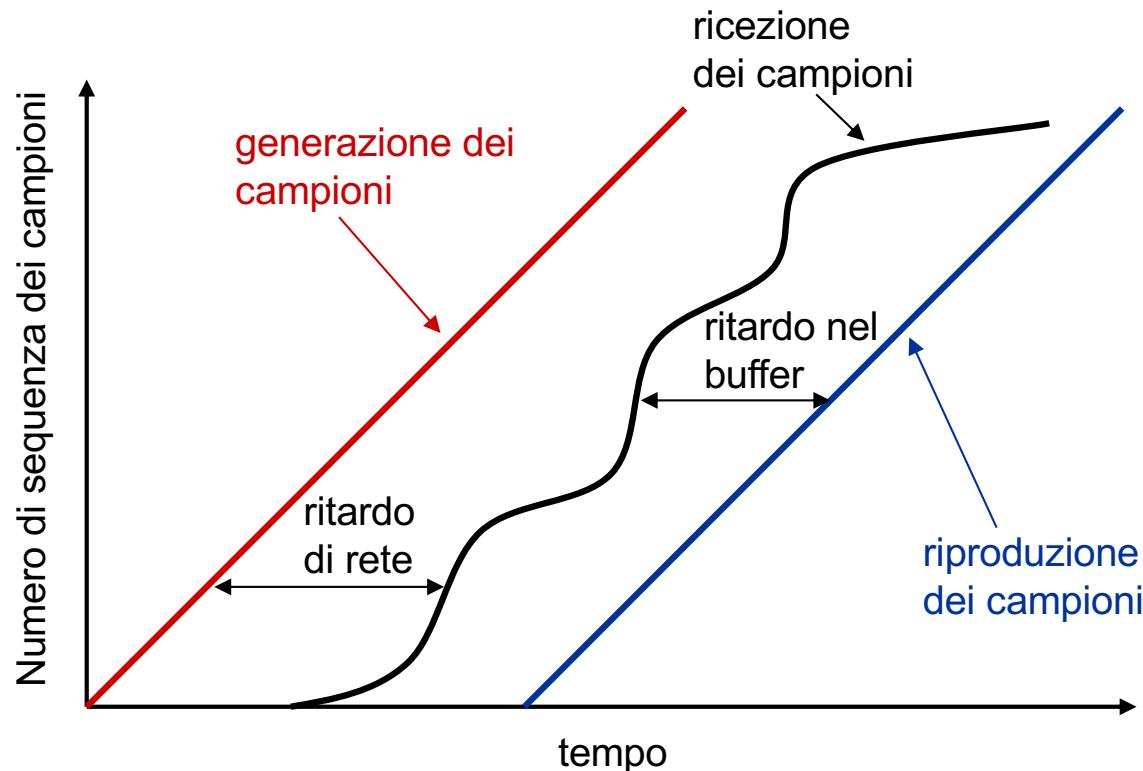
Application	Reliability	Delay	Jitter	Bandwidth
E-mail	High	Low	Low	Low
File transfer	High	Low	Low	Medium
Web access	High	Medium	Low	Medium
Remote login	High	Medium	Medium	Low
Audio on demand	Low	Low	High	Medium
Video on demand	Low	Low	High	High
Telephony	Low	High	High	Low
Videoconferencing	Low	High	High	High

Da A.S. Tanenbaum, “Computer Networks”



Compensazione al terminale

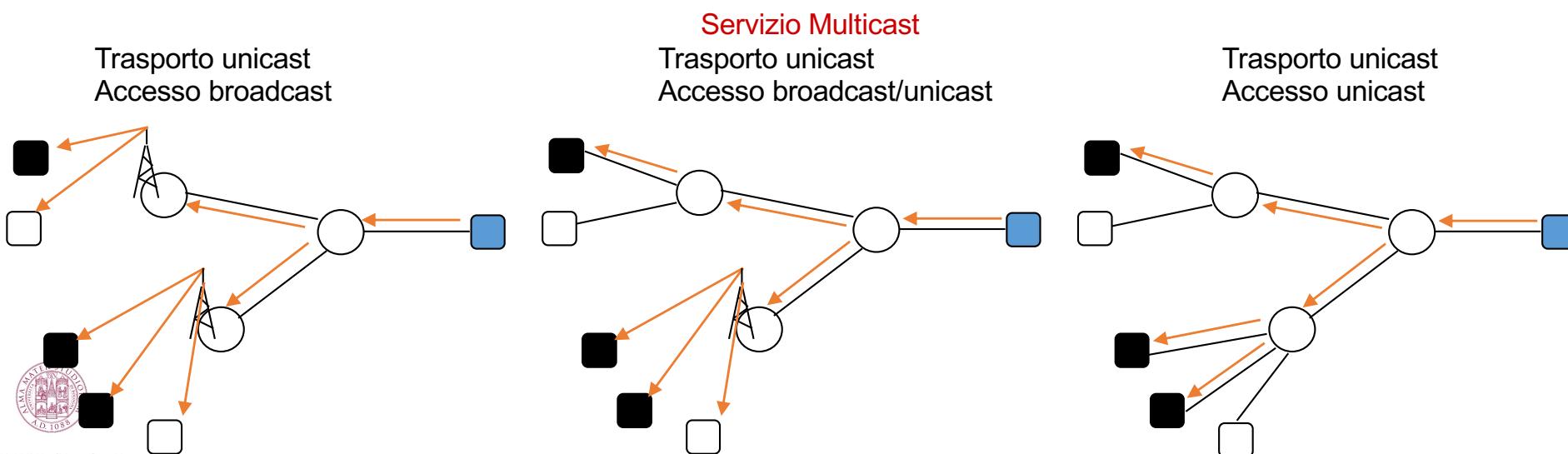
- Un terminale per la trasmissione di un segnale audio digitale genera dati ottenuti da una conversione A/D (campioni)
- Al ricevitore i dati devono essere consegnati con la tempistica appropriata per una corretta conversione D/A
- Al ricevitore si memorizzano i campioni in una memoria tampone (**playback buffer**) in modo da garantire la conversione con la corretta tempistica
 - ad ogni campione viene aggiunto un ritardo di playback variabile per compensare il ritardo (aleatorio) introdotto dalla rete



Servizio e canale

- Non esiste diretta corrispondenza fra tipologia di canale e tipologia di servizio
- Ad esempio lo stesso *servizio multicast* può essere implementato con
 - Canali broadcast
 - Canali punto-punto
 - Architettura mista

Es: Si vuole inviare informazione ai terminali di colore verde



Passato, presente, futuro

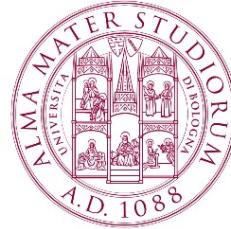
- Le reti si sono evolute in base al servizio
 - Ogni servizio aveva la propria rete dedicata e separata dalle altre
- Dai primi anni '80 viene introdotta la digitalizzazione:
 - Le tecnologie di rete cambiano ma inizialmente i servizi rimangono inalterati
- Il XXI secolo apre alla possibilità di **integrazione**
 - Servizi diversi sulla stessa rete oppure stesso servizio su diverse reti
- All'incirca dal 2010 lo smartphone diviene hub delle comunicazioni personali
 - Integrazione dei servizi nel terminale



Effetti sulla rete

- Servizi diversi = requisiti diversi
- Una rete totalmente integrata nei servizi dovrebbe essere estremamente flessibile
 - Allocazione
 - Distribuzione delle risorse in conformità alle necessità del servizio
 - Gestione
 - Le richieste dei vari servizi non devono interferire (o devono farlo il meno possibile)
- *Entrambi questi problemi sono ancora parzialmente irrisolti*





ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Multiplazione, codifica e QoS

Franco CALLEGATI



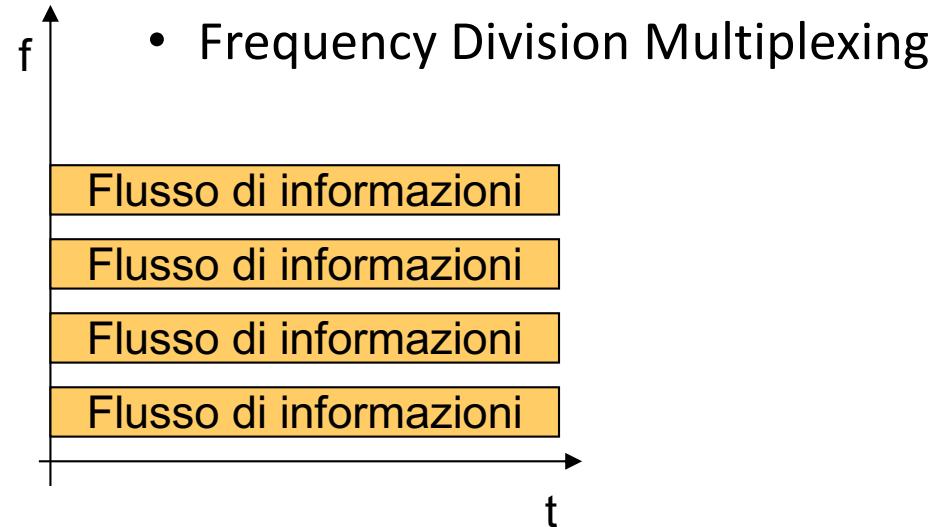
Mezzi trasmittivi e canali: multiplazione

- Più canali sono trasportati dallo stesso mezzo di trasmissione (collegamento)
- Si può realizzare utilizzando
 - Tempo
 - Frequenza
 - Codice
 - Spazio
- Dal punto di vista teorico ed in condizioni ideali queste modalità sono equivalenti
 - Differiscono per modalità di implementazione
 - La tecnologia di implementazione rende più o meno conveniente una soluzione rispetto alle altre

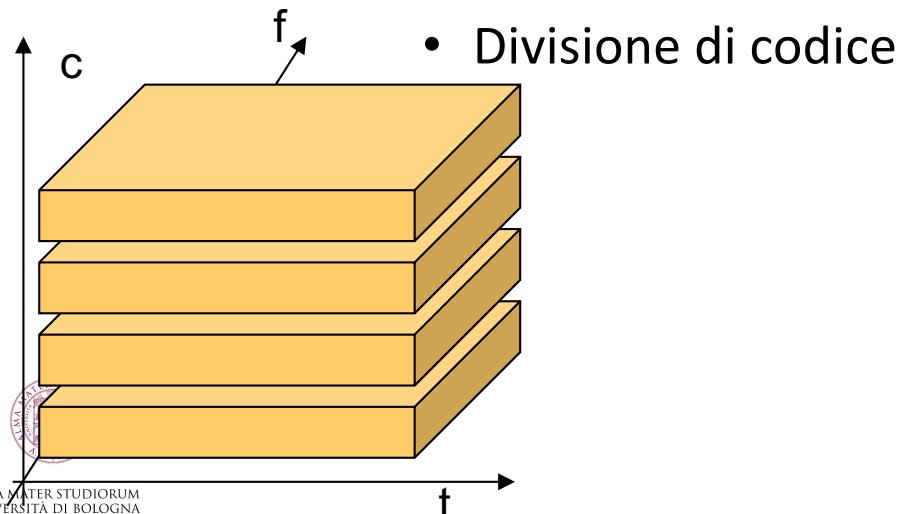


Alternative

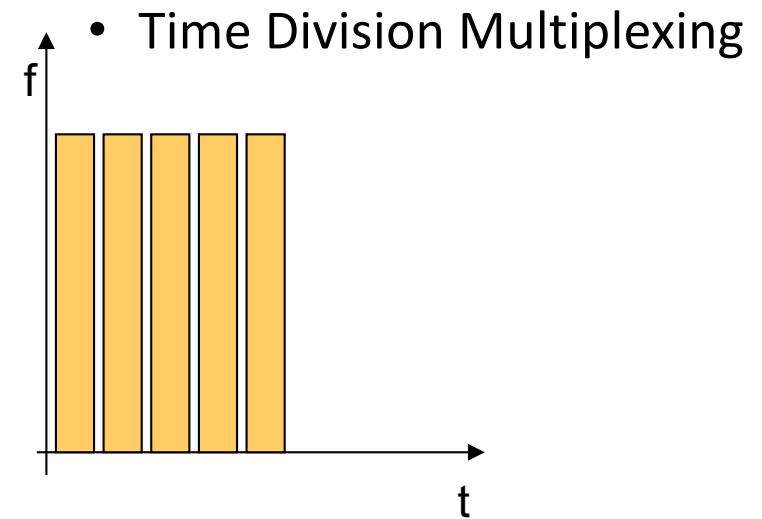
- FDM



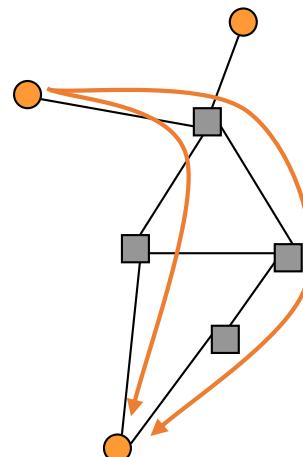
- CDM



- TDM



- Divisione di spazio



La multiplazione a divisione di tempo

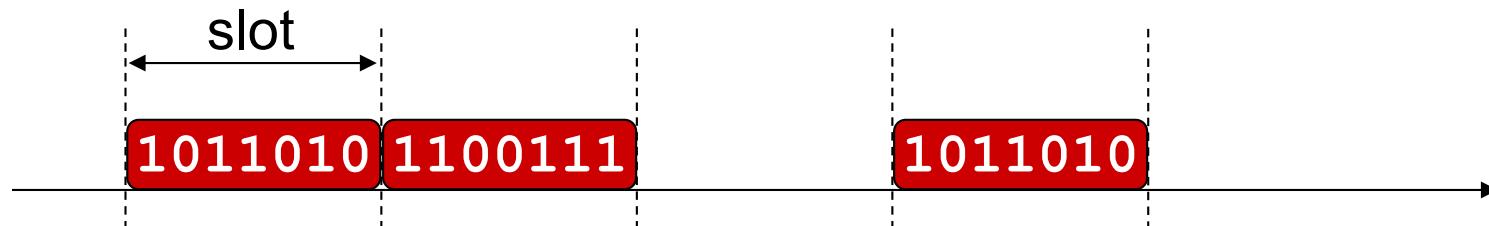
- Nelle reti numeriche viene principalmente utilizzata la multiplazione a divisione di tempo
 - Time division Multiplexing o TDM

multiplazione a divisione di tempo	
slotted	unslotted
framed	unframed
assegnazione statica della banda	assegnazione dinamica della banda



TDM slotted/unslotted

- TDM **slotted**
 - l'asse dei tempi è suddiviso in intervalli di durata prefissa (slot)
 - le unità informative hanno tutte la stessa lunghezza commisurata al singolo slot



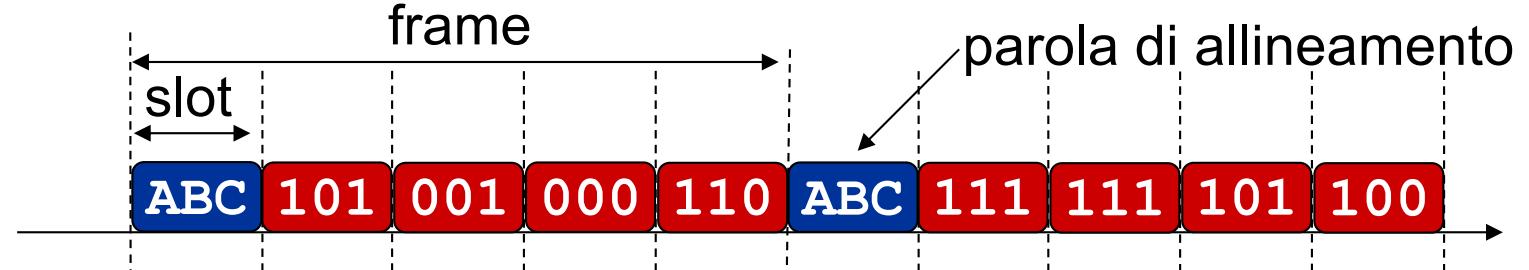
- TDM **unslotted**
 - l'asse dei tempi non è suddiviso a priori
 - si possono adottare unità informative di lunghezza variabile
 - è necessario un sistema esplicito di delimitazione delle unità informative



TDM slotted: framed/unframed

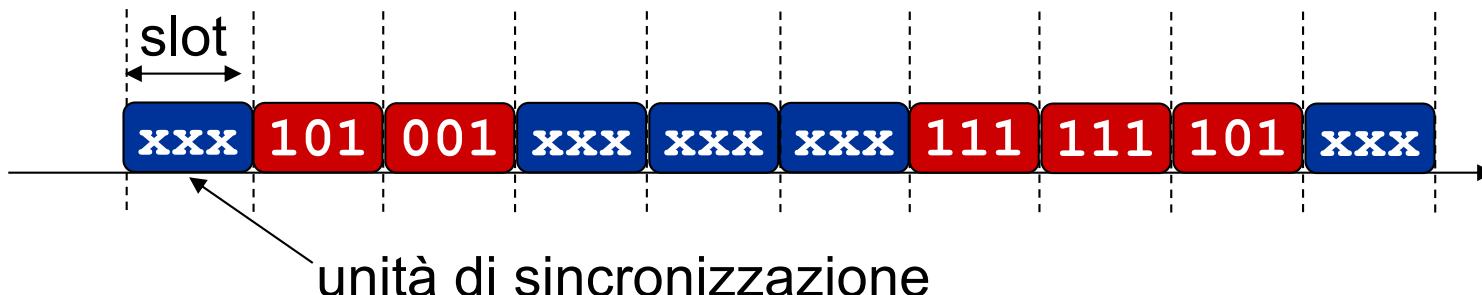
- TDM slotted **framed**

- gli slot vengono strutturati in trame (frame)
 - Si sincronizza la trama
 - non è necessaria la sincronizzazione a livello del singolo slot)



- In uno schema di multiplazione TDM slotted **unframed**

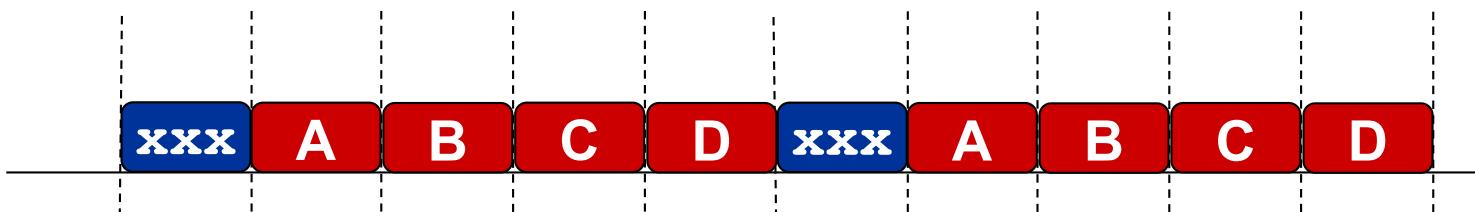
- gli slot si susseguono senza una struttura predefinita
 - occorre un sistema di sincronizzazione di slot



Assegnazione della banda

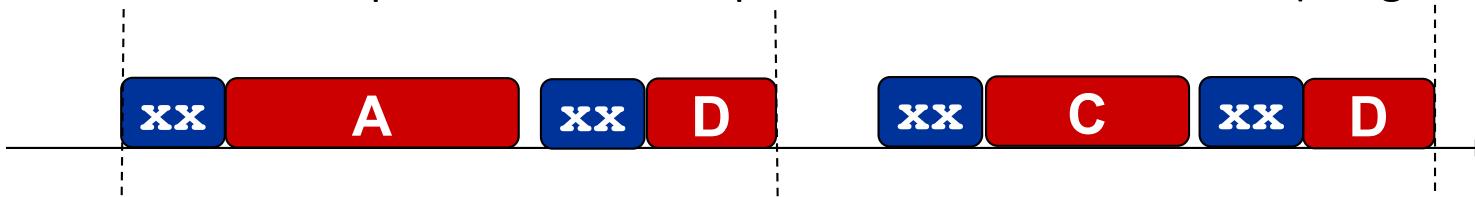
- Assegnazione statica

- Flusso informativo = banda dedicata (bit/sec)
- La banda non può cambiare a comunicazione in corso
- La richiesta complessiva di banda è ben controllabile se si controlla il numero di flussi attivi

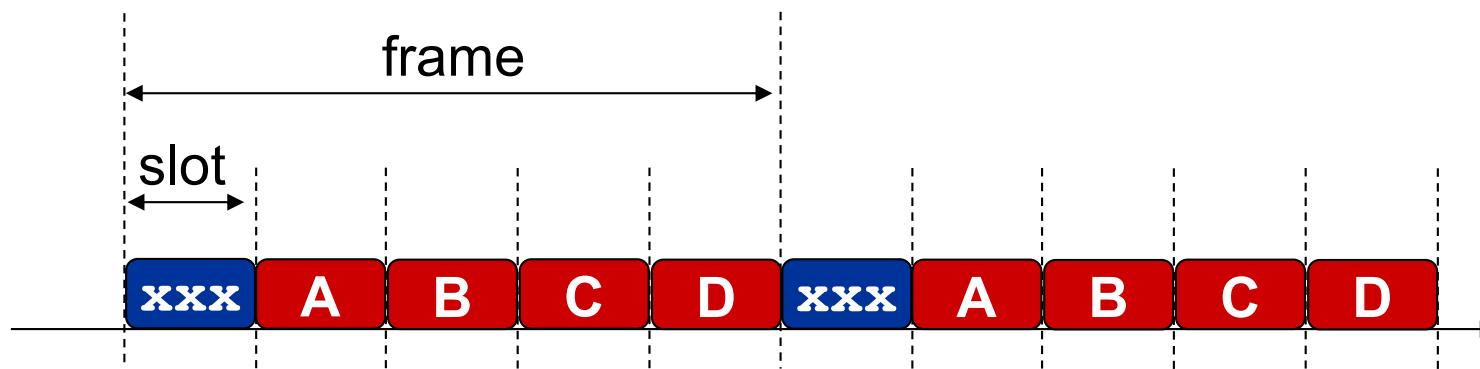


- Assegnazione dinamica

- Molti flussi informativi condividono liberamente la banda in base alle necessità
- La banda può cambiare a comunicazione in corso
- La richiesta complessiva di banda può diventare intollerabile (congestione)



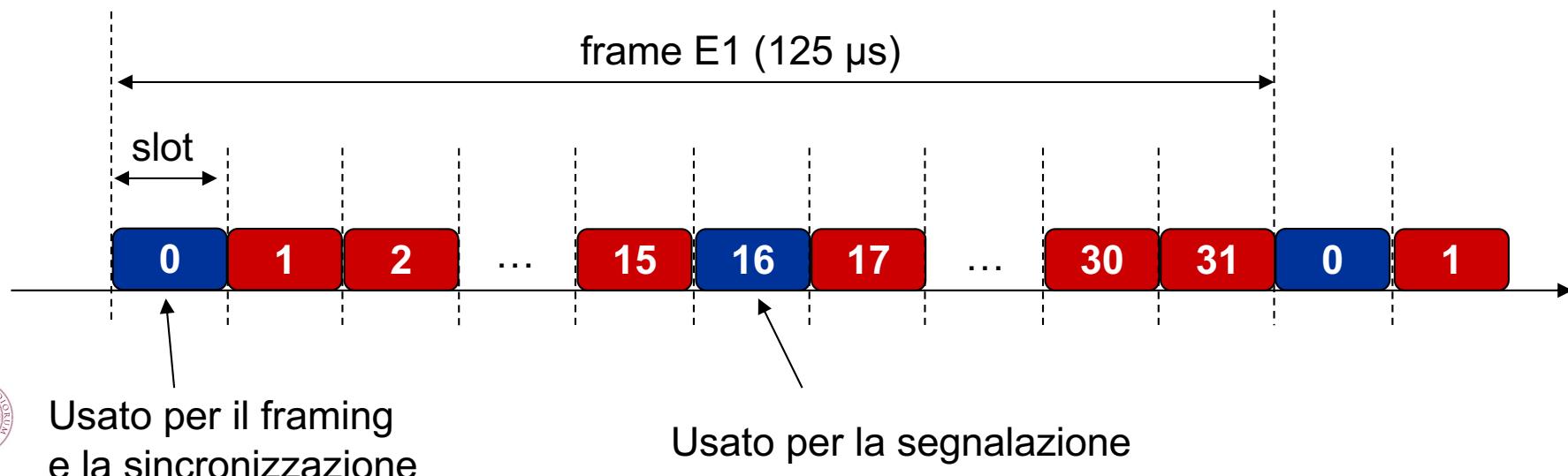
S-TDM





Sistema TDM/PCM plesiocrono

- Pulse Code Modulation (PCM) è utilizzato nella rete telefonica
- Singolo canale telefonico
 - Banda 300 Hz - 4 KHz
 - Campionamento
 - Ogni 125 us (8000 campioni/sec)
 - Quantizzazione a 8 bit
 - Risultano $8 \times 8000 = 64000$ bit/s
- Multiplazione S-TDM secondo una gerarchia predeterminata
- Framing ITU-T di primo livello (E1)
 - multipla 30 canali tributari + 2 di sincronizzazione e segnalazione
 - $64 \text{ kbps} \times 32 = 2.048 \text{ Mbps}$



Assegnazione dinamica della banda

- L'**assegnazione dinamica della banda** può essere effettuata sia nel caso slotted che unslotted
 - non c' è un' allocazione fissa di sottocanali alle chiamate
 - la capacità del canale può essere vista come una risorsa condivisa a cui si accede mediante una procedura di controllo
 - possono insorgere situazioni di contesa per l' utilizzo del canale
 - la presenza di unità informative di una stessa chiamata sul canale non è più necessariamente periodica
- **A-TDM = Asynchronous Time Division Multiplexing**
- In questo caso occorre definire:
 - le modalità di **assegnazione della banda**
 - le modalità di **gestione delle situazioni di contesa**





ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

La rete: funzioni di rete e commutazione

Franco CALLEGATI



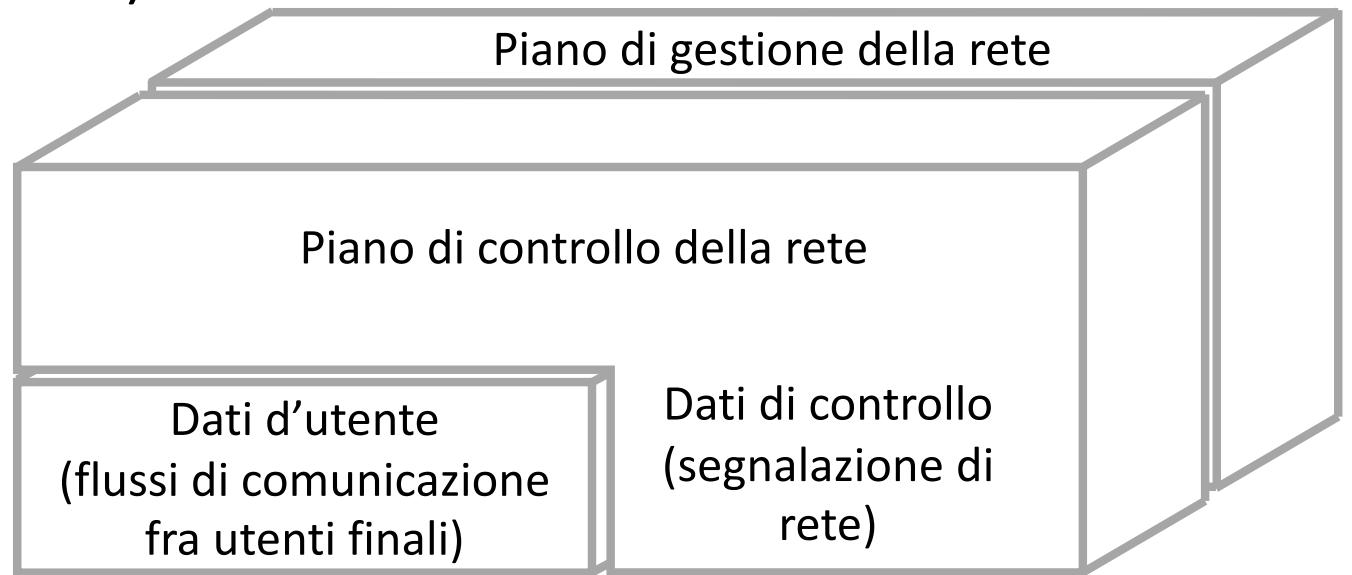
La rete

- In generale qual è l'obiettivo della rete?
 - Consentire una comunicazione
 - Tra una combinazione qualunque di elementi terminali
 - Con un livello accettabile di qualità di servizio
- Combinazione qualunque di terminali
 - Riconfigurazione dinamica dell'infrastruttura
- QoS accettabile
 - Assegnazione delle risorse
 - Controllo del canale di comunicazione



L'ipercubo della rete

- Le comunicazioni fra utenti costituiscono solamente una parte delle informazioni che viaggiano in rete
 - Si deve garantire il corretto comportamento della rete
 - Si devono gestire riconfigurazioni e malfunzionamenti
 - Si devono gestire gli aspetti economici legati alla rete (tariffazione ecc.)



Funzioni di rete

- **Trasmissione**
 - Trasferimento fisico del segnale da punto a punto o da un punto a molti punti
- **Commutazione**
 - Instradamento delle informazioni all' interno della rete (nodi e collegamenti) al fine di permettere la comunicazioni fra punti terminali per soddisfare le richieste degli utenti.
- **Segnalazione**
 - Scambio delle informazioni necessarie per la gestione della comunicazione e della rete
 - Segnalazione utente e rete
 - Segnalazione interna alla rete
- **Gestione**
 - Tutto ciò che concerne il mantenimento delle funzioni della rete; riconfigurazione di fronte ai guasti o cambiamenti strutturali, allacciamento di nuovi utenti ecc.



01/03/23

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA



Tecniche di commutazione

- Esistono numerose tecniche per realizzare la funzione di commutazione
- Le due tecniche tradizionalmente utilizzate e tuttora più diffuse sono
 - **Commutazione di circuito** (rete telefonica)
 - Informazione analogica o digitale
 - **Commutazione di messaggio** (rete telegrafica) o **di pacchetto** (reti di calcolatori)
 - Informazione digitale



01/03/23

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA



Commutazione di circuito

- La rete crea un canale di comunicazione dedicato fra due terminali che vogliono colloquiare
 - Circuito di comunicazione
- *Il circuito è riservato ad uso esclusivo* dei terminali chiamante e chiamato
- Esiste quindi un *ritardo iniziale* dovuto al tempo necessario per *instaurare il circuito (call set-up time)*
 - dopo di ciò la rete è trasparente (è garantita la trasparenza temporale) per l'utente ed equivale ad un collegamento fisico diretto



01/03/23

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA



Fasi della comunicazione

- Instaurazione del circuito
 - Prima che i segnali di utente possano essere trasmessi la rete deve instaurare un circuito fra terminale chiamante e terminale chiamato (circuito *end-to-end*).
 - Richiede un'opportuna segnalazione.
- Dialogo
 - I due terminali si scambiano informazioni utilizzando il circuito dedicato.
- Disconnessione del circuito
 - Al termine del dialogo il circuito deve essere rilasciato, al fine di poter essere utilizzato per altre chiamate.



01/03/23

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA



Pro e contro

- Pro
 - il circuito è dedicato e garantisce sicurezza ed affidabilità
 - è garantita la trasparenza temporale
 - le procedure di controllo sono limitate ad inizio e fine chiamata
- Contro
 - se le sorgenti hanno un basso tasso di attività il circuito è sottoutilizzato,
 - la capacità del canale è fissata dalla capacità del circuito e non si può variare.



01/03/23

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA



Commutazione di messaggio o pacchetto

- Trasporta informazioni in forma numerica
- Le informazioni di utente sono strutturate in messaggi unitamente ad opportune informazioni di segnalazione
 - indirizzamento, verifica della correttezza delle informazioni, ecc.
- Comutazione di Pacchetto:
 - i messaggi vengono suddivisi in sotto-blocchi di lunghezza massima prefissata detti pacchetti, per motivi
 - Di linea: evitare frammenti troppo lunghi in relazione al rumore
 - Di rete: limitare i tempi medi di attesa nei nodi
- I messaggi o i pacchetti vengono trasmessi da un nodo di commutazione all'altro utilizzando in tempi diversi risorse comuni



01/03/23

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA



Tecniche di commutazione di pacchetto

- Circuito virtuale (**connection oriented**)
 - scambio delle informazioni preceduto da una procedura di segnalazione (apertura) durante la quale viene stabilito il percorso dei pacchetti da origine a destinazione
 - a tale percorso viene associato un numero di ***circuito virtuale***
 - tutti i pacchetti di dati seguono lo stesso percorso,
 - contengono solamente il numero di circuito virtuale.
- Datagramma (**connectionless**)
 - ogni pacchetto viene gestito e instradato in modo indipendente, senza relazione con pacchetti precedenti o successivi, anche appartenenti alla stessa connessione
 - ogni pacchetto porta tutte le informazioni di indirizzamento necessarie per raggiungere la destinazione finale
 - pacchetti diversi di una stessa connessione possono seguire percorsi diversi e quindi avere tempi di percorrenza diversi



01/03/23

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA



Pro e contro

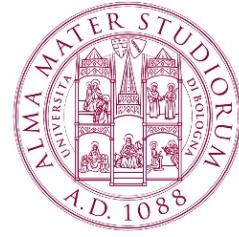
- Pro
 - L'efficienza nell'utilizzazione dei collegamenti è maggiore, poiché la stessa linea è condivisa in modo dinamico da più chiamate
 - La rete può supportare diverse velocità ed effettuare anche conversioni tramite memorizzazione.
 - È facile implementare meccanismi per il controllo dell' errore (trasparenza semantica)
- Contro
 - È difficile garantire un predeterminato tempo di transito, quindi è poco adatta per servizi di tipo real time



01/03/23

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA





ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Le reti a pacchetto: architettura

Franco CALLEGATI



Sistemi chiusi

- Tutte le reti di calcolatori della prima generazione nascono e si evolvono come **sistemi chiusi**
 - nel mondo dell'Informatica
 - tutti i componenti della rete devono essere dello stesso costruttore (**captivity**),
 - nel mondo delle Telecomunicazioni
 - una rete specializzata per ogni servizio.
- Questo crea **incompatibilità**, ponendo ostacoli alla comunicazione:
 - a causa della diversità delle reti gli apparati non riescono ad interpretare i segnali degli altri,
 - anche se i calcolatori riescono a connettersi non riescono a colloquiare perché parlano linguaggi diversi,
 - i programmi applicativi non riescono ad operare in ambiente distribuito.



ISO-OSI

- A partire dal 1976 la ISO ha dato il via a lavori per giungere ad una serie di standard unificati per la realizzazione di **reti di calcolatori aperte**.
- La ISO ha per prima cosa proposto un modello di riferimento
 - **Open System Interconnection Reference Model (OSI-RM)**
 - È diventato standard internazionale nel 1983 (ISO 7498).
 - È basato sul concetto centrale di una **architettura a strati**.
- L'architettura a strati ha alcuni grandi vantaggi:
 - scomponete il problema in **sottoproblemi** più semplici da trattare,
 - rende i vari livelli **indipendenti**,
 - definendo solamente **servizi e interfacce**, livelli diversi possono essere sviluppati da enti diversi.



Sistemi aperti

- Obiettivo:
 - Realizzare una rete di calcolatori in cui **qualsiasi** terminale comunica con **qualsiasi** fornitore di servizi mediante **qualsiasi** rete.
- Per realizzare un sistema aperto è necessario stabilire delle regole comuni :
 - Sono necessari degli **standard**
- Tutte le soluzioni proposte hanno in comune **un'architettura a strati**



Cosa definisce?

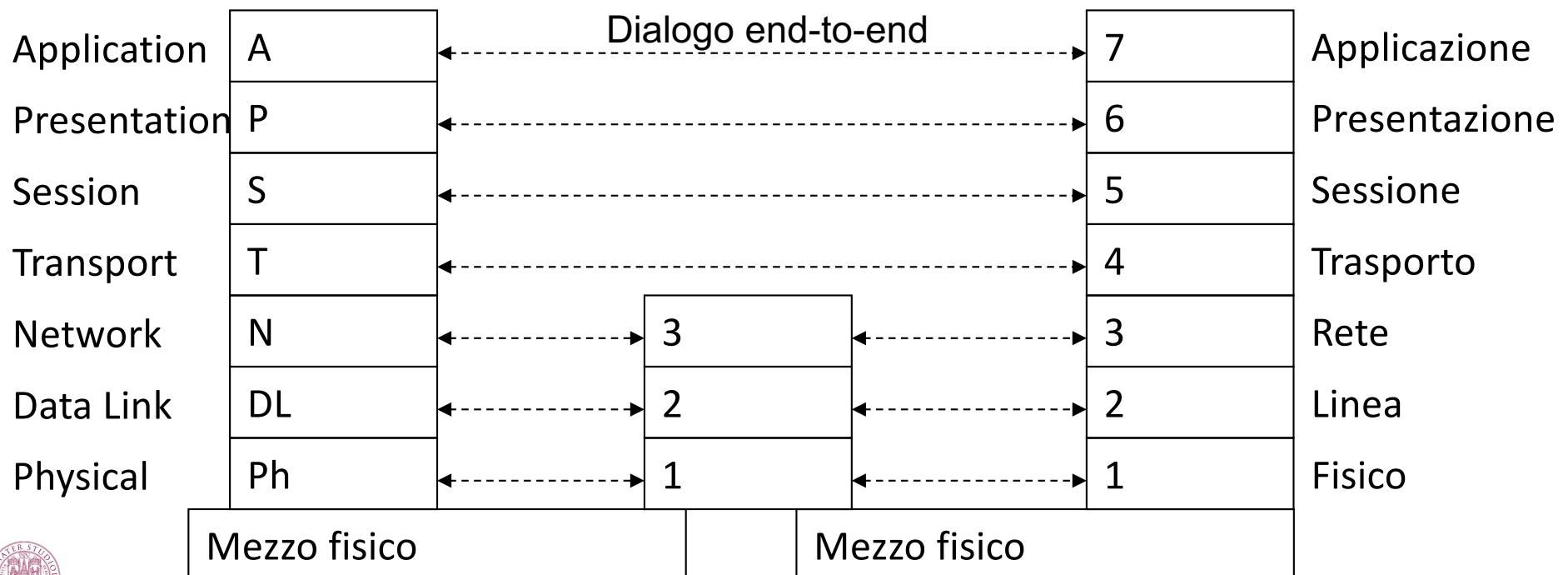
Le definizioni contenute nell' OSI coinvolgono tre livelli di astrazione:

- **Modello di riferimento:**
 - schema concettuale
 - numero degli strati coinvolti
 - definizione generale delle funzioni degli strati e delle relazioni fra di essi.
- **Definizione dei servizi:**
 - definizione astratta di *ciò che viene fornito* da uno strato.
- **Specifiche di protocolli ed interfacce:**
 - descrizione di *come viene fornito* un servizio da uno strato.



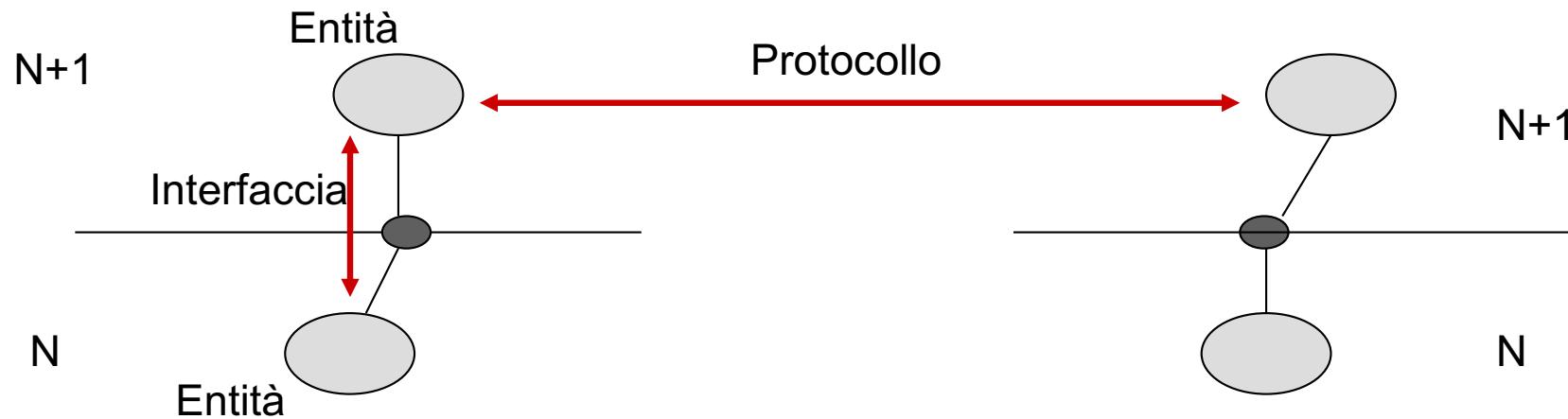
Modello di riferimento

- Architettura a **7 strati**, numerati dal basso verso l'alto da 1 a 7
 - gli strati 1, 2, 3 sono detti **lower o network oriented layers**
 - gli strati 5, 6, 7 sono detti **upper o application oriented layers**
 - Lo strato 4 funge da raccordo fra gli upper e lower layers
 - si possono avere funzione di ripetizione (**relay**) ai livelli 1, 2, 3, che si dice operano **link-by-link**
 - gli strati dal 4 in su operano solo **end-to-end**



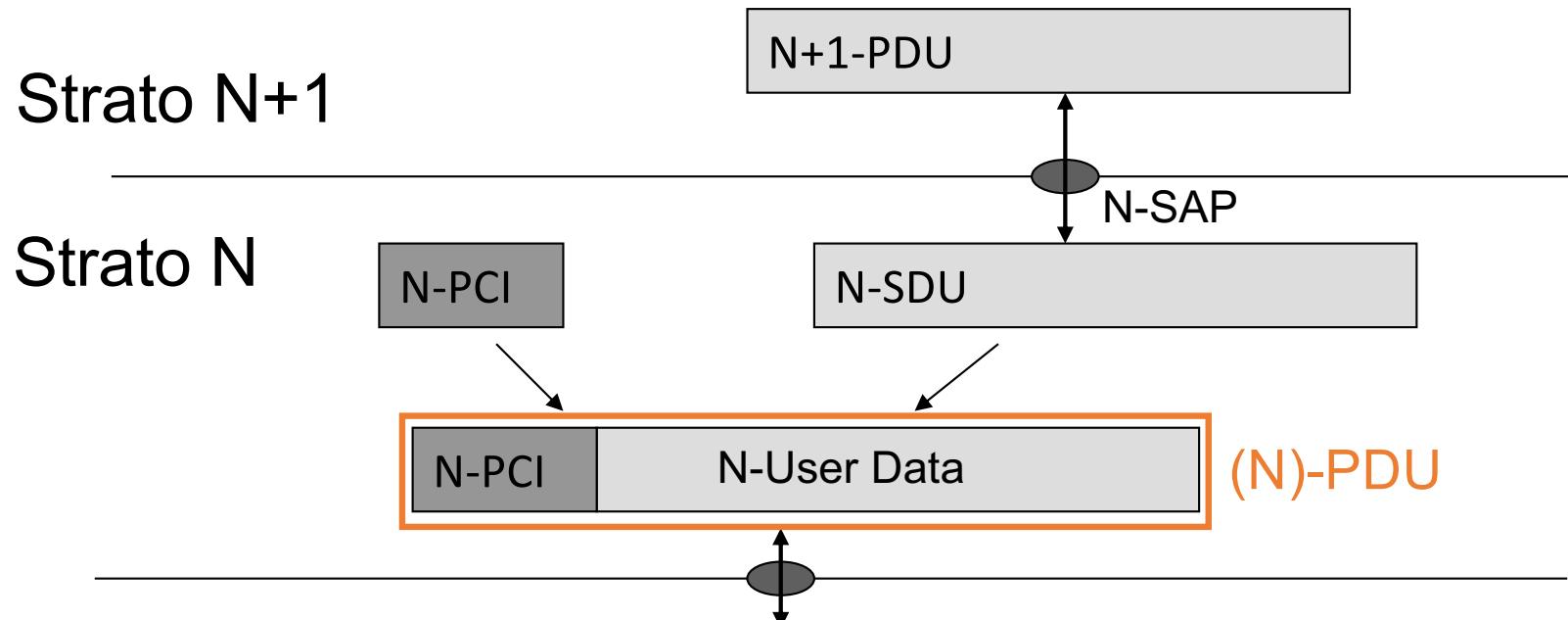
Entità, interfacce, protocolli

- **Entità** ogni elemento attivo in uno strato, identificata da un *nome simbolico (title)*
 - Nello strato N-esimo possono essere attive una o più entità
- **Protocollo**: regole di dialogo fra entità dello stesso livello
- **Interfaccia**: regole di dialogo fra entità di livelli adiacenti



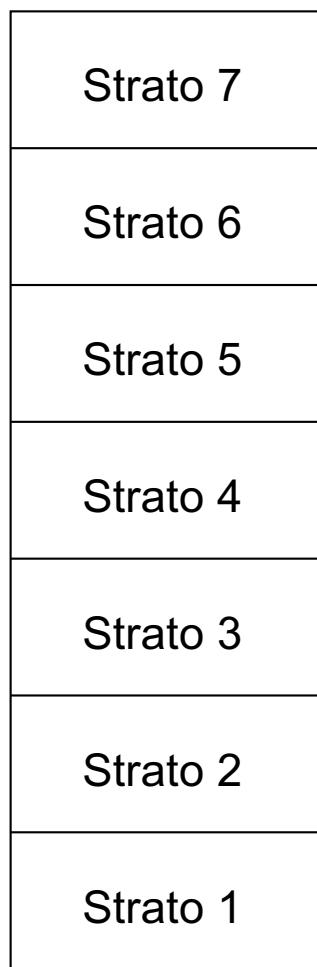
Trasferimento dei dati

- N-Protocol Data Unit (**PDU**): dati trasferiti fra entità di strato N
- N-Service Data Unit (**SDU**): dati passati allo strato N dallo strato N+1
- N-Service Access Point (**SAP**): indirizzo di identificazione del flusso dati fra N+1 ed N
- N-Protocol Control Information (**PCI**): informazioni aggiuntive per il controllo del dialogo a livello N
- **Encapsulation:** N-PDU = N-PCI+ N-SDU



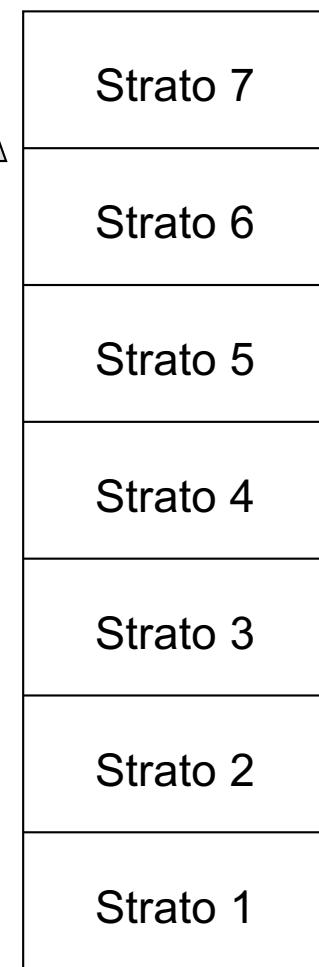
Flusso informativo

Parte trasmittente



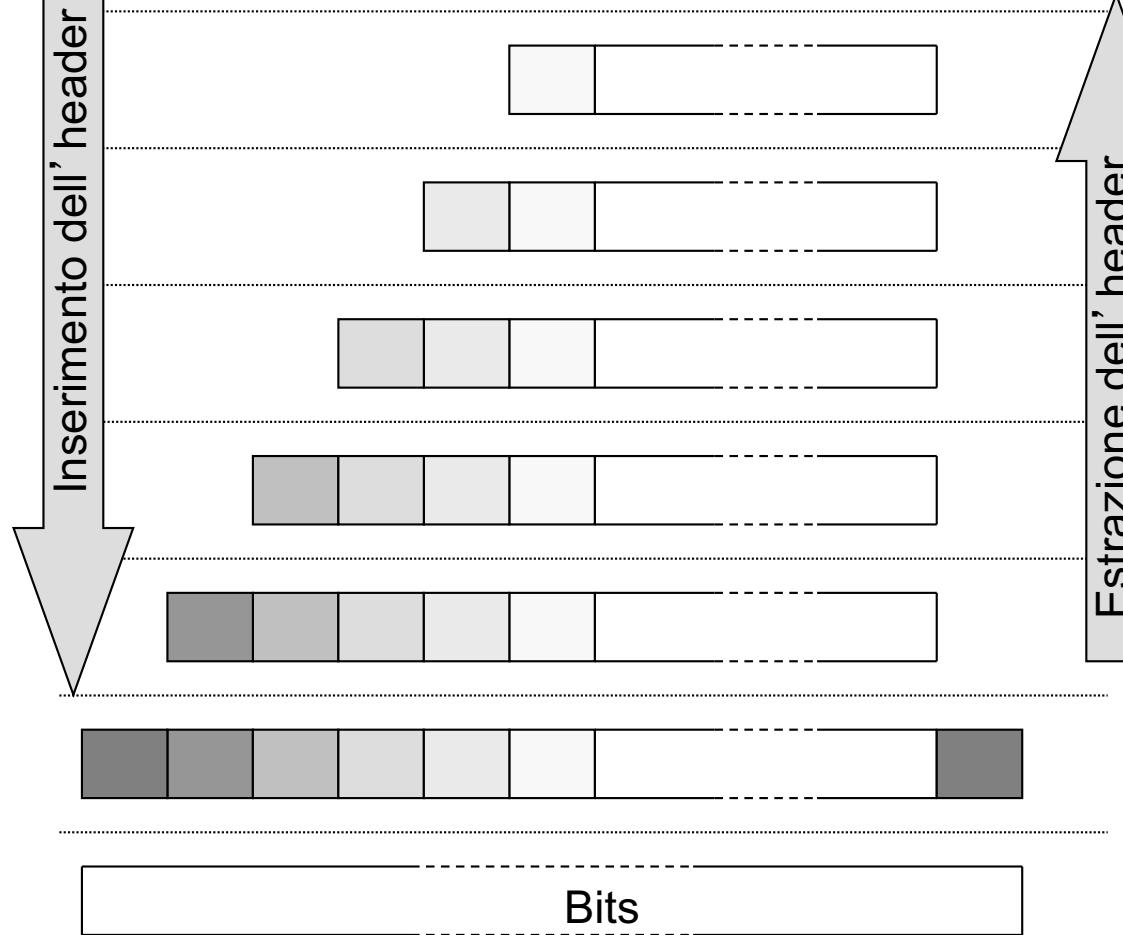
Dati d' utente

Parte ricevente



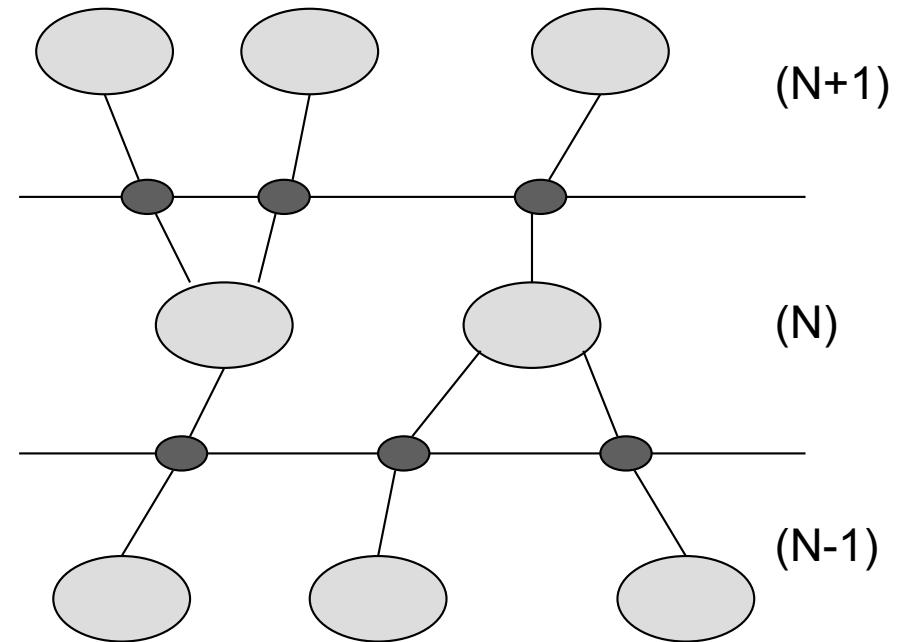
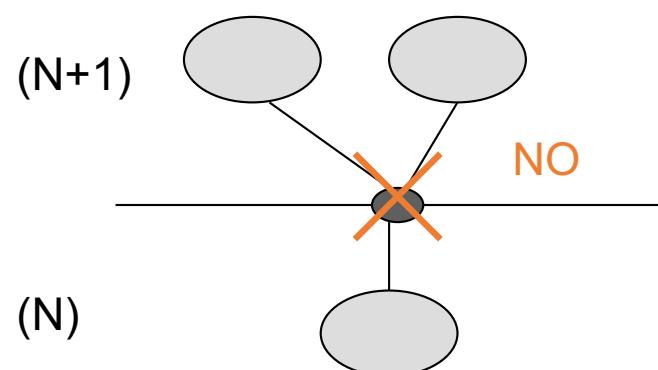
Inserimento dell' header

Estrazione dell' header



Uso dei SAP

- Un' entità di strato N può servire più (N)-SAP contemporaneamente.
- Un utilizzatore di strato N può servirsi di più (N)-SAP contemporaneamente



- Non è permesso connettere più (N)-user allo stesso (N)-SAP
 - Si genererebbe ambiguità sulla provenienza/destinazione dei dati
 - Ad ogni indirizzo deve essere univocamente associato un nome



Modalità di Servizio

- Una modalità di fornire un servizio si dice *Connection Oriented* quando si stabilisce una *connessione*:
 - Connessione = associazione logica fra due o più sistemi al fine di trasferire informazioni
 - Il processo di comunicazione si compone normalmente di tre fasi
 - *instaurazione* della connessione, tramite lo scambio di opportune informazioni iniziali,
 - *trasferimento* dei dati veri e propri,
 - *chiusura* della connessione.
- Qualora i dati vengano trasferiti senza prima stabilire una connessione si parla di servizio *Connectionless*
 - Per ogni accesso al servizio vengono fornite tutte le informazioni necessarie per il trasferimento dei dati
 - Ogni unità di dati viene trasferita in modo indipendente dalle altre



Modalità di dialogo

- **Confermato**

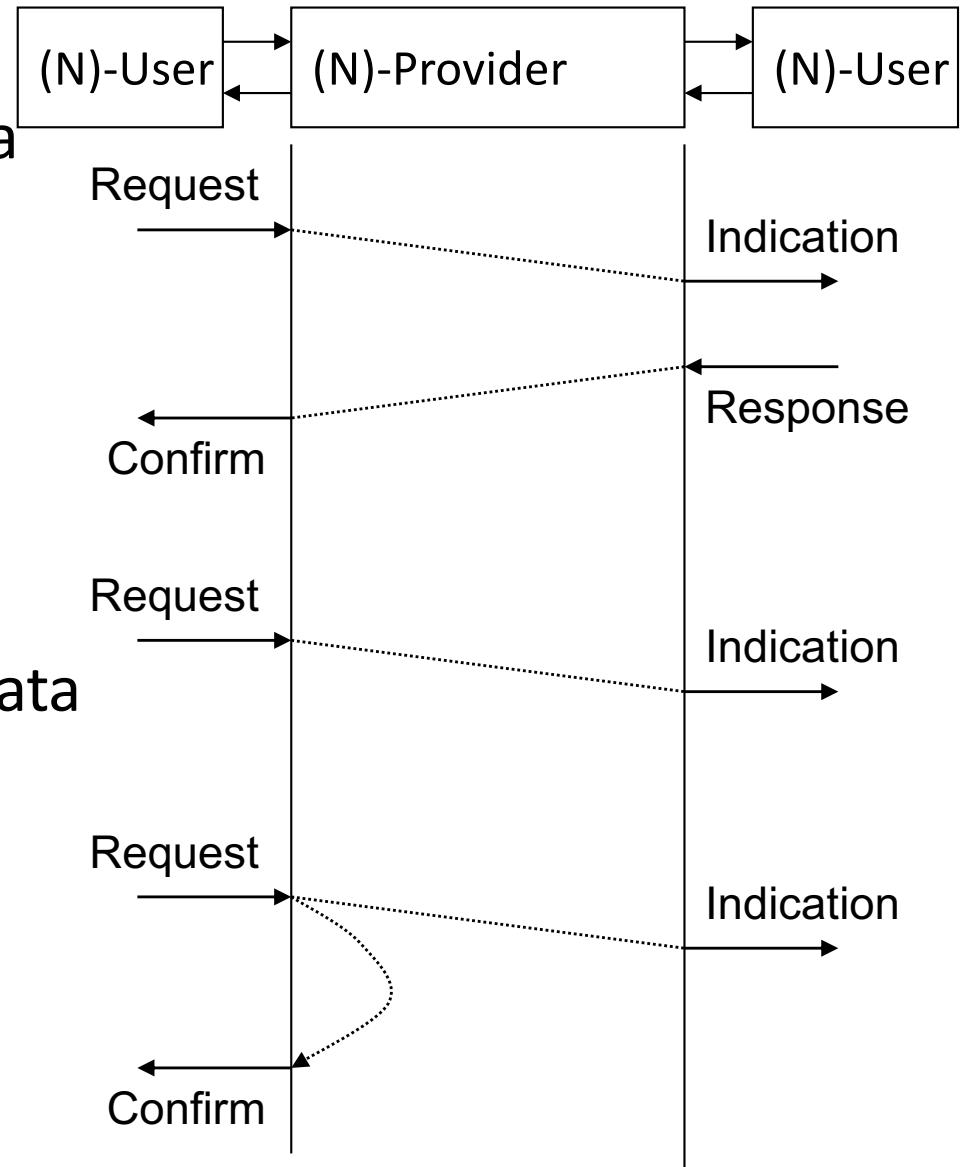
- Prevede esplicita conferma da parte del destinatario

- **Non confermato**

- Non prevede alcuna conferma

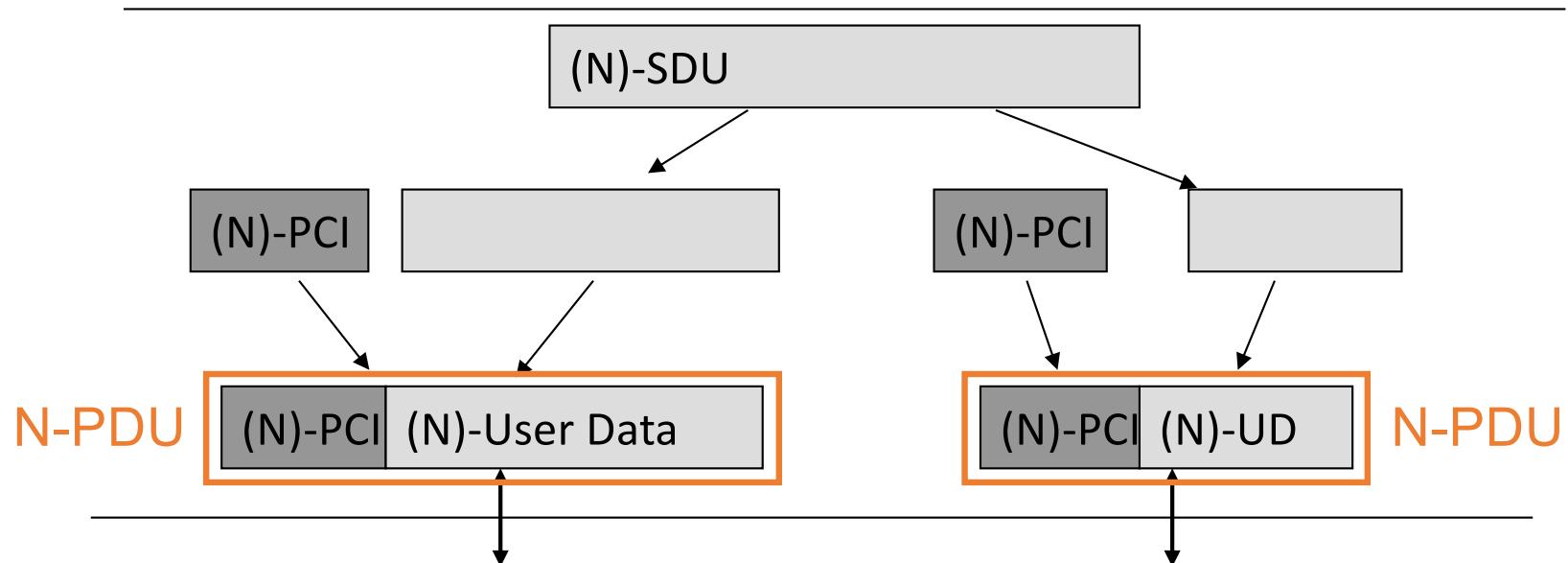
- **Parzialmente confermato**

- La richiesta viene confermata dal service-provider



Segmentazione e riassemblamento

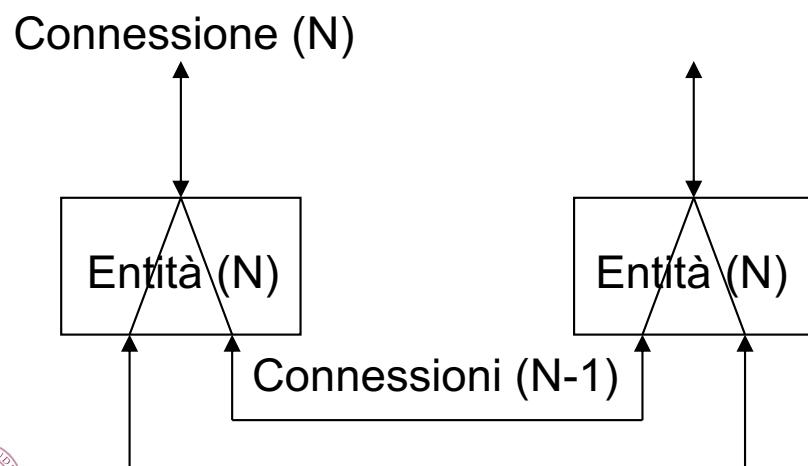
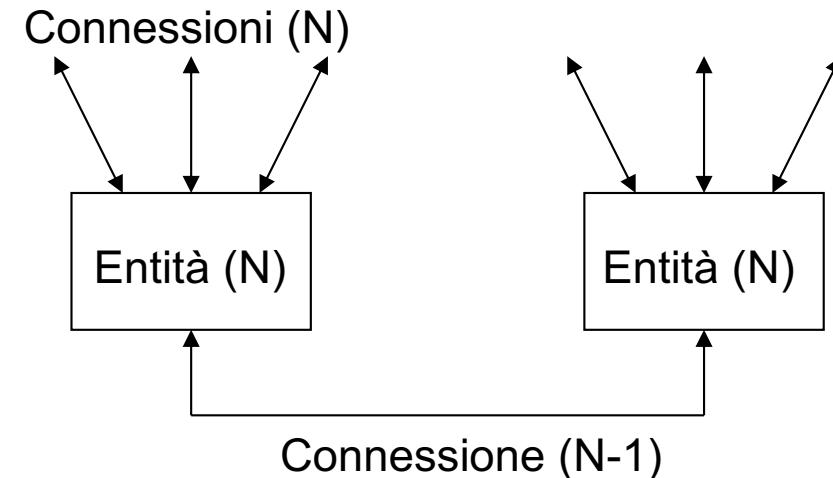
- E' possibile dividere il contenuto di una SDU in una o più PDU
 - La suddivisione si dice segmentazione e la ricostruzione si dice riassemblamento
 - (E' possibile anche accoppare più SDU in una PDU)
- Tipicamente la segmentazione serve per conformarsi a limitazioni sulla lunghezza massima dei messaggi



Multiplazione e Splitting

- **Multiplazione**

- più connessioni di strato N vengono mappate in una di strato N-1
- L' obiettivo è la condivisione delle risorse



- **Splitting**
- È duale alla multiplazione
- Aumenta la flessibilità e la velocità di trasferimento dei dati



Strato 1: fisico

- Scopo dello strato fisico è quello di attivare, mantenere e disattivare la connessione fisica fra due entità di strato 2.
- Specifica le modalità di invio dei **singoli bit** sul mezzo di fisico di trasmissione
- Per fare questo deve specificare le caratteristiche:
 - **meccaniche**:
 - forma di prese e spine, numero di contatti,
 - **elettriche**:
 - voltaggio e caratteristiche elettriche dei segnali associati all'interfaccia,
 - **funzionali**:
 - significato dei vari segnali,
 - **procedurali**:
 - combinazioni e sequenze dei segnali all'interfaccia necessarie al fine di regolarne il funzionamento.



Strato 2: linea

- Lo strato di linea deve
 - attivare, mantenere e disattivare la connessione fisica fra due entità di strato 3;
 - rendere **affidabile** il collegamento fra i nodi di rete
- Le funzioni tipicamente svolte dallo strato 2 sono le seguenti:
 - strutturazione del flusso di dati in unità di dialogo, denominati **trame** o **frames**;
 - controllo e gestione degli **errori di trasmissione**;
 - controllo di **flusso**;
 - controllo di **sequenza**.



Strato 3: rete

- Scopo dello strato di rete è di far giungere le unità di informazione, dette **pacchetti (packets)**, al destinatario *scegliendo la strada* attraverso la rete
- Si occupa dunque del problema della **commutazione**
 - Nelle reti di calcolatori si usa la **commutazione di pacchetto** e la funzione svolta dallo strato 3 viene detta **routing**
- Occorre un modo per individuare i destinatari: è necessario uno **schema di indirizzi**.
 - In una rete globale lo schema di indirizzi deve essere **universale**.
- Si sono sviluppate reti parziali, ora denominate **sottoreti** e per arrivare ad una rete unica occorre definire un protocollo di **interconnessione di reti**



Strato 4: trasporto

- Scopo dello strato di trasporto è *fornire un canale sicuro end-to-end*, svincolando gli strati superiori da tutti i problemi di rete
- Una tipica funzione è *adattare la dimensione* dei frammenti forniti dagli strati superiori (*files*) a quella richiesta dalle reti (*pacchetti*):
 - funzione di Pacchettizzazione (*fragmenting/reassembling*)
- Può avere molte altre funzioni fra cui
 - controllo dell' errore,
 - controllo di flusso,
 - gestione di dati prioritari, ecc..
- Non tutti le applicazioni hanno bisogno delle stesse funzioni,
 - Si possono definire diverse *Classi* di trasporto



Strato 5: sessione

- Suddivide il dialogo fra le applicazioni in **unità logiche** (dette appunto **sessioni**),
 - Una sessione deve essere identificata, eventualmente interrotta e ripresa per fare fronte a vari eventi catastrofici: perdita di dati, caduta della linea, momentaneo crash di uno dei due interlocutori...
- Permette la **chiusura ordinata** (soft) del dialogo
 - Garanzia che tutti i dati trasmessi siano arrivati a destinazione
- Per fare le sue funzione introduce dei **punti di sincronizzazione**
- Anche gli strati di sessione hanno molte funzionalità e possono essere più o meno completi a seconda delle richieste



Strato 6: presentazione

- Adatta il **formato (sintassi)** dei dati usato dagli interlocutori preservandone il **significato (semantica)**
- Ogni interlocutore ha una sua **Sintassi locale** e durante il dialogo bisogna concordare una **Sintassi di trasferimento**
- E' stato definito un linguaggio detto **Abstract Syntax Notation 1 (ASN 1)** per descrivere e negoziare le sintassi



Strato 7: applicazione

- Lo strato di Applicazione è **l'utente** della rete e pertanto non deve offrire servizi a nessuno
 - Rappresenta il **programma applicativo (Applicazione)** che per svolgere i suoi compiti ha bisogno di comunicare con altre applicazioni remote
- Le applicazioni non possono essere standardizzate completamente:
 - ISO ha cominciato a standardizzare dei moduli applicativi denominati Application Service Element (ASE) su richiesta di gruppi di utenti interessati



Trasporto e interconnessione

- Se si vuole una rete universale diffusa ed unica a livello mondiale
 - Lo strato di **trasporto** dove essere unico
 - Parte dello strato di rete (**internetworking**) dove essere unico
- OSI definisce **i protocolli** che devono essere adottati da tutti i computer per creare una rete aperta universale
 - Il Protocollo IP (ISO 8473)
 - Il Protocollo di Trasporto (ISO 8073)



ISO/OSI vs TCP/IP

OSI	TCP/IP	Protocolli
Application		HTTP, TELNET, FTP, SMTP, POP, DNS, SNMP
Presentation	Application	
Session		
Transport	Transport	TCP, UDP
Network	Network	IP, ICMP, IGMP, ARP, RARP
Data Link		ETHERNET, IEEE 802, HDLC, PPP
Physical	Link	





ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Internet

Franco CALLEGATI



Nascita di Internet

Bruce Sterling, autore di fantascienza, scrive sulla rivista "Fiction & Science fiction" nel febbraio '93:

Circa 30 anni fa, la RAND Corporation, una delle principali fucine di idee degli U.S.A. del periodo della guerra fredda, si trovò ad affrontare un singolare problema strategico. “Come avrebbero potuto comunicare le autorità americane dopo una guerra nucleare?” Gli U.S.A. Post guerra atomica avrebbero comunque avuto bisogno di una rete di comando e controllo, da città a città, da base a base, da stato a stato ... la soluzione proposta dalla RAND fu resa pubblica nel 1964.

In primo luogo la rete *non avrebbe dovuto avere alcuna autorità centrale*. Inoltre doveva essere progettata fin dal principio *per operare anche se a pezzi...*

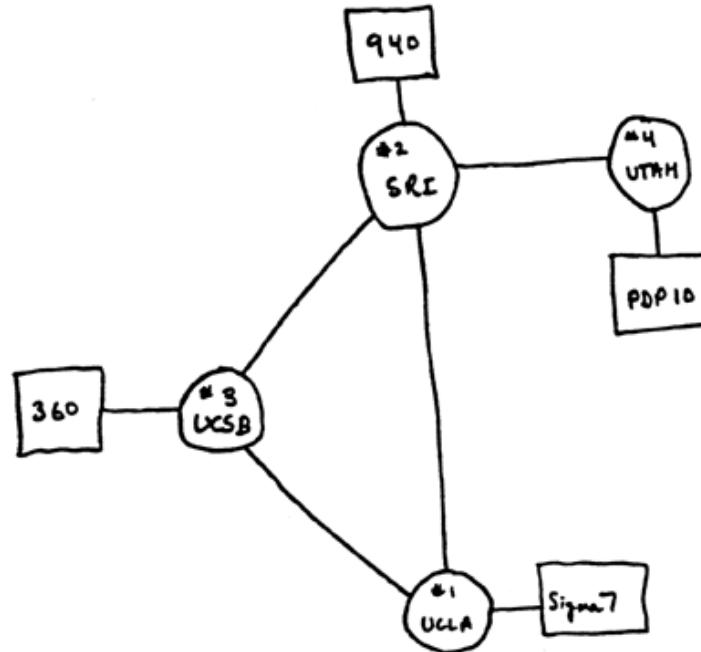


Il principio era semplice: si doveva assumere che la rete stessa potesse essere inaffidabile in qualunque momento.



ARPANet

- 1969: Il dipartimento della difesa USA (DoD) attraverso l' Agenzia per i Progetti di Ricerca Avanzati (ARPA), finanzia la sperimentazione di una rete di calcolatori (ARPANET) fra:
 - UCLA (University of California at Los Angeles)
 - Stanford Research Center
 - UCSB (University of California at Santa Barbara)
 - Università dello Utah

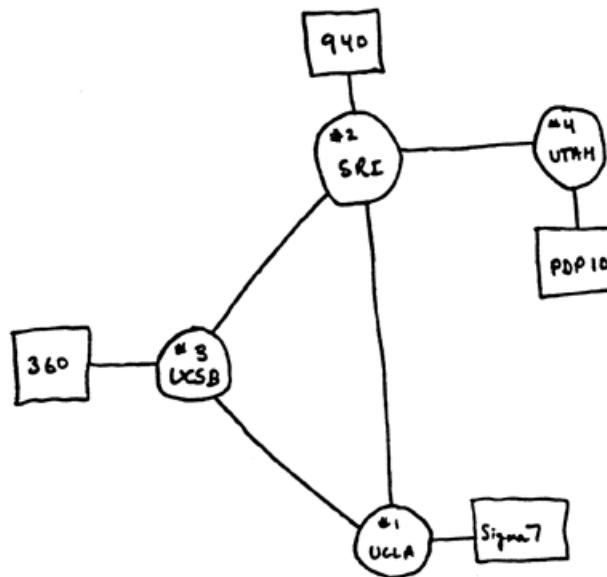


08/03/23

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA



Il Prof. Leonard Kleinrok e l'IMP



Un po' di storia...

- 1973: Prima connessione internazionale (Inghilterra e Norvegia).
- 1982: ARPANET adotta come standard i protocolli TCP/IP
 - nasce la prima definizione di Internet come un insieme di reti interconnesse, utilizzanti TCP/IP
 - il 1 Gennaio 1983 ARPANET passa al TCP/IP
 - L'università della California a Berkley rende di pubblico dominio il codice sorgente del TCP/IP
- 1986: Arpanet viene potenziata creando NSFNET (National Science Foundation NETwork).
- 1989: il numero di hosts supera 100.000. L'Italia si connette a NSFNET e quindi a Internet.
- 1990: ARPANET cessa di esistere, ma tutte le sue funzioni e infrastrutture rimangono operanti inalterate.
- 1992: viene rilasciato dal CERN il World Wide Web (WWW) ed il numero di hosts supera 1.000.000.
- 1993: la Casa Bianca e le Nazioni Unite si connettono ad Internet.
- 1994: iniziano i primi servizi commerciali e i primi centri commerciali virtuali. E' possibile ordinare una pizza direttamente tramite Internet presso "Pizza HUT".
- 1995: si vieta il trasporto di traffico di tipo commerciale su NSFNET, che rimane ad uso esclusivo degli enti di ricerca.
- 1996: WWW diventa il primo servizio di Internet in termini di quantità di bytes trasportati.

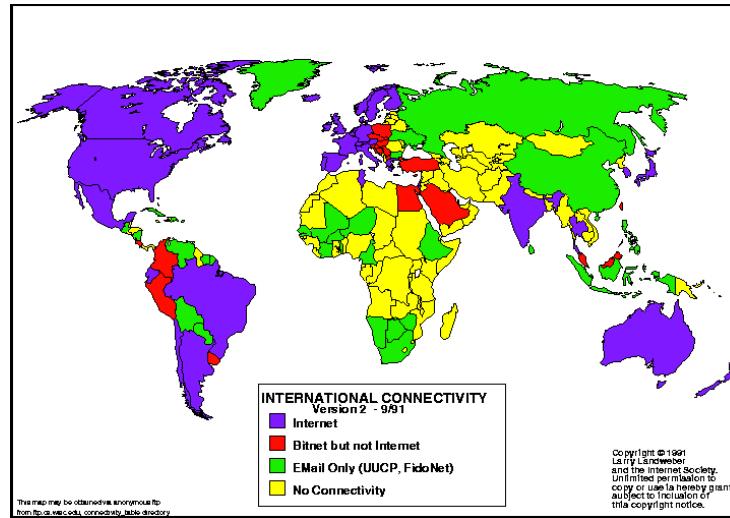


08/03/23

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

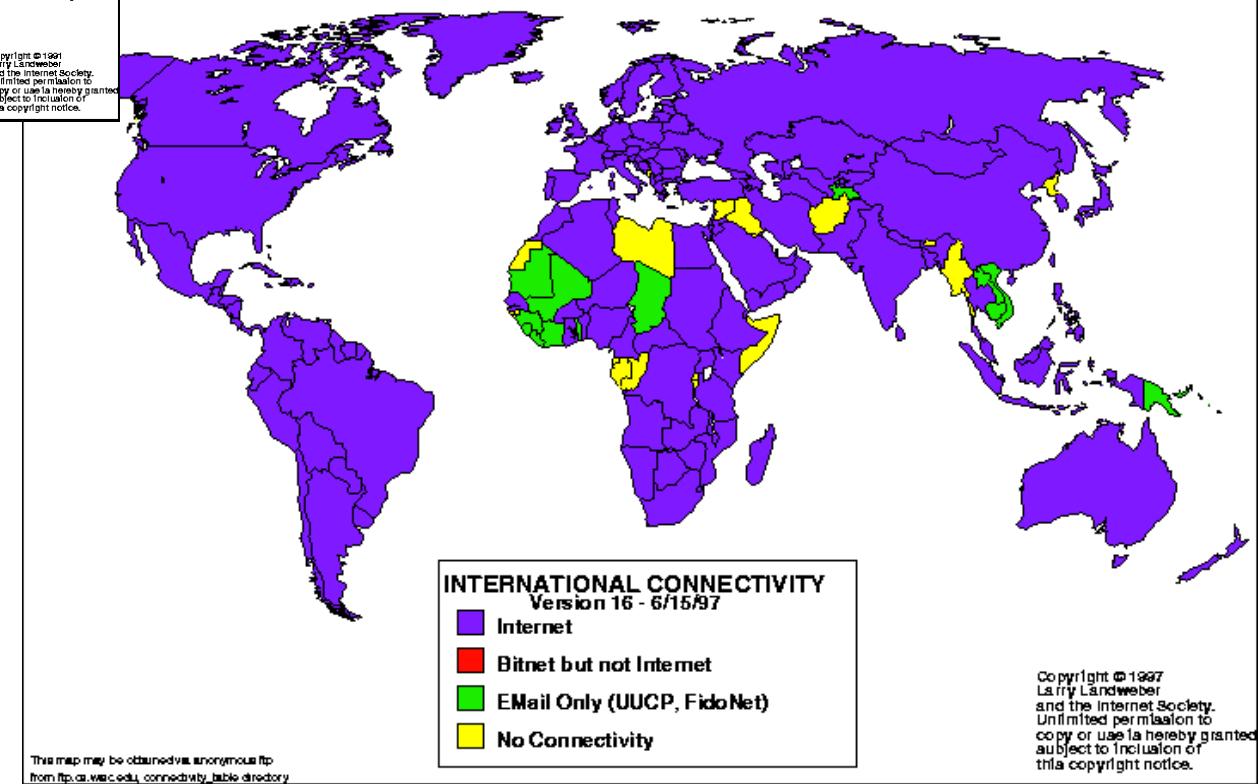


Diffusione di Internet



INTERNATIONAL CONNECTIVITY
Version 2 - 9/91
■ Internet
■ Bitnet but not Internet
■ EMail Only (UUCP, FidoNet)
■ No Connectivity

This map may be obtained via anonymous ftp from ftp.cs.wisc.edu, connectivity_table directory.
Copyright © 1991
Larry Landweber
and the Internet Society.
Unlimited permission to
copy or use is hereby granted
subject to inclusion of this
copyright notice.



INTERNATIONAL CONNECTIVITY
Version 16 - 6/15/97
■ Internet
■ Bitnet but not Internet
■ EMail Only (UUCP, FidoNet)
■ No Connectivity

This map may be obtained via anonymous ftp
from ftp.cs.wisc.edu, connectivity_table directory.
Copyright © 1997
Larry Landweber
and the Internet Society.
Unlimited permission to
copy or use is hereby granted
subject to inclusion of
this copyright notice.

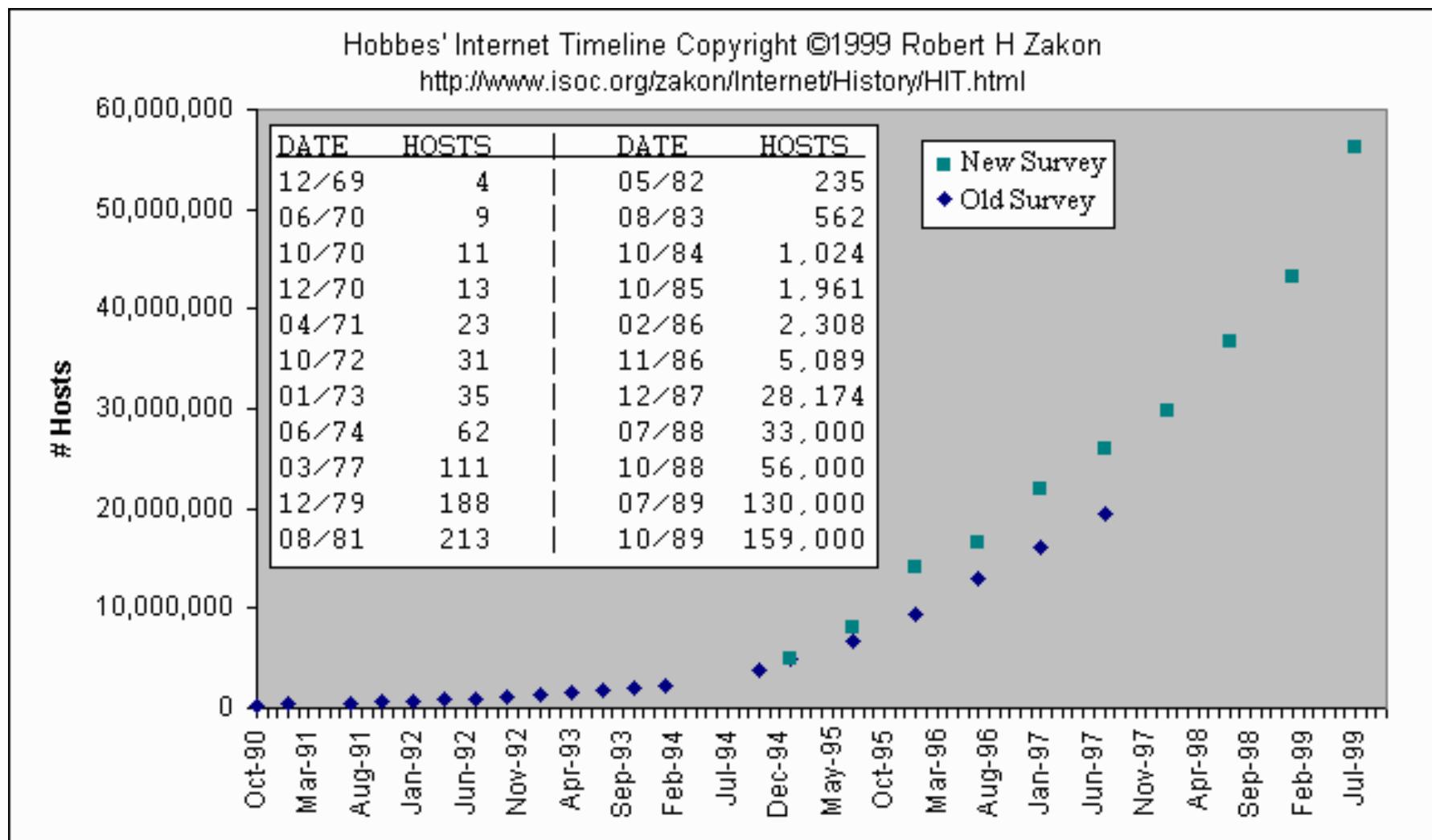


08/03/23

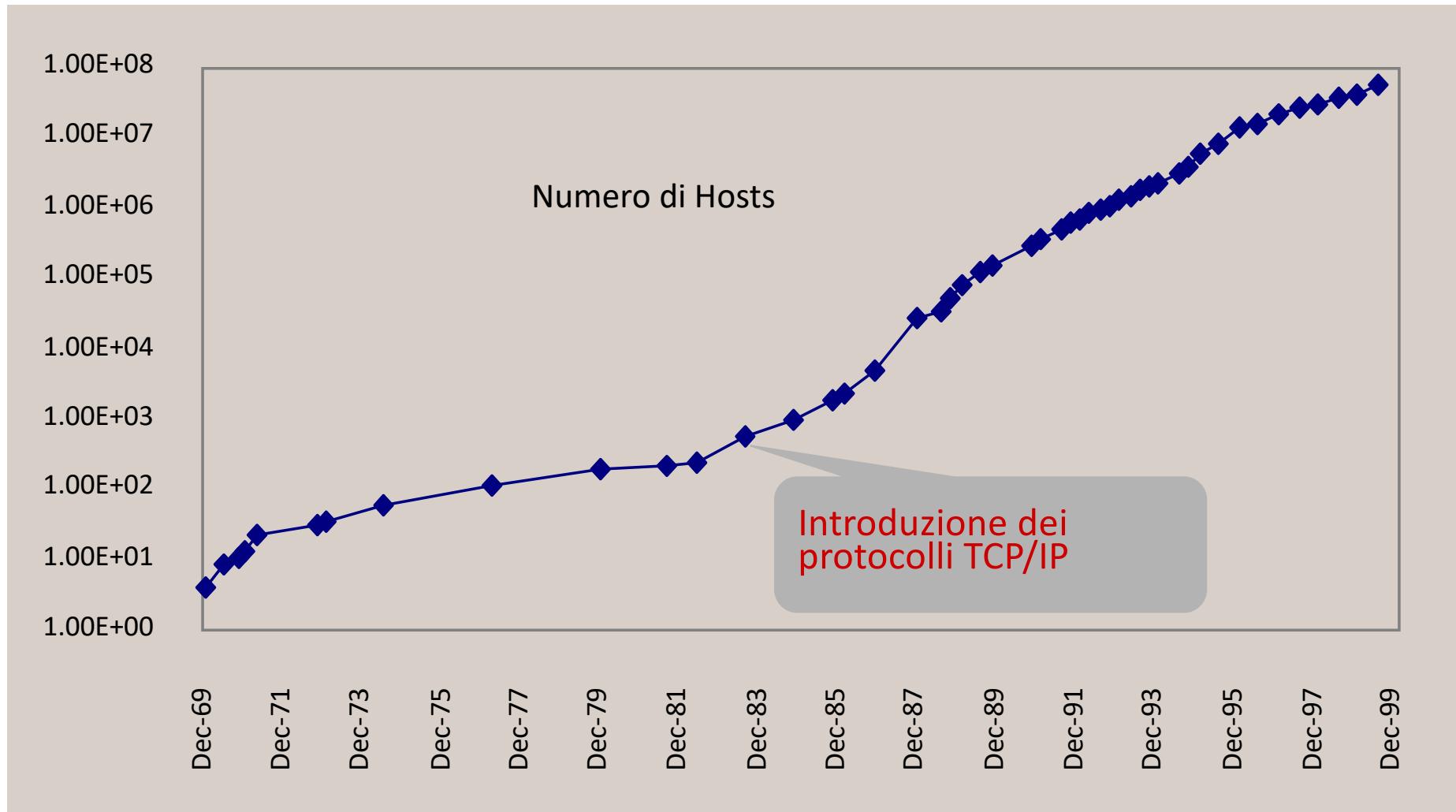
ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA



La crescita di Internet



La crescita di Internet



Standard di Internet: Enti di gestione

- Non esistono veri e propri enti che svolgono la funzione di gestione, ma solo enti di coordinamento delle attività di ricerca e di sviluppo che ora convergono nella Internet Society
- Da questa ora dipende il cosiddetto Internet Advisory Board (IAB) e si compone di due sottogruppi principali
 - **Internet Engineering Task Force (IETF)**, con lo scopo di coordinare le attività di ingegnerizzazione ed implementazione
 - **Internet Research Task Force (IRTF)**, con lo scopo di coordinare le attività di ricerca



RFC

- I protocolli sono frutto del lavoro di gruppi di ricerca
- I vari protocolli sono definiti in documenti detti **Request For Comment (RFC)**
- **RFC** sono documenti di pubblico dominio, distribuiti liberamente a chiunque li richieda, consultabili all'indirizzo

<http://www.ietf.org/rfc.html>

- Alcuni RFC diventano Internet Standard
 - Prima Proposed Standards
 - Poi Draft standards



08/03/23

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA



Altri Enti importanti

- NSF ha fondato un ente chiamato **InterNIC** (**Network Information Center**) allo scopo di fornire alcuni servizi specifici relativi ad Internet:
 - Registrazione di nuove reti e domini
 - Manutenzione di indici e database degli enti interconnessi
 - Servizi di tipo informativo sulla rete
- **IANA = Internet Assigned Number Authority**
 - Mantiene i database dei numeri che hanno significati convenzionali nei protocolli di Internet

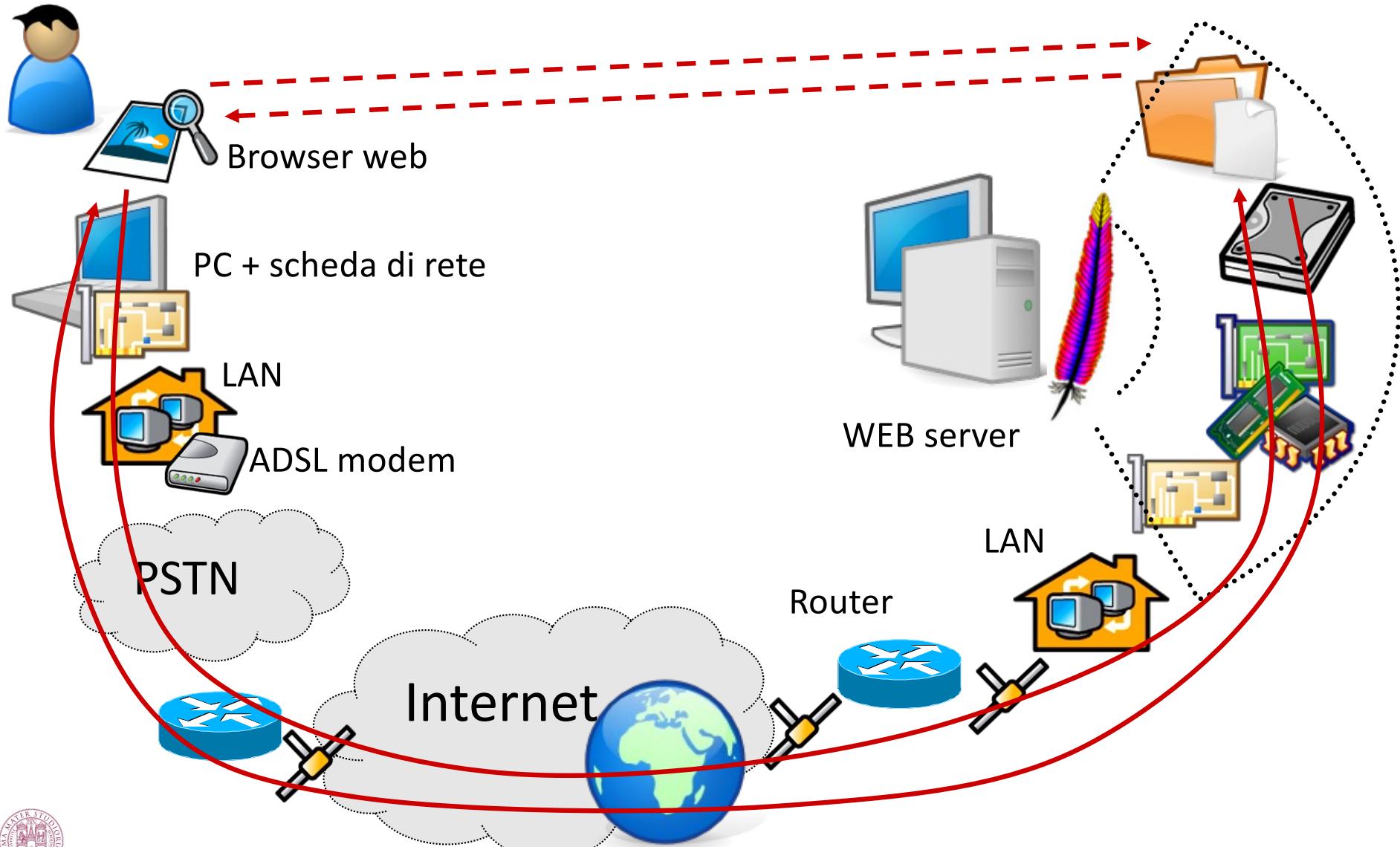


08/03/23

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

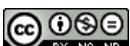


Utilizzo di Internet: chi è coinvolto?

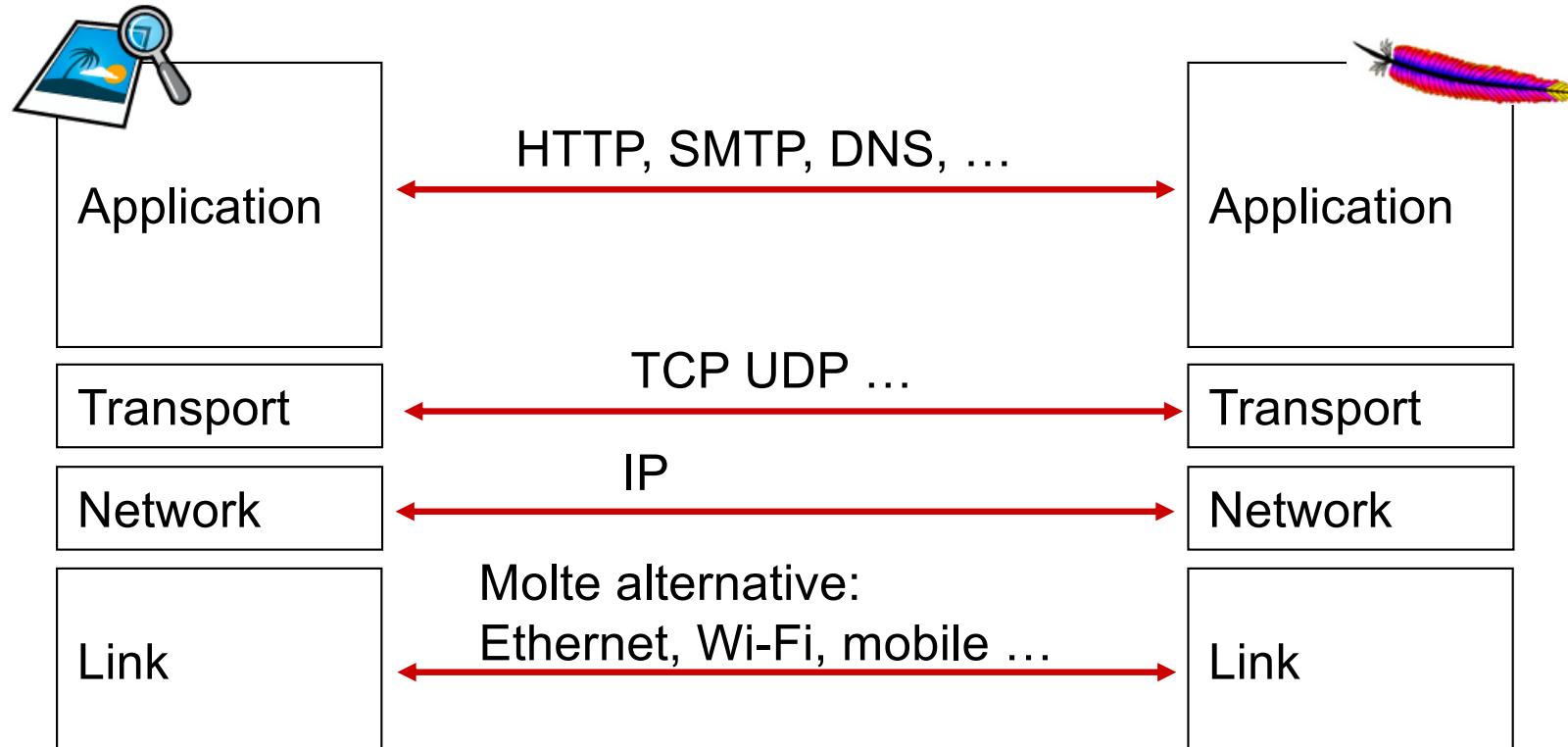


08/03/23

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA



Il modello a strati di Internet



Indirizzamento

- La comunicazione coinvolge due o più entità
 - Come fa il chiamante a specificare il chiamato?
- Diversi modi di “indirizzare”:
 - Esseri umani
 - Fanno riferimento a nomi simbolici facilmente ricordabili e scrivibili
 - Nodi di commutazione
 - Fanno riferimento a indirizzi, tipicamente numerici ben definiti e standardizzati
 - Sistemi di sicurezza
 - Fanno riferimento alle identità (certificate in qualche modo)
 - Applicazioni
 - Fanno riferimento a identificativi opportunamente definiti



08/03/23

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA



In Internet

- Tipicamente dobbiamo distinguere fra
 - Identifier
 - Identificativo di una certa risorsa di rete
 - Locator
 - Indirizzo necessario per localizzare tale risorsa
- Vengono definiti
 - Uniform Resource Identifier o URI
 - Uniform resource locator o URL



08/03/23

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA



Alcuni esempi

- Mobilità
 - Un terminale si sposta da una rete all'altra
 - Locator cambia nel tempo
 - Serve un identifier per mantenere la relazione con il terminale
- Multi-homing
 - Un terminale è connesso con più interfacce a infrastrutture diverse
 - Molteplici locator attivi in contemporanea
 - Potenzialmente un unico identifier



08/03/23

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA



Indirizzo globale e indirizzo locale

- Indirizzo *globale*

- È valido per tutta la rete
- Deve essere univoco (non devono esistere indirizzi replicati) per evitare ambiguità
- Va “assegnato” seguendo una procedura di gestione “globale” che assicura la non replicazione

- Indirizzi *locale*

- È valido limitatamente ad un certa sottoporzione della rete
 - Internamente ad un terminale
 - In un dominio di rete specifico
- Può non essere univoco
- Può essere assegnato con una procedura puramente “locale”



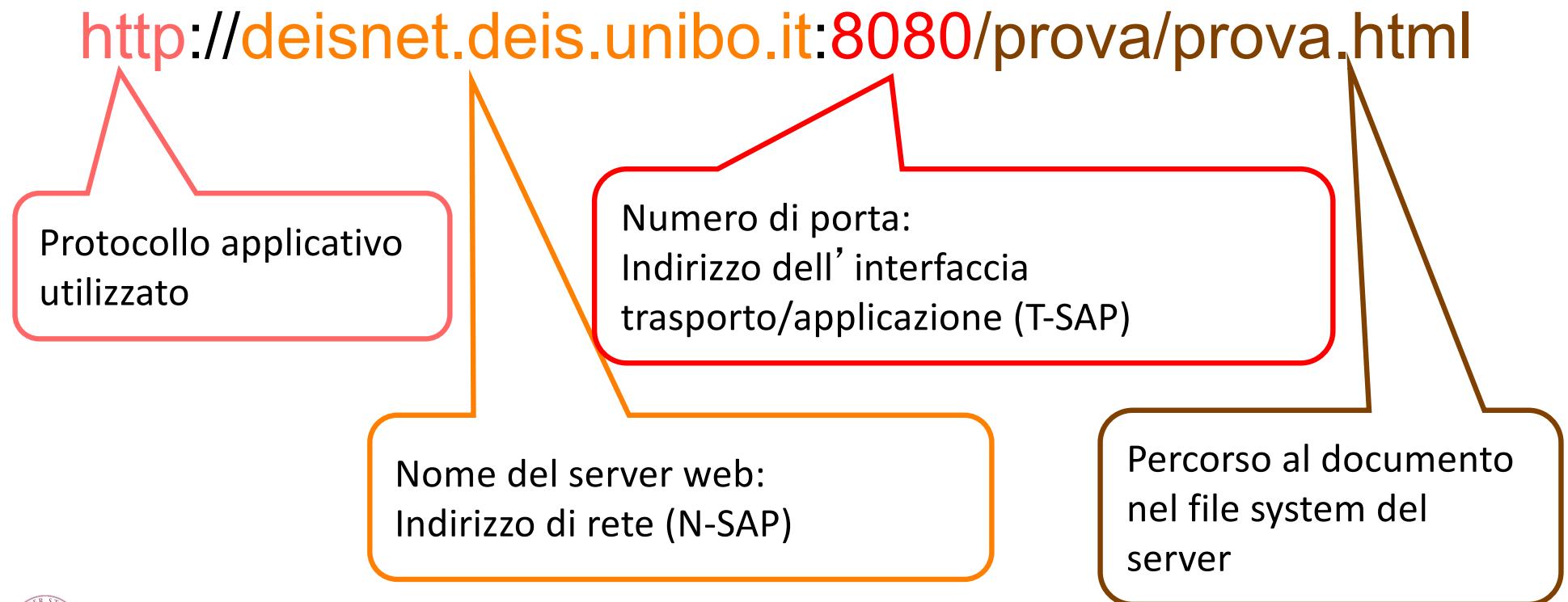
08/03/23

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA



URL

- La risorsa R è univocamente identificata da un indirizzo che la localizza (locator)
 - Uniform Resource Locator (URL)
 - URL è un indirizzo complesso che riflette l'organizzazione a livelli della rete



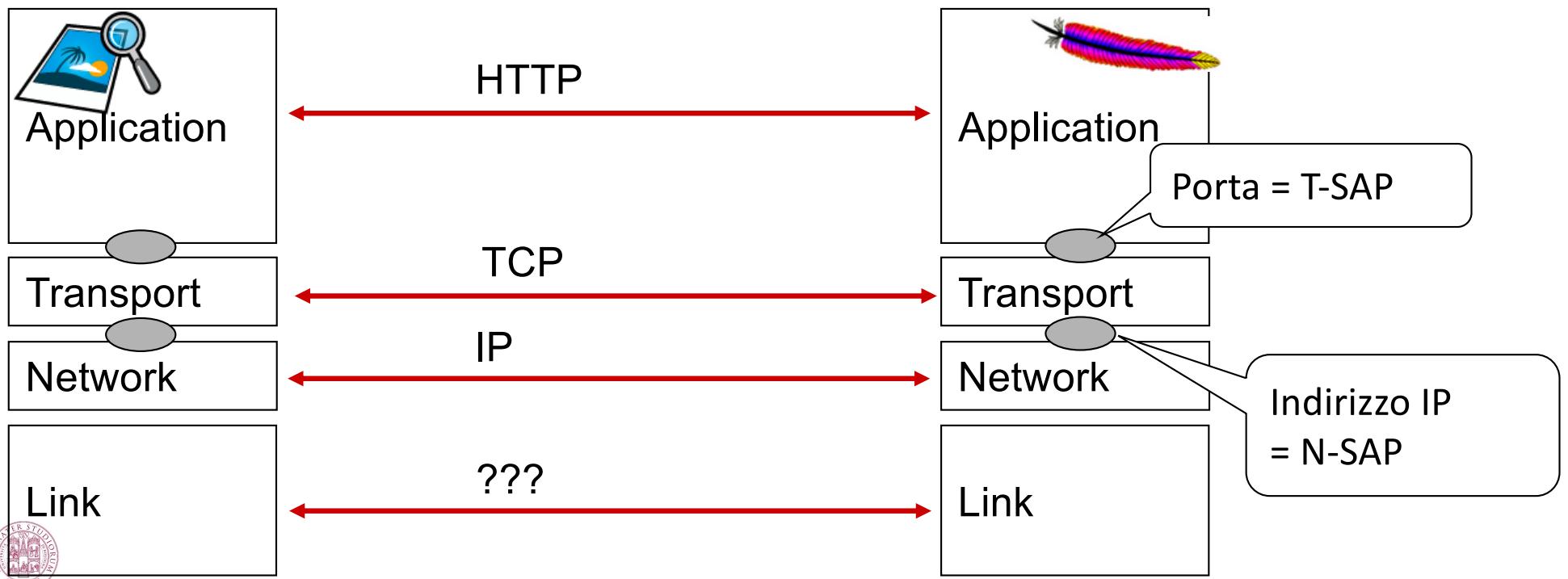
08/03/23

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

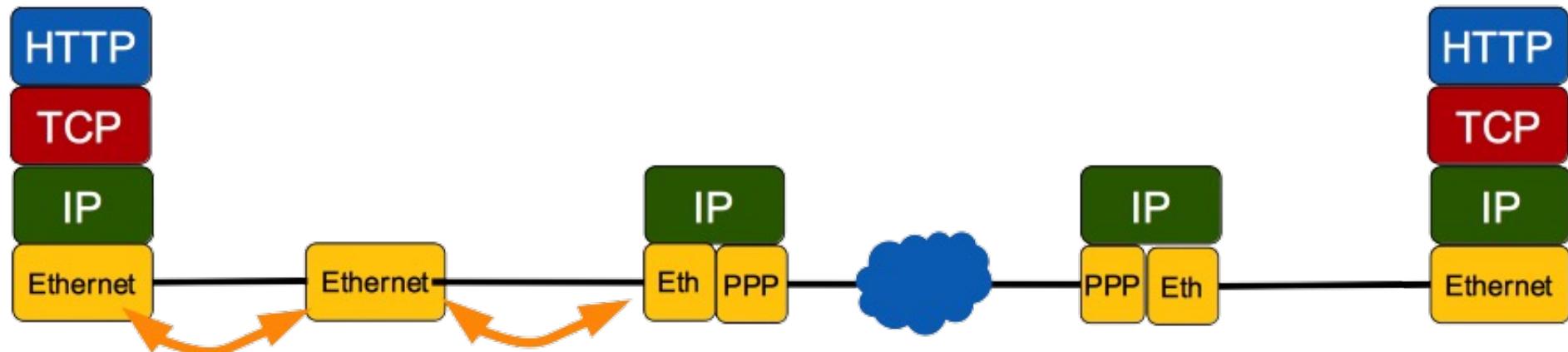
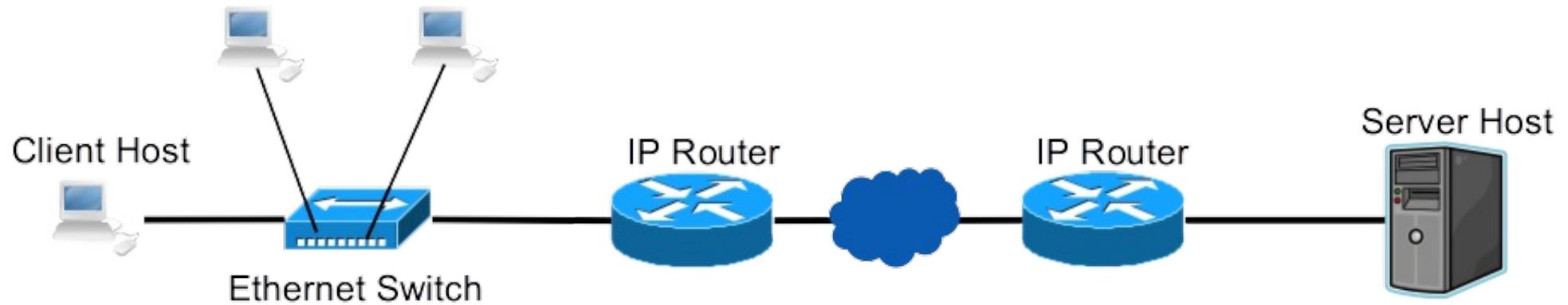


Cosa indirizzare

- Le applicazioni sono locali al calcolatore (terminale)
 - Alcune parti dell'URL hanno validità locale
 - Numero di porta
 - Percorso nel filesystem
- Il calcolatore va identificato univocamente su Internet
 - Almeno una parte dell'indirizzo deve avere significato unico e universale
 - Indirizzo di rete (numero IP)



Data Link



Obiettivo: connettività locale

Problema tipico: canale condiviso, tutti ricevono tutto

Soluzione: indirizzo per coinvolgere lo strato Ip solo se necessario

Infrastruttura fisica di accesso alla rete

- Tipicamente un calcolatore si connette alla rete tramite una rete di accesso locale detta LAN
- LAN = Local Area Network
- Le tecnologie LAN tipicamente implementano strato 2 e strato 1 secondo il modello ISO-OSI
 - Adottando soluzioni adatta al collegamento su breve distanza e finalizzate a
 - Canale di elevata capacità (bit rate > 10-100 Mbit/s)
 - Costi limitati (accessibili all'utente finale)
- Le tecnologie LAN oggi più comuni sono state sviluppate adottando un canale di trasmissione/ricezione condiviso fra tutti i calcolatori della LAN



08/03/23

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA



Canale condiviso

- Implicazioni
 - Comunicazione broadcast: uno parla tutti sentono
- Problemi
 - Controllo dell'accesso al canale
 - Come evitare di parlare uno sull'altro rendendo la comunicazione incomprendibile
 - Limitazione della quantità di dati da elaborare per lo strato 3
 - Se tutti i dati vengono ricevuti lo strato 3 dovrebbe elaborare anche le conversazioni degli altri
- È necessario un indirizzo LAN
 - L'interfaccia di strato 2 (che parla e riceve con la LAN) legge e passa allo strato 3 solamente i dati di sua pertinenza
- Chi decide gli indirizzi per la LAN?



08/03/23

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

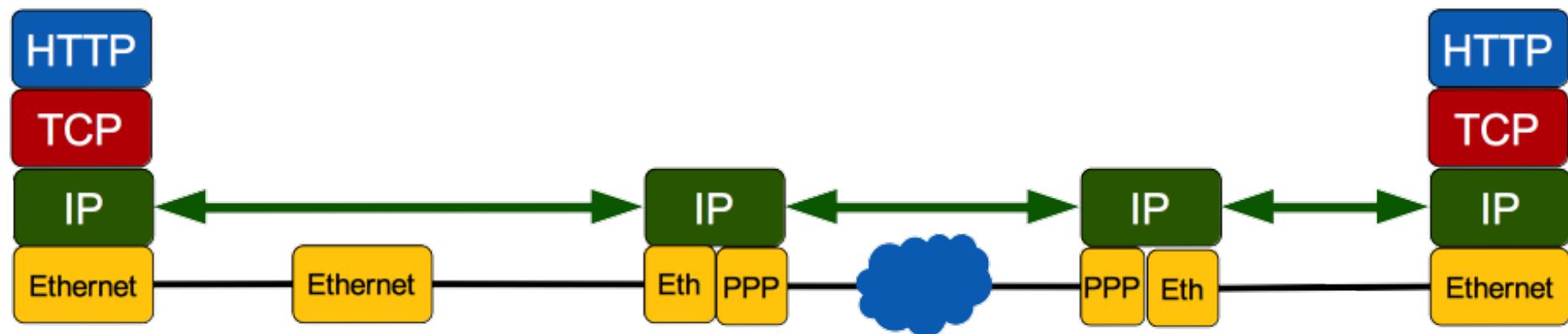
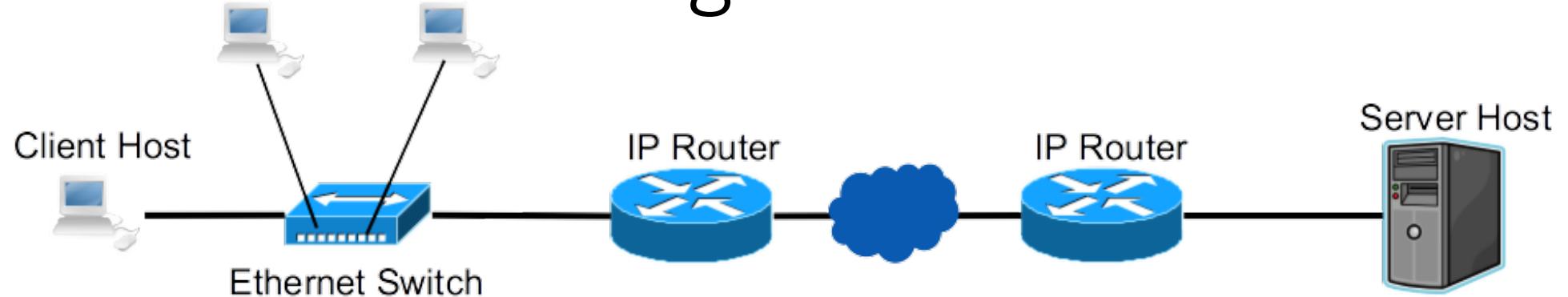


Gli indirizzi locali (MAC address)

- Sono composti da 48 bit (6 byte)
- Sono cablati nella scheda di rete
- Sono univoci a livello mondiale
 - i primi 3 byte individuano il costruttore
 - <http://www.macvendorlookup.com>
 - I secondi 3 numerano progressivamente le schede
- E' possibile specificare
 - un singolo destinatario (unicast)
 - 00-60-b0-78-e8-fd
 - un indirizzo di gruppo (multicast)
 - il primo bit deve essere a 1
 - un invio a tutte le stazioni (broadcast)
 - ff-ff-ff-ff-ff-ff



Internetworking



Obiettivo: connettività globale

Creare ponti fra le varie tecnologie locali

Indirizzo per indicare la destinazione finale a prescindere dai passi intermedi

L'indirizzo IP

- Indirizzi di lunghezza fissa pari a **32 bit**
- Scritti convenzionalmente come sequenza di 4 numeri decimali, con valori da **0** a **255**, separati da punto (rappresentazione **dotted decimal**)

10001001.11001100.11010100.00000001

137.204.212.1

- Numero teorico max. di indirizzi

$$\mathbf{2^{32} = 4.294.967.296}$$

- In realtà si riesce a sfruttare un numero molto inferiore



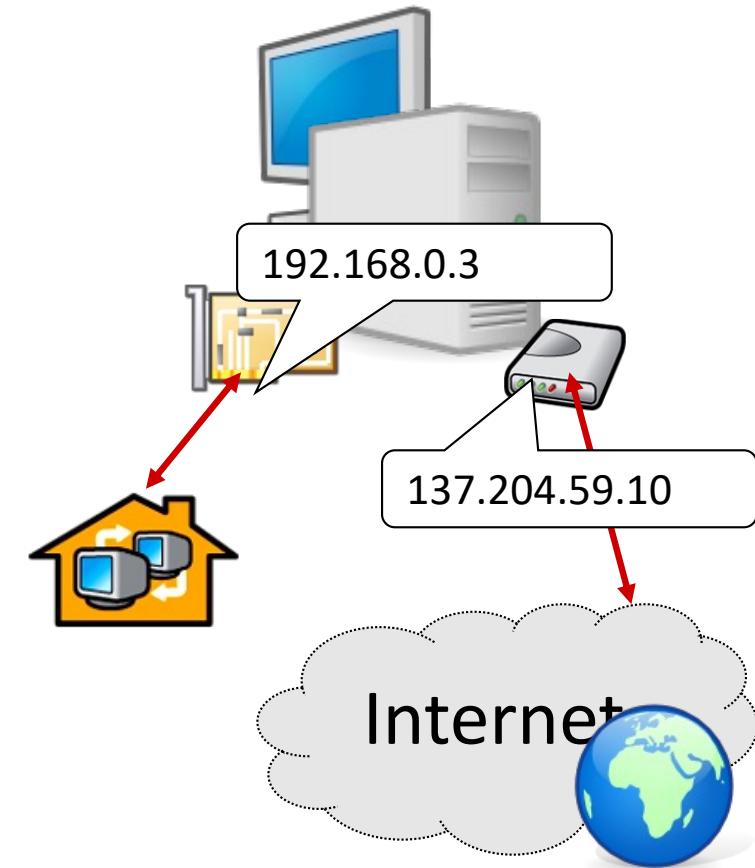
08/03/23

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA



Indirizzi e interfacce di rete

- L'indirizzo identifica i punti di interconnessione di un host con la rete
 - Non identifica un host individuale, ma una delle sue interfacce di rete
- **Multi-homed hosts**
 - host con due o più interfacce di rete
- Esempio: un router che collega N reti ha
 - N interfacce di rete
 - N distinti indirizzi IP, uno per ogni interfaccia di rete

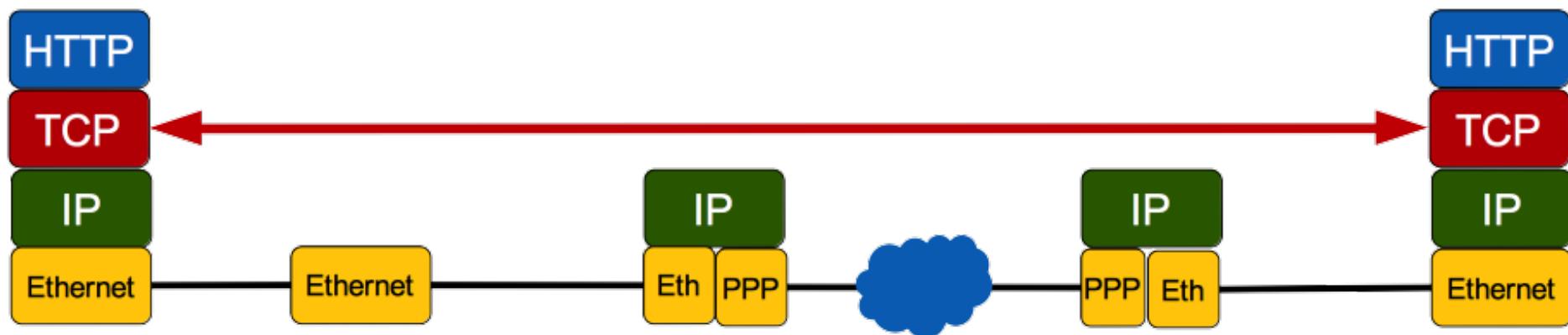
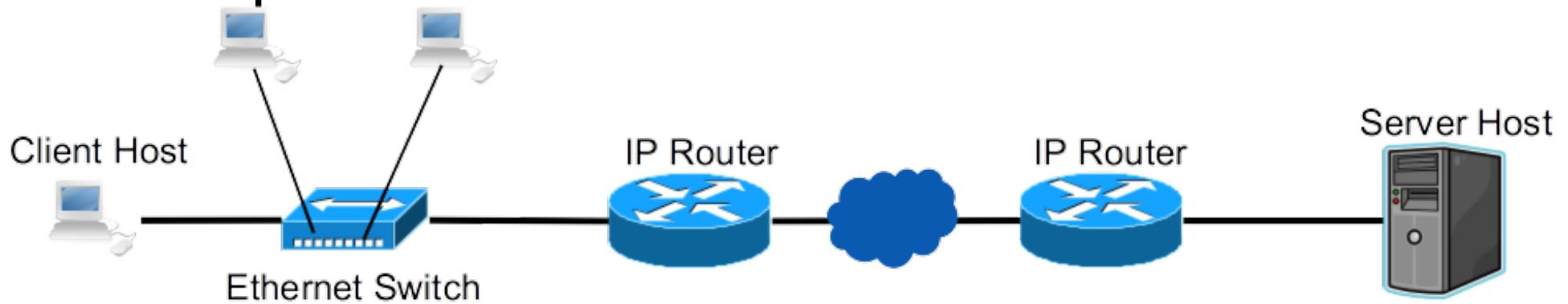


08/03/23

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA



Trasporto end-to-end



Obiettivo: garantire il dialogo affidabile fra le applicazioni finali

Protocollo end-to-end

Indirizzamento per identificare la singola applicazione una volta
raggiunto il calcolatore

Numero di porta

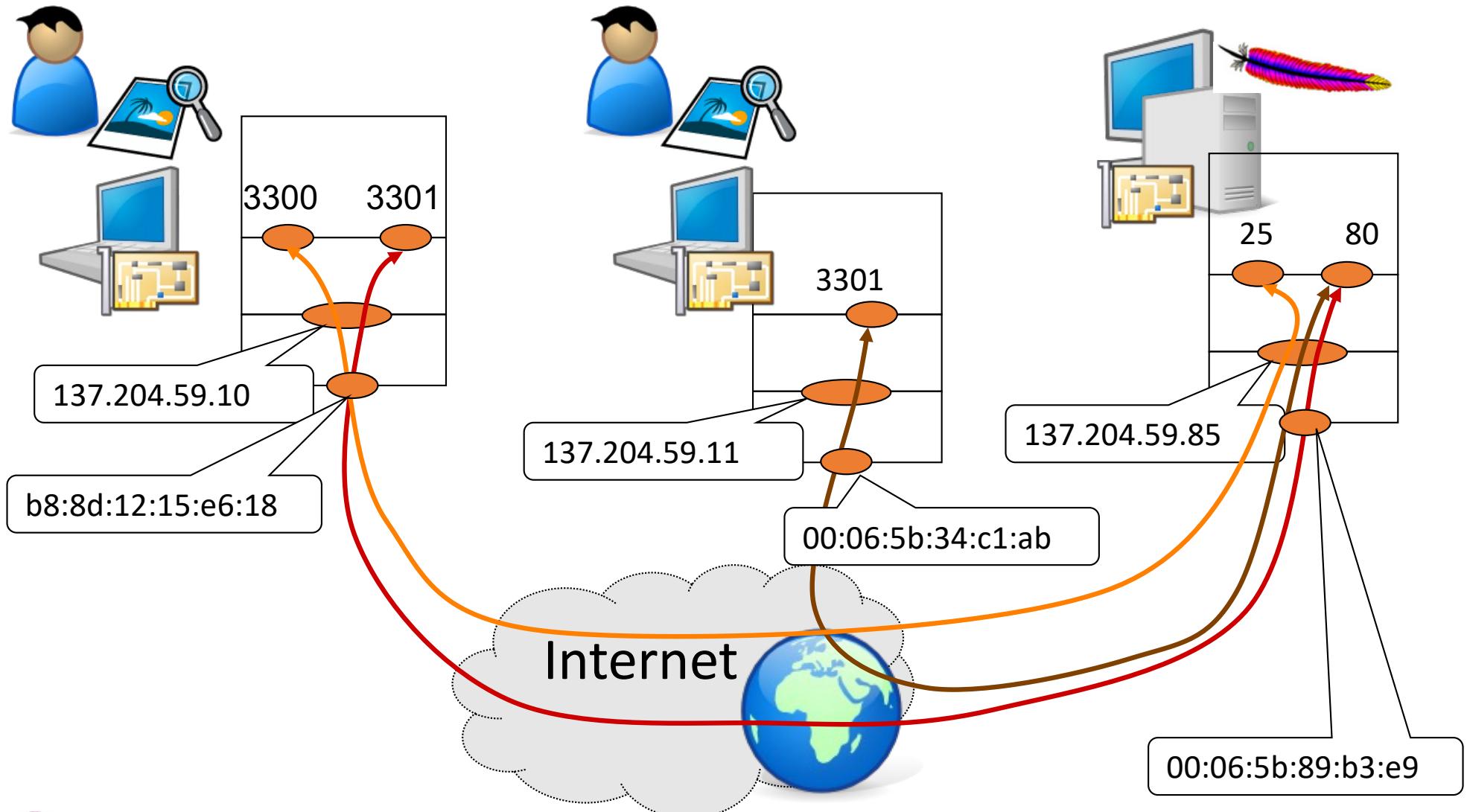
- Indirizzo di 16 bit
 - Valori decimali da 0 a 65535
- Locale al singolo calcolatore, ripetute su tutti i calcolatori
- Condiviso fra tutti i protocolli di trasporto

The screenshot shows the Wireshark interface with the following details:

- Filter:** tcp.stream eq 0
- Table Headers:** No., Time, Source, Destination, Protocol, Info
- Table Data:** 7 rows of captured network frames.
- Frame Details:** Frame 1 is expanded, showing:
 - Frame 1: 62 bytes on wire (496 bits), 62 bytes captured (496 bits)
 - Ethernet II, Src: DellComp_89:b3:e9 (00:06:5b:89:b3:e9), Dst: DellComp_ec:46:62 (00:b0:d0:ec:46:62)
 - Internet Protocol, Src: 192.168.10.199 (192.168.10.199), Dst: 137.204.57.85 (137.204.57.85)
 - Transmission Control Protocol, Src Port: ddt (1052), Dst Port: http (80), Seq: 0, Len: 0
 - Source port: ddt (1052)
 - Destination port: http (80)
 - [Stream index: 0]
 - Sequence number: 0 (relative sequence number)
 - Header length: 28 bytes
 - Flags: 0x02 (SYN)
 - Window size: 65535
 - Checksum: 0x6bd [validation disabled]
 - Options: (8 bytes)
- Hex View:** Shows the raw binary data for the selected frame.
- Statistics:** Frame (frame), 62 bytes
- Display:** Packets: 31 Displayed: 31 Marked: 0 Load time: 0:00.005
- Profile:** Default
- CC BY NC ND License:** A standard Creative Commons license logo is in the bottom right corner.



Flussi di comunicazione



Connessioni

- Per identificare il singolo flusso è sufficiente conoscere:
 - IP sorgente
 - IP destinazione
 - Porta sorgente
 - Porta destinazione
- Nell'esempio precedente sono attivi
 - 137.204.59.10:3300 \longleftrightarrow 137.204.57.85:25
 - 137.204.59.10:3301 \longleftrightarrow 137.204.57.85:80
 - 137.204.59.11:3301 \longleftrightarrow 137.204.57.85:80



08/03/23

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA





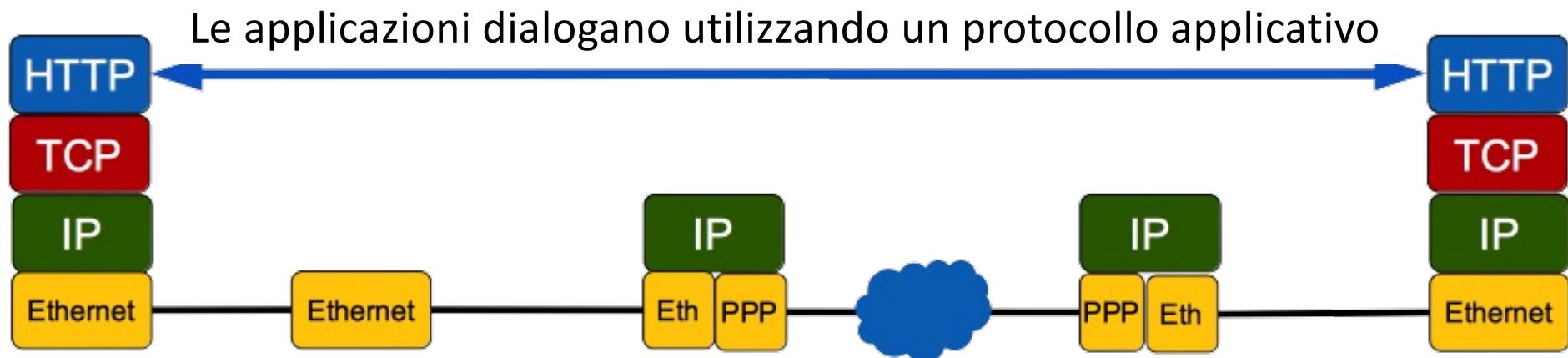
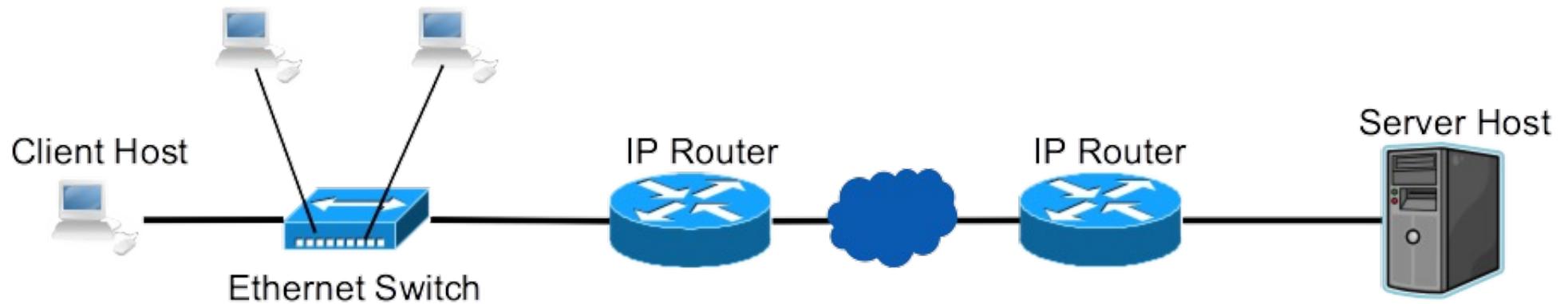
ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Implementazione dei servizi e modelli di interazione

Franco CALLEGATI



Il modello di Internet



Server e client

- Con il termine server indichiamo un'applicazione che
 - rende disponibile un servizio
 - mediante un'interfaccia standard (protocollo)
- Con il termine client indichiamo un'applicazione che
 - È in grado di utilizzare i servizi messi a disposizione da un server.
- *Attenzione:* a volte gli stessi termini vengono usati impropriamente per indicare l'host che ospita l'applicazione



Apertura

- Il processo Server si predisponde a ricevere una connessione eseguendo una **apertura passiva**
 - Si mette in ascolto in attesa dell'arrivo di una richiesta di connessione
 - Questo processo nel mondo Unix è chiamato Demone
- Il processo Client esegue una **apertura attiva** tentando di collegarsi al processo server di destinazione



08/03/23

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA



Caratteristiche del client server

- Modello molti (client) a uno (server)
- Tipicamente sincrono e bloccante
 - Se il server non risponde il client non può avanzare
 - Si implementa nel client come reagire a fronte di un timeout (inferire un guasto?)
- Tipicamente con accoppiamento (binding) dinamico
 - Ad ogni invocazione il client può scegliere il server
 - Se il server non è disponibile dove atteso, la rete restituisce un errore
- Il server può essere progettato per offrire diversi tipi di prestazione
 - Servizio sequenziale o parallelo
 - Connessione persistente con notifiche push
 - Autenticazione e autorizzazione



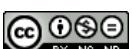
Peer to peer

- La variante **Peer-to-peer** (P2P)
 - Gli host in rete sono tutti equivalenti (peer, appunto) e fungono alternativamente sia da client che da server verso altri nodi
 - In una rete P2P qualsiasi nodo utilizza e mette a disposizione contemporaneamente risorse ed informazioni in rete
- Auto scalabile
 - Nuovi peer agenti da server possono essere aggiunti fornendo nuove capacità e funzionalità



08/03/23

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA



La ricerca della destinazione

- Il client deve conoscere indirizzo IP e Numero di porta del server di destinazione
 - Come fa a scoprirli?
- Nell' URL sono specificati
 - Protocollo applicativo
 - A cui corrisponde una well known port
 - Eventuale numero di porta non standard
 - Il numero IP o il nome del server
 - Il nome deve tramutarsi in un numero IP
 - Il numero IP identifica in modo univoco il punto di accesso alla rete del server
- Come fa il nome a diventare un numero?



08/03/23

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA



Nomi simbolici

- Affinchè un calcolatore possa parlare con altri calcolatori su Internet:
 - **DEVE** essere equipaggiato con almeno una interfaccia di rete
 - L'interfaccia di rete **DEVE** avere associato un numero IP
- Al calcolatore **PUO'** essere associato un nome simbolico
 - Sequenza di stringhe alfanumeriche separate da punti
deisnet.deis.unibo.it
- Perché un nome simbolico? Per renderlo più facilmente fruibile agli esseri umani



Il Domain Name System

- I calcolatori devono utilizzare i numeri IP a 32 bit nelle PCI delle PDU di livello 3
- Deve esistere un metodo per «tradurre» un nome simbolico in un numero IP
- Questo metodo si basa su un'applicazione di rete, che svolge funzioni di gestione della rete stessa, chiamato Domain Name System o DNS
- Del DNS si parlerà in seguito.



Il numero di porta

- Viene scelto dalla singola applicazione quando si attiva
- In teoria potrebbe cambiare di volta in volta
- Per permettere ai Client di raggiungere i Server questi ultimi utilizzano sempre gli stessi numeri di porta
 - Vengono detti numeri di porta «**Well Known**»



IANA (Internet Assigned Numbers Authority)

PORt NUMBERS

(last updated 2010-09-21)

The port numbers are divided into three ranges: the Well Known Ports, the Registered Ports, and the Dynamic and/or Private Ports.

The Well Known Ports are those from 0 through 1023.

DCCP Well Known ports SHOULD NOT be used without IANA registration. The registration procedure is defined in [RFC4340], Section 19.9.

The Registered Ports are those from 1024 through 49151

DCCP Registered ports SHOULD NOT be used without IANA registration. The registration procedure is defined in [RFC4340], Section 19.9.

The Dynamic and/or Private Ports are those from 49152 through 65535



08/03/23

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA



Numeri di porte well known

- Regole d' uso
 - Da 1 a 1023 (in origine da 1 a 255): **Riservati**
 - possono essere usati solo dai server
 - da 1024 a 49151: **Registrati**
 - Sono usati da alcuni servizi ma anche da client
 - Da 49151 a 65535: **ad uso dei client**
- Una parte dei numeri di porta sono riservati (Well Known Ports)

#	Protocol	Servizio
21	FTP-CONTROL	File Transfer Protocol Trasferimento file (control)
20	FTP-DATA	File Transfer Protocol Trasferimento files (dati)
23	TELNET	Accesso via terminale
25	SMTP	Trasferimento di posta elettronica
53	DNS	Accesso al DNS
80	HTTP	Web server
109	POP2	Post Office Protocol (Version 2) Lettura posta elettronica
22	SSH	Secure Socket Accesso via terminale cifrato
110	POP3	Post Office Protocol (version 3) Lettura posta elettronica
137	NETBIOS Name Service.	Servizio di rete per applicazioni in ambiente DOS (Windows)
138	NETBIOS Datagram Service.	
139	NETBIOS Session Service.	
443	HTTPS	HTTP over SSL/TLS WEB cifrato



Analisi di protocollo

- Esistono strumenti software per analizzare il traffico di rete
 - Wireshark <http://www.wireshark.org/>

The screenshot shows the Wireshark interface with the following details:

- Toolbar:** Standard file operations (File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Tools, Help), selection tools (Select, All, Previous, Next, Find, Replace, Copy, Paste, Cut, Delete, Undo, Redo), and a zoom slider.
- Filter Bar:** Shows the current filter: `tcp.stream eq 0`. Includes a dropdown for expression, a clear button, and an apply button.
- Packets List:** A table showing 17 captured TCP packets. The columns are No., Time, Source, Destination, Protocol, and Info. The "Info" column provides detailed protocol analysis for each packet.
- Details Pane:** Shows the detailed structure of the selected packet (Frame 6). It includes fields for wire length (1514 bytes), capture length (1514 bytes), source (DellComp_ec:46:62), destination (DellComp_89:b3:e9), Internet Protocol version (4), source IP (137.204.57.85), destination IP (192.168.10.199), and Transmission Control Protocol port information.
- Hex pane:** Displays the raw byte sequence of the selected frame, starting with 0000 and ending with 0070.
- ASCII pane:** Displays the ASCII representation of the selected frame's payload.

Funzioni di controllo del software

Qui viene mostrata la sequenza dei messaggi
Una riga per messaggio

Contenuto del messaggio evidenziato sopra
Vengono evidenziati i vari protocolli utilizzati nel messaggio

Contenuto del messaggio in forma di sequenza di byte

In conclusione

- L'utente finale interagisce con il software di applicativo
- L'applicazione dialoga con una o più applicazioni remote utilizzando i protocolli applicativi necessari
- I protocolli applicativi sfruttano il servizio di trasporto di uno dei protocolli di trasporto per raggiungere l'applicazione remota
- Il protocollo di trasporto utilizza le capacità di instradamento di IP per la consegna dei dati al calcolatore remoto dove risiede l'applicazione
- IP consegna i dati sfruttando l'infrastruttura di rete a cui gli host sono connessi tramite l'interfaccia LAN



08/03/23

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA





ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

GRAZIE PER L'ATTENZIONE

www.unibo.it/sitoweb/franco.callegati





ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Domain Name System

Franco CALLEGATI



Nomi e indirizzi

- Per comodità degli utenti ai numeri IP sono associati dei nomi simbolici
- Nome simbolico
 - Sequenza di stringhe alfanumeriche separate da punti
- Il numero di stringhe è virtualmente illimitato

deisnet.deis.unibo.it



Domain Name System

- Per eseguire la ricerca degli indirizzi a partire dai nomi si utilizza un servizio automatico
 - Una sorta di elenco telefonico informatico
- **Domain Name System (DNS)** è un database distribuito che associa ad ogni Nome il relativo indirizzo di rete
- La consultazione del DNS avviene tramite opportuni “server” DNS
 - La consultazione è tipicamente trasparente per l’utente
 - Il browser sa cosa fare per consultare il DNS senza doverlo chiedere all’utente finale



Gestione del database dei nomi

- **PROBLEMA** – come fare a gestire un database con tutti i nomi degli host di Internet?
- **SOLUZIONE** – database distribuito
 - Lo spazio dei nomi è suddiviso in **zone (domini)** non sovrapposte, che contengono uno o più sottodomini
 - Ciascuna zona prevede un **name server** principale ed uno o più server secondari
 - Ogni name server è a conoscenza degli indirizzi IP corrispondenti ai nomi degli host contenuti nella sua zona, di cui è responsabile



Come si compone un nome?

- Le stringhe non sono arbitrarie
- Le componenti del nome riflettono l'organizzazione gerarchica dei **Domini**
- Ai domini vengono associate dei nomi convenzionali
 - **it** = stringa identificativa del dominio Italia
 - **unibo** = stringa identificativa del dominio Università di Bologna
 - **deis** = stringa identifica del Dipartimento entro Unibo
- I domini possono essere suddivisi in **sottodomini**
 - *unibo* è un sottodominio di *it*
 - *deis* è un sottodominio di *unibo*



Il nome

- Sequenza dei nomi di dominio a partire dal più esteso a destra

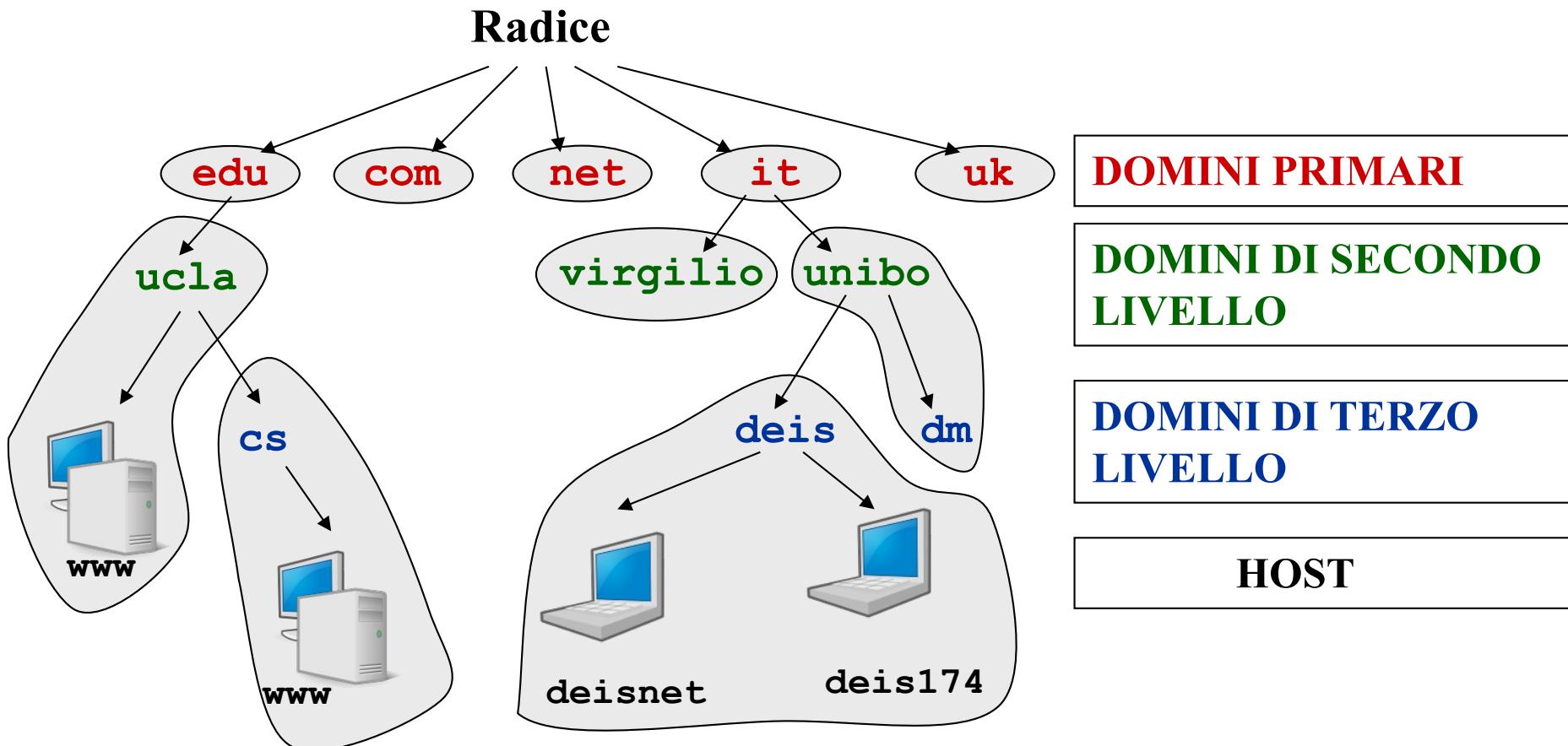
deisnet.deis.unibo.it



- Il nome specifico dell'host è arbitrario
- I nomi dei domini sono assegnati da IANA
- Non devono esistere due nomi uguali per host diversi



Suddivisione in zone



Domini primari

edu istituzioni scolastiche o di ricerca USA

gov istituzioni governative USA

com organizzazioni commerciali

mil gruppi militari USA

org altre organizzazioni

net centri di supporto alla rete

country code sigle standard per identificare le nazioni
(ISO 3166)

it fr uk de au jp ie dk br ...



Registro.it

- Il Registro è l'anagrafe dei domini Internet .it
 - Soltanto qui è possibile chiedere, modificare o cancellare uno o più domini .it.
- Su richiesta degli utenti, il Registro associa un gruppo di indirizzi numerici a un nome
 - L' associazione è memorizzata nel **Dbna** (database dei nomi assegnati) che tutti i computer collegati in rete devono consultare per raggiungere un dominio .it
- Le regole della rete sono fissate da un'organizzazione internazionale, Icann (Internet Corporation for Assigned Names and Numbers).
- Nel 1987, Icann ha incaricato il Consiglio Nazionale delle Ricerche di gestire i domini Internet a targa .it.
 - E' nato così il Registro .it, che ha sede all'Istituto di Informatica e Telematica del Cnr di Pisa



Il servizio Whois

- Con il servizio whois è possibile verificare se è assegnato ed a chi un determinato nome di dominio
- Ricercando **unibo.it** si ottiene

Dominio

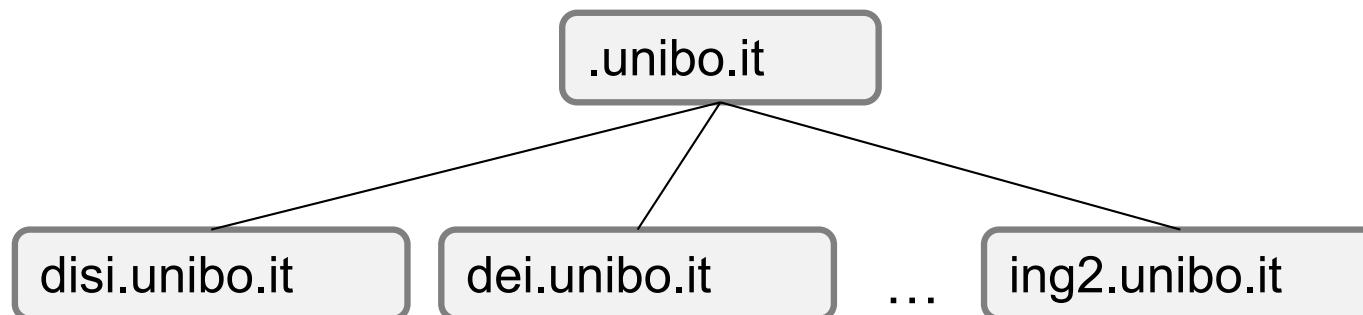
Dominio: unibo.it
Stato: ok
Firmato: no
Data Creazione: 29 gen 1996, 00:00:00 CET
Data Scadenza: 2 mag 2023 CET
Data Aggiornamento: 18 mag 2022, 00:59:38 CET

Registrante

Organizzazione: ALMA MATER STUDIORUM - Universita' di Bologna
Indirizzo: v.le Filopanti, 3
40126 - Bologna (BO)
it
Nazionalità: it
Telefono: +39.0512080300
Fax: +39.0512095919
E-Mail: assistenza.cesia@unibo.it
Data Creazione: 1 mar 2007, 10:47:03 CET
Data Aggiornamento: 9 apr 2021, 08:54:13 CET

La gerarchia

- L'assegnatario di un dominio è responsabile della gestione di eventuali sottodomini
 - I sottodomini non vengono registrati



Dominio

Dominio:

disi.unibo.it

Dominio non assegnabile



Risolvere un nome

- Per convertire un nome in numero IP
 - L'host deve essere equipaggiato con un programma specifico detto name resolver
 - Dipende da implementazione e sistema operativo
 - Il protocollo di dialogo col server è standard
 - Nell'host deve essere configurato l'indirizzo IP del (dei) server DNS della zona di appartenenza
 - Nell'host possono essere pre-configure alcune corrispondenze nomi-numeri in un archivio locale
 - Nome del file e sintassi dipendenti dall'implementazione
- Quando un'applicazione deve risolvere un nome invoca il *name resolver*



Name resolver

- Si possono verificare i seguenti casi
 - Il name resolver può risolvere il nome localmente (grazie ad un archivio locale, cache o file)
 - Comunica direttamente il numero IP all' applicazione
 - Il resolver non può risolvere il nome localmente
 - Interroga il name server della zona a cui appartiene l' host
- Il name server della zona risolve il nome cooperando con server DNS di altre zone
 - Contatta prima di tutto il name server del dominio di primo livello del nome da risolvere
 - Eventualmente contatta domini di livello inferiore

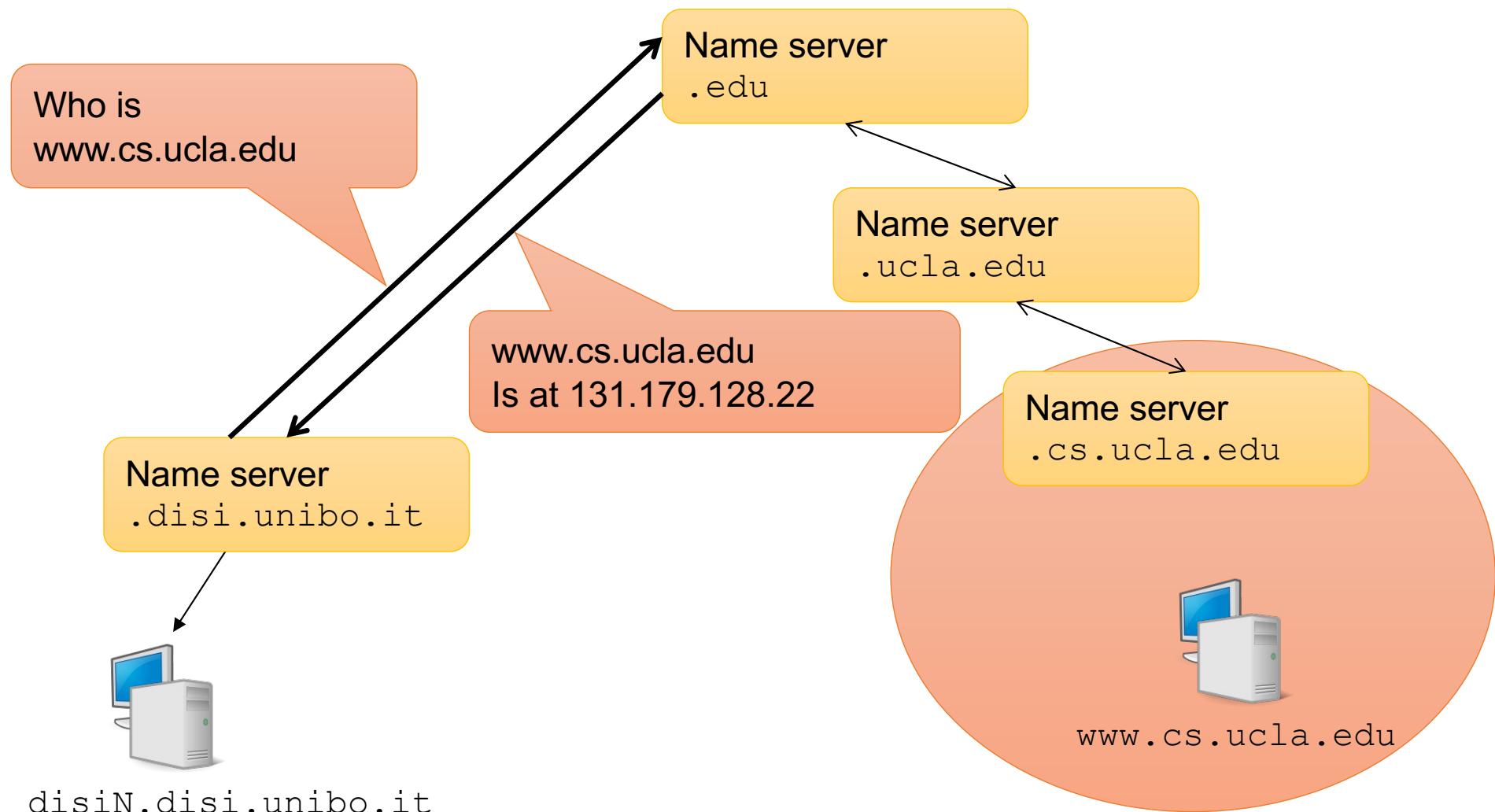


Risposta in modalità ricorsiva e iterativa

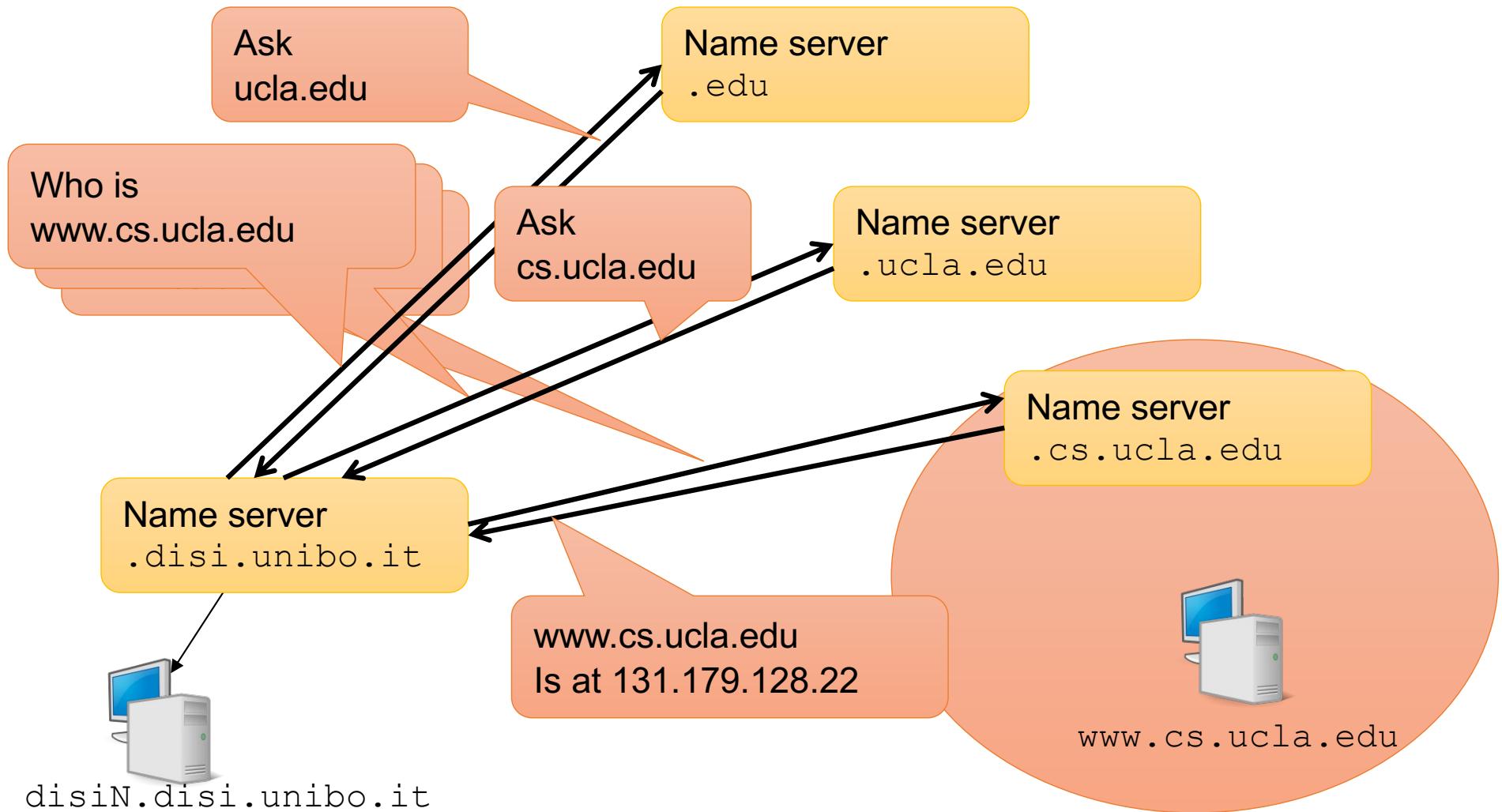
- La risposta all'interrogazione del name server di zona può avvenire in modalità
 - Ricorsiva
 - Il name server interrogato si preoccupa di risolvere il nome interrogando eventuali server di sotto-dominio e risponde alla richiesta
 - Iterativa
 - Il name server interrogato risponde indicando un name server di sottodominio a cui delega la risoluzione della richiesta



Modalità ricorsiva



Modalità iterativa

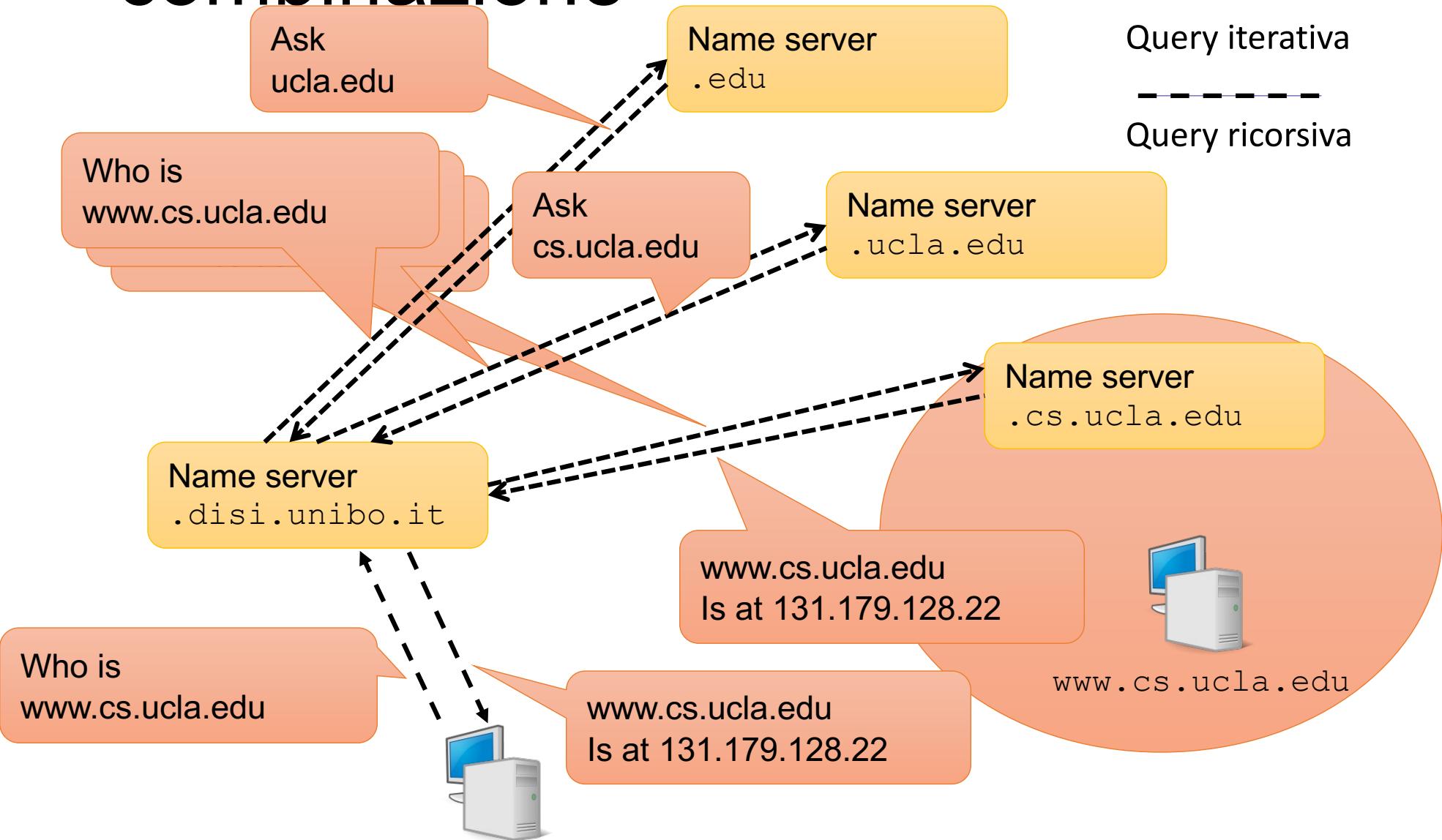


In sintesi

- Iterativa
 - “Io non conosco questo nome ma puoi chiederlo a questo server”
 - Il server DNS a cui viene inviata la query
 - NON fornisce risposta esaustiva alla query
 - fornisce il nome del server che lui ritiene possa avere la risposta alla query
- Ricorsiva
 - “Attendi che penso io a recuperare il nome richiesto”
 - Il server DNS a cui viene inviata la query
 - Si affida al server radice del dominio richiesto per avere la traduzione
 - Fornisce risposta esaustiva alla query



Normalmente si usa una combinazione



Server ricorsivi e autorevoli

- Recursive DNS

- Server che può produrre una risposta per la domanda considerata
- Tipicamente il primo server raggiungibile nel dominio
- Non contiene tutti i dettagli sul nome di dominio

- Authoritative DNS

- Tipicamente il server del possessore di dominio
- Fornisce l'ultima parola su una risposta DNS
- Massima affidabilità anche di prestazioni



Richiesta DNS

Screenshot of Wireshark showing a DNS query and response. The first two rows are highlighted with a red box.

No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.10.199	137.204.59.1	DNS	Standard query A deisnet.deis.unibo.it
2	0.001714	137.204.59.1	192.168.10.199	DNS	Standard query response CNAME deis85.deis.unibo.it A 137.204.57.85
3	0.002763	192.168.10.199	137.204.57.85	TCP	nim > http [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1
4	0.003487	137.204.57.85	192.168.10.199	TCP	http > nim [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM=1
5	0.003512	192.168.10.199	137.204.57.85	TCP	nim > http [ACK] Seq=1 Ack=1 Win=65535 Len=0
6	0.003732	192.168.10.199	137.204.57.85	HTTP	GET / HTTP/1.1
7	0.005748	137.204.57.85	192.168.10.199	TCP	http > nim [ACK] Seq=1 Ack=737 Win=6624 Len=0
8	0.009821	137.204.57.85	192.168.10.199	TCP	[TCP segment of a reassembled PDU]
9	0.011087	137.204.57.85	192.168.10.199	TCP	[TCP segment of a reassembled PDU]
10	0.011148	192.168.10.199	137.204.57.85	TCP	nim > http [ACK] Seq=737 Ack=2921 Win=65535 Len=0
11	0.014612	137.204.57.85	192.168.10.199	TCP	[TCP segment of a reassembled PDU]
12	0.015858	137.204.57.85	192.168.10.199	TCP	[TCP segment of a reassembled PDU]

Details pane:

- Frame 1: 81 bytes on wire (648 bits), 81 bytes captured (648 bits)
- Ethernet II, Src: DellComp_89:b3:e9 (00:06:5b:89:b3:e9), Dst: DellComp_ec:46:62 (00:b0:d0:ec:46:62)
- Internet Protocol, Src: 192.168.10.199 (192.168.10.199), Dst: 137.204.59.1 (137.204.59.1)
- User Datagram Protocol, Src Port: startron (1057), Dst Port: domain (53)
- Domain Name System (query)

Hex and ASCII panes show the DNS request and response frames.

Callout box content:

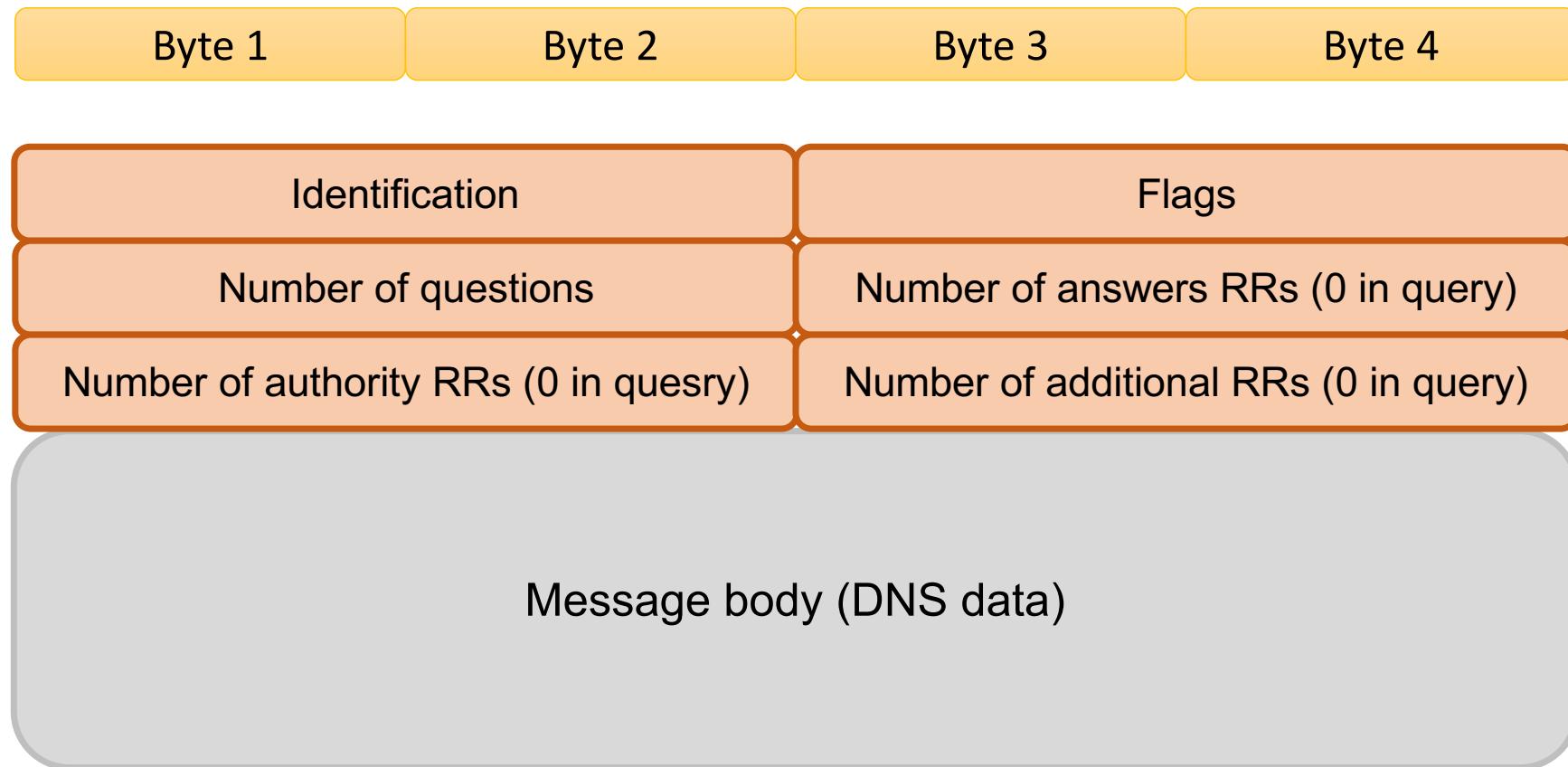
- Il protocollo DNS è un protocollo di livello applicativo per la rete
- Il DNS si posiziona allo stesso livello di HTTP
- Per il trasporto usa UDP (connectionless)

Le PDU DNS

- Il protocollo prevede due tipi di PDU
 - Query
 - Suddiviso in due sezioni
 - HEADER (PCI)
 - QUESTION (le domande al server DNS)
 - Response
 - Suddiviso in 5 sezioni
 - HEADER (PCI)
 - QUESTION (copia delle domande della query)
 - Records
 - Answer records
 - Authoritative records
 - Additional records

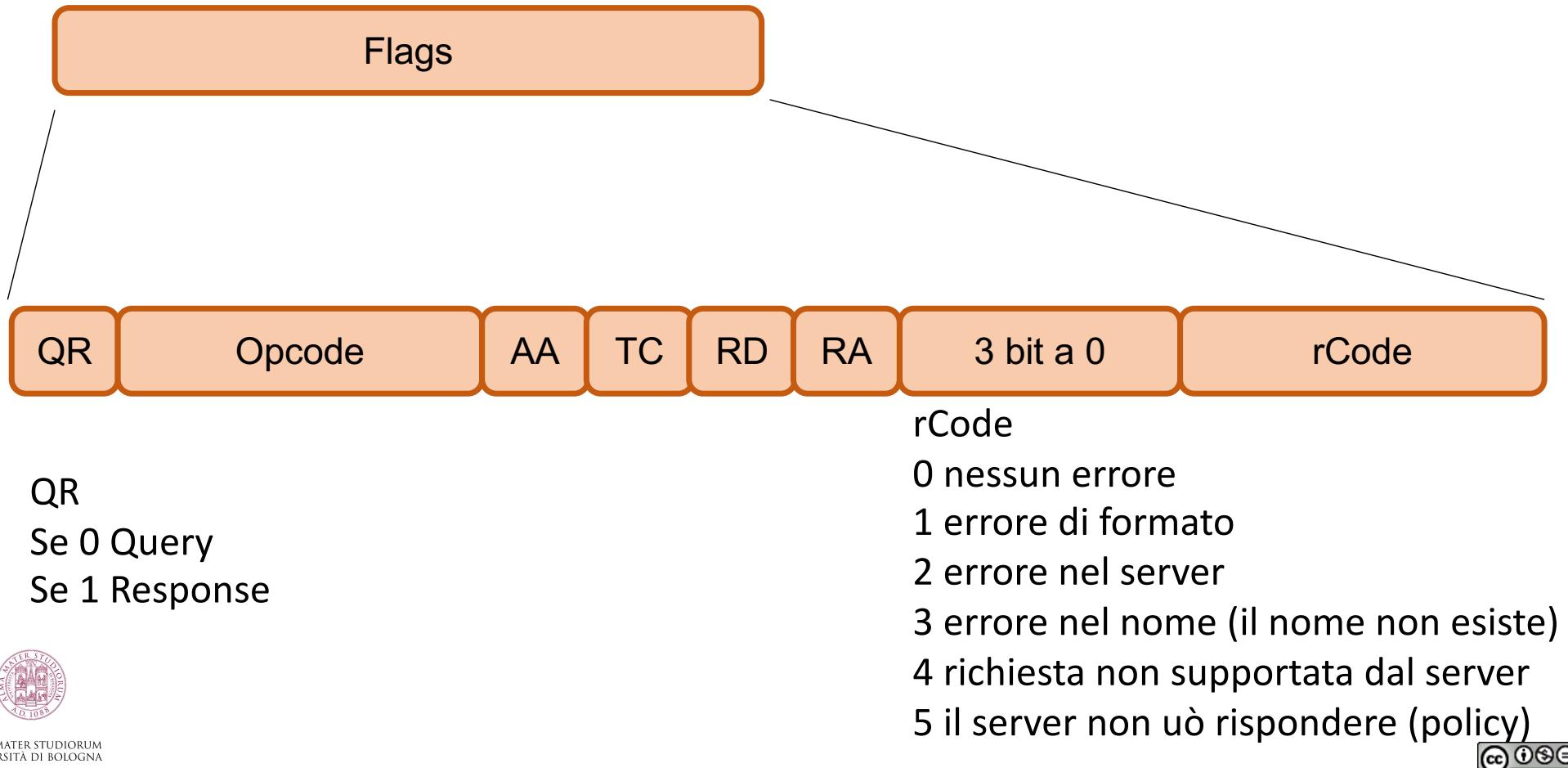


DNS PDU



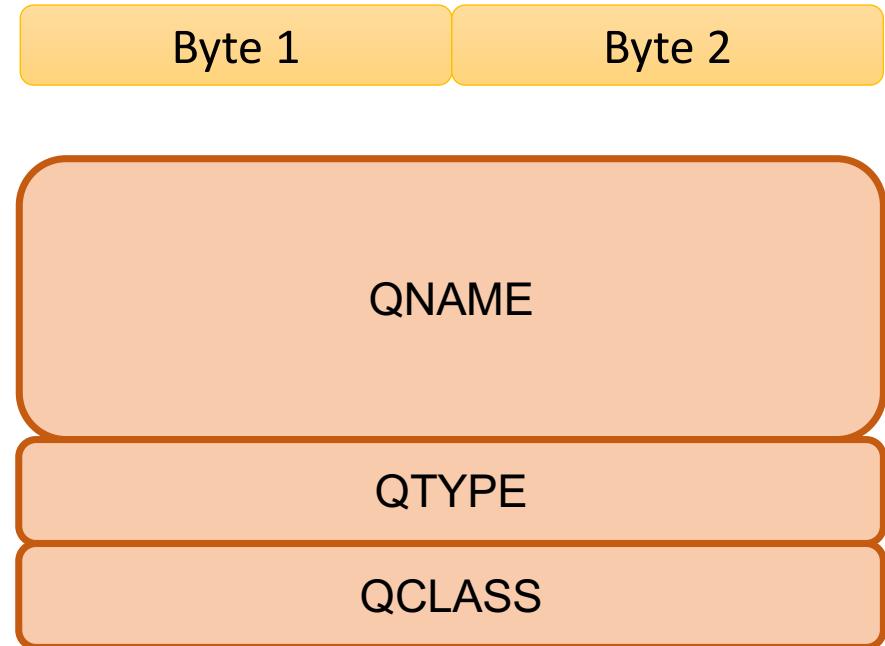
I campi delle PCI

- Flags (bandierine)
 - Singoli bit a valore binario
 - Piccoli gruppi di bit a più valori



Formato della domanda

- **QNAME**
 - Un nome di dominio per cui si effettua la richiesta
- **QTYPE**
 - Tipo della richiesta (codificato in due byte)
- **QCLASS**
 - Classe della domanda



Formato della risposta

- NAME

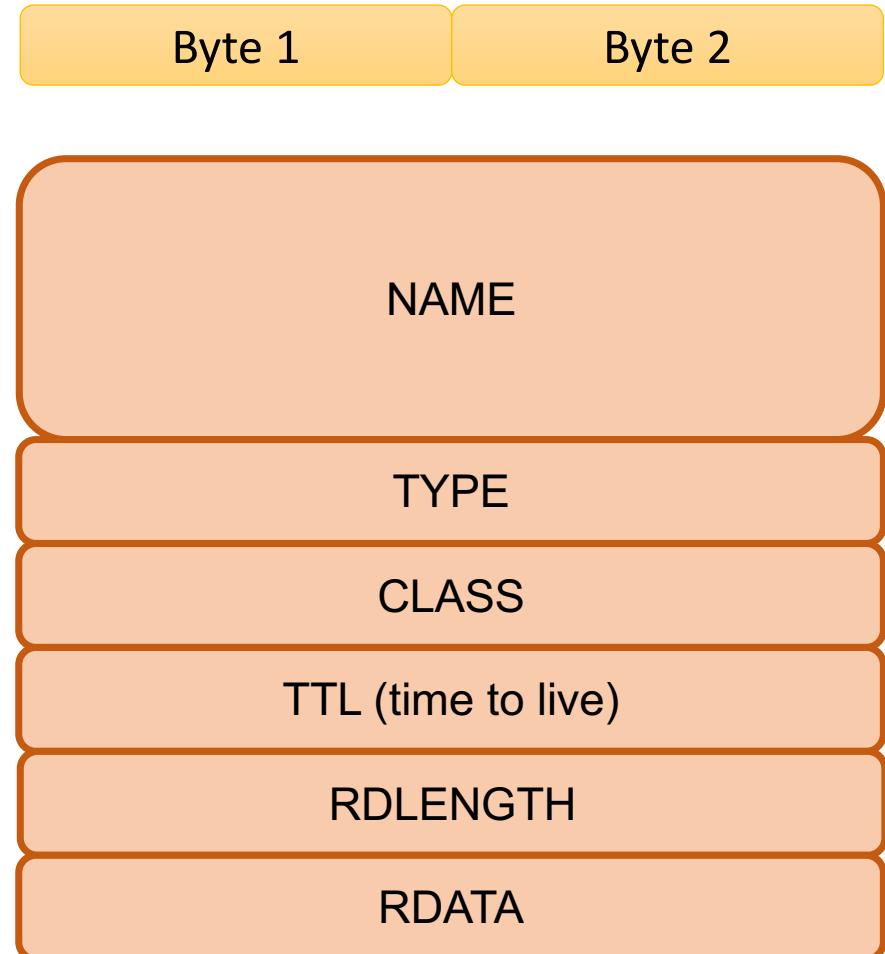
- Un nome di dominio per cui si effettua la richiesta

- TYPE

- Tipo della risposta
(significato del contenuto in RDATA)

- TTL

- Durata in secondi del tempo per il quale la risposta può essere mantenuta in memoria



I tipi di richiesta e risposta

- Richieste e risposte fanno riferimento ai record di risorsa (Resource Record o RR)
 - Formato RR: (name, value, type, ttl)
- Type
 - A
 - Restituisce un indirizzo IPv4 a 32 bit corrispondente ad un nome simbolico indicato nella richiesta
 - NS
 - Indica un server DNS autorevole per il nome di dominio inserito nella richiesta
 - CNAME
 - Permette di collegare un nome DNS ad un altro. La risoluzione continuerà con il nuovo nome indicato dal record CNAME.
 - MX
 - Collega un nome di dominio ad una lista di server di posta autorevoli per quel dominio





ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

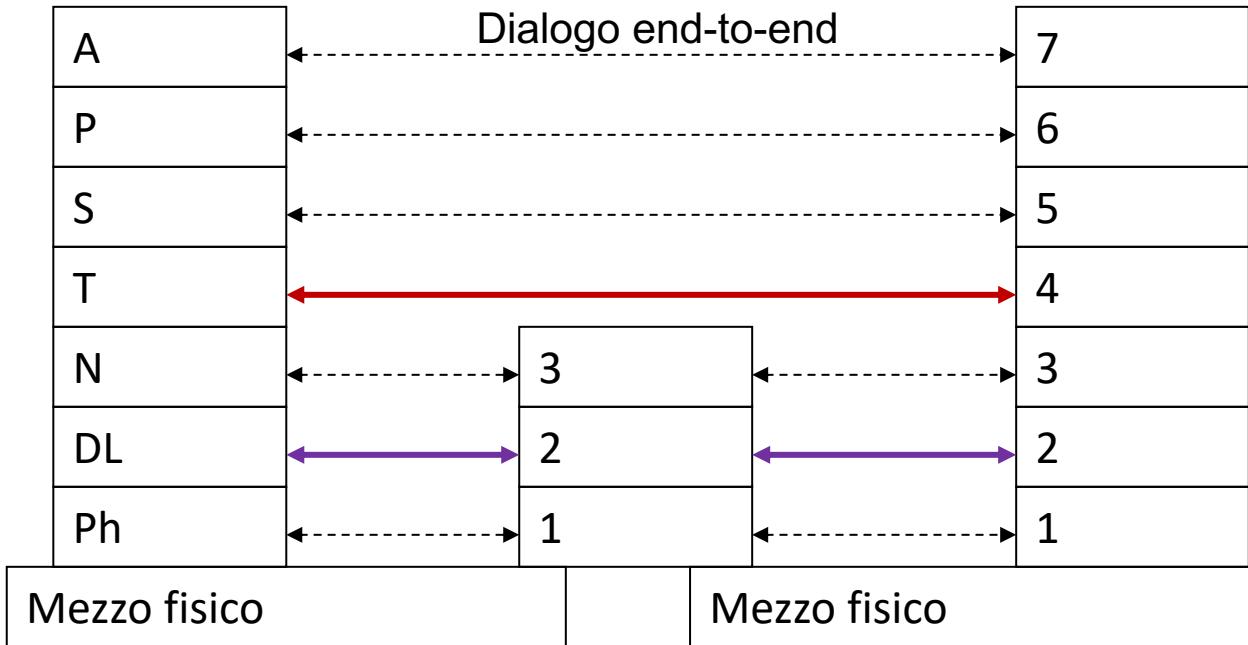
Controllo del canale

Franco CALLEGATI

Dipartimento di Informatica: Scienza e Ingegneria



Controllare il canale?





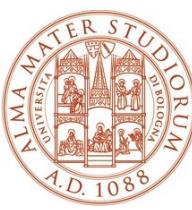
Canale di comunicazione

- Protocolli di linea
 - Canale sequenziale a banda costante di tipo punto-punto o punto-multipunto
 - Le trame arrivano nella stessa sequenza con cui sono inviate a meno degli errori
 - Tutte sperimentano ritardi di propagazione circa uguali
- Protocolli di trasporto
 - Canale non sequenziale a capacità variabile
 - Perdite di dati (errori di trasmissione, scarto nei nodi)
 - Duplicazione dei dati
 - Ritardi variabili
 - Arrivi fuori sequenza



Controllo del canale: strato 2

- I servizi di controllo del canale intendono
 - rendere **affidabile e sicuro** il servizio di collegamento che lo strato 2 offre alle entità di strato 3
- Le funzioni tipicamente svolte dallo strato 2 per il controllo del canale
 - **strutturazione** del flusso di dati
 - Le PDU di strato 2 sono dette **trame** o **frame**
 - **controllo e gestione degli errori** di trasmissione
 - **controllo di flusso**
 - **controllo di sequenza**
 - gestire il **protocollo di accesso** per un collegamento **punto-multipunto**
- Non tutti i protocolli di strato 2 svolgono tutte queste funzioni, alcuni implementano solo dei sottoinsiemi



Problematiche di Sincronismo

- Nelle trasmissioni numeriche per riconoscere i bit in ricezione occorre determinare gli **istanti di campionamento** per ricostruire il **sincronismo di cifra**
- È un problema dello strato 1 che ha dei riflessi sullo strato 2
- Un circuito nel ricevitore estrae il segnale di sincronismo ma ha bisogno di **agganciarsi**
- Sono possibili due modalità
 - Il canale può essere tenuto **sempre pieno di bit**
 - L'aggancio avviene in fase di inizializzazione e viene poi sempre mantenuto
 - Il protocollo di linea deve garantire la presenza di segnale anche quando non ha dati da trasmettere
 - Il canale può avere **momenti di vuoto di segnale**
 - All'inizio di ogni nuova trasmissione deve essere inserito un **preambolo di sincronismo**



Il sincronismo di trama

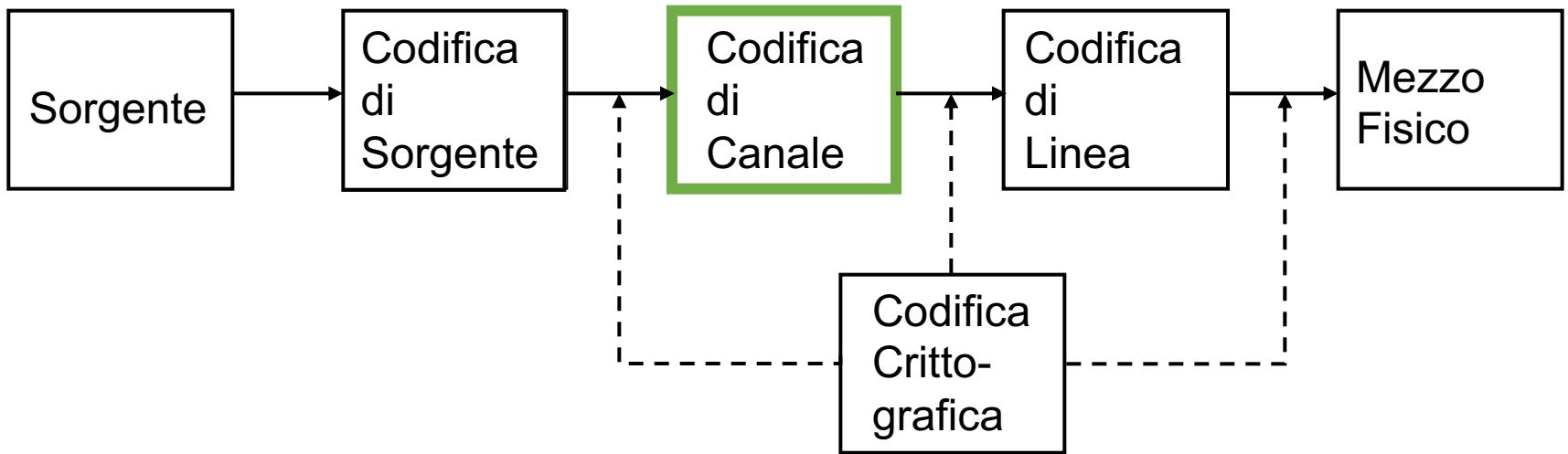
- Il sincronismo di cifra garantisce la corretta lettura dei singoli bit
- Rimane il problema di distinguere le PDU una dall'altra
- Si deve garantire il **sincronismo di trama**
 - Protocolli asincroni a livello di trama
 - Le trame possono iniziare e finire in ogni istante
 - Informazioni aggiuntive (nel PCI) vengono usate per riconoscere correttamente inizio e fine delle trame
 - Protocolli sincroni a livello di trama
 - Le trame devono iniziare e terminare in istanti predefiniti
 - Non sono necessarie PCI per il sincronismo



Garantire affidabilità

- Come garantire affidabilità? Prima di consegnare i dati allo strato superiore si controllano
 - Errori di trasmissione
 - Codifica di canale con codici a rivelazione di errore
 - Conferma di ricezione e ritrasmissione
 - Sequenzialità dei dati
 - Numerazione delle unità informative
 - Conferma di ricezione e ritrasmissione
 - Flusso dei dati
 - Finestra scorrevole
 - Conferma dei dati

La codifica di canale



- A ognuno di questi blocchi corrisponde una **Decodifica** in ricezione
- Le operazioni di codifica possono essere combinate in vari modi (canale/linea, sorgente/canale, ...)
- La crittografia può essere inserita in diversi punti e in diversi strati dell' architettura OSI



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Controllo dell'errore



Alcune definizioni

- Codici a blocco
 - Si applica la codifica a blocchi di k bit di informazione
 - Vengono calcolati r bit di ridondanza come *funzione combinatoria* dei suddetti k bit e trasmessi $n=k+r$ bit
- Codici convoluzionali
 - Vengono calcolati r bit di ridondanza ogni k di informazione mediante reti logiche sequenziali
 - Nel calcolo dei bit di ridondanza si tiene conto dei k bit di informazione e di *variabili di stato* dipendenti dalle operazioni passate
- Faremo riferimento d'ora in poi solamente a *codici a blocco*



Gestione dell'errore : la codifica

- Nella codifica a blocchi
 - k bit vengono codificati in una parola di n bit aggiungendo $r=n-k$ bit
 - Sono disponibili 2^n parole di codice per trasportare 2^k messaggi
 - 2^k sono parole di codice ammesse (valide)
 - $2^n - 2^k$ sono parole di codice non ammesse (invalidi)
- Codici a rivelazione di errore
 - La ricezione di una parola di codice invalida indica la presenza di errori di trasmissione
 - Non si può dire quali siano i bit errati
 - Per garantire la trasparenza semantica è necessaria la **ritrasmissione** dei dati errati
- Codici a correzione di errore
 - Una parola di codice invalida
 - Indica la presenza di errori di trasmissione
 - Permette di individuare la parola di codice valida corrispondente (identifica gli errori)
 - Garantisce la trasparenza semantica in tutti i casi in cui l' errore è correggibile



Codifica di canale: correzione o rivelazione?

- PROBLEMA

- Trasmissione di 1 Mbit di dati in trame lunghe 1000 bit

- Codice a correzione di errore richiede 10 bit aggiuntivi per trama
 - Codice a rivelazione richiede 1 solo bit per trama
 - Alla rivelazione dell'errore fa seguito la ritrasmissione

- Caso 1: tasso di errore per bit del canale pari a 10^{-6}

- In media un errore ogni 1000 trame: bit aggiuntivi
 - 10000 bit nel caso a correzione,
 - $1000+1001=2001$ nel caso a rivelazione

Conviene la rivelazione

- Caso 2: tasso di errore per bit del canale pari a 10^{-5}

- In media un errore ogni 100 trame: bit aggiuntivi
 - 10000 bit nel caso a correzione,
 - $1000+10*1001=11010$ nel caso a rivelazione.

Circa equivalente

- Caso 3: tasso di errore per bit del canale pari a 10^{-4}

- In media un errore ogni 10 trame: bit aggiuntivi
 - 10000 bit nel caso a correzione,
 - $1000+111*1001=112111$ nel caso a rivelazione.

Conviene la correzione



In generale

- **Correzione di errore** (anche forward error correction o FEC)
 - Richiede un numero abbastanza elevato di bit aggiuntivi
 - Permette la correzione dei dati errati in base ai soli dati ricevuti
- **Rivelazione d'errore**
 - Richiede un numero limitato di bit aggiuntivi
 - Rende necessaria la *ritrasmissione* dei dati errati
- In linea con l'esempio precedente
 - Conviene la rivelazione se il canale è affidabile per cui ci sono pochi errori
 - Conviene la correzione se il canale produce molti errori di trasmissione
- Nelle reti di solito
 - Si usano **codici a correzione di errore nello strato fisico**
 - Si usa la **rivelazione di errore nei protocolli di linea e di trasporto**



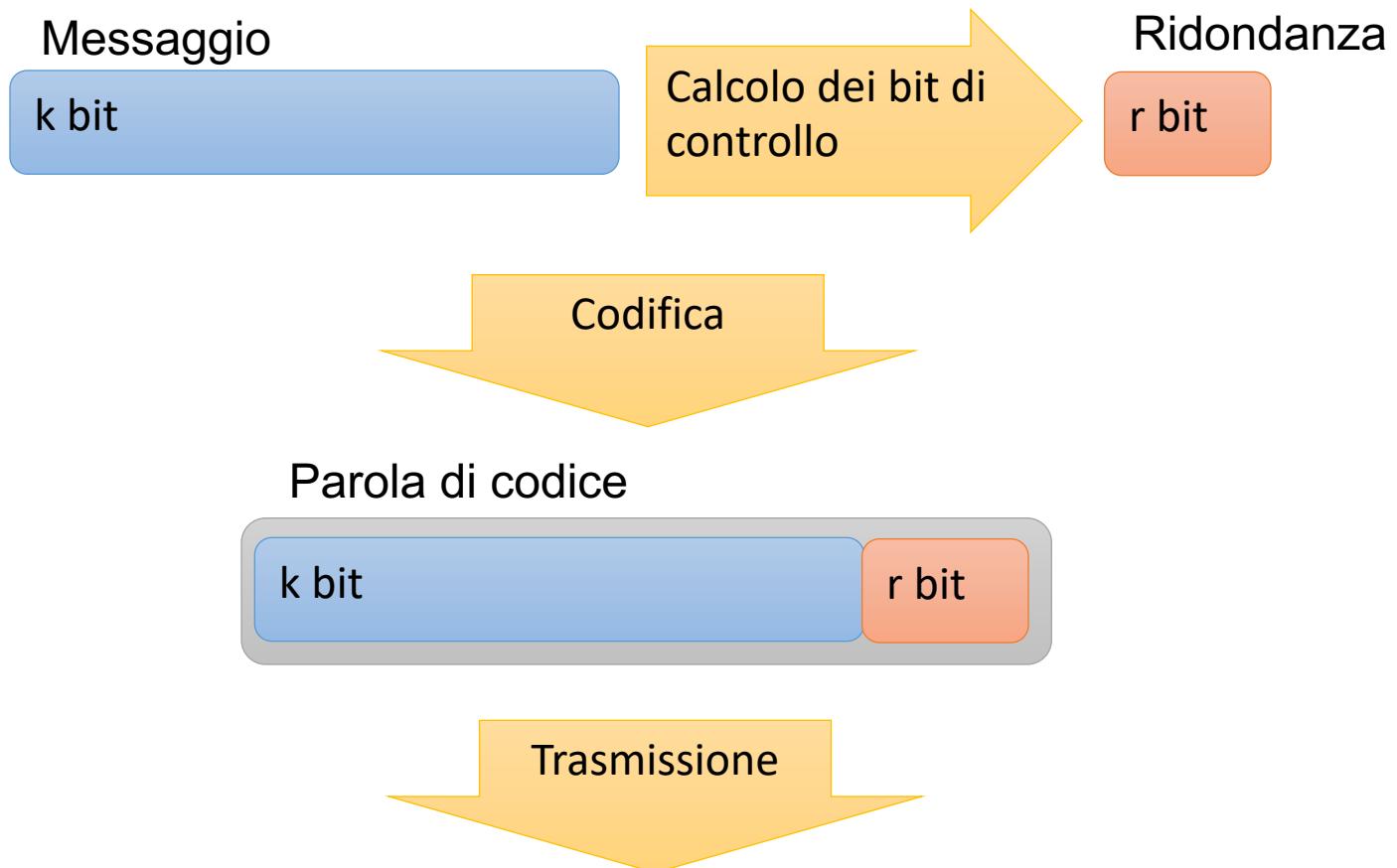
Definizioni

- Codici **lineari**
 - Dati due messaggi di k bit m_1 e m_2
 - Ricavate le parole di codice c_1 e c_2
 - Il codice si dice lineare se
$$m_3 = m_1 + m_2 \text{ da origine a } c_3 = c_1 + c_2$$
- Codificatori **sistematici**
 - nella sequenza di n bit da trasmettere i k bit di informazione, mantenuti distinti dagli r bit di ridondanza, vengono trasmessi inalterati



Uso del codice: in trasmissione

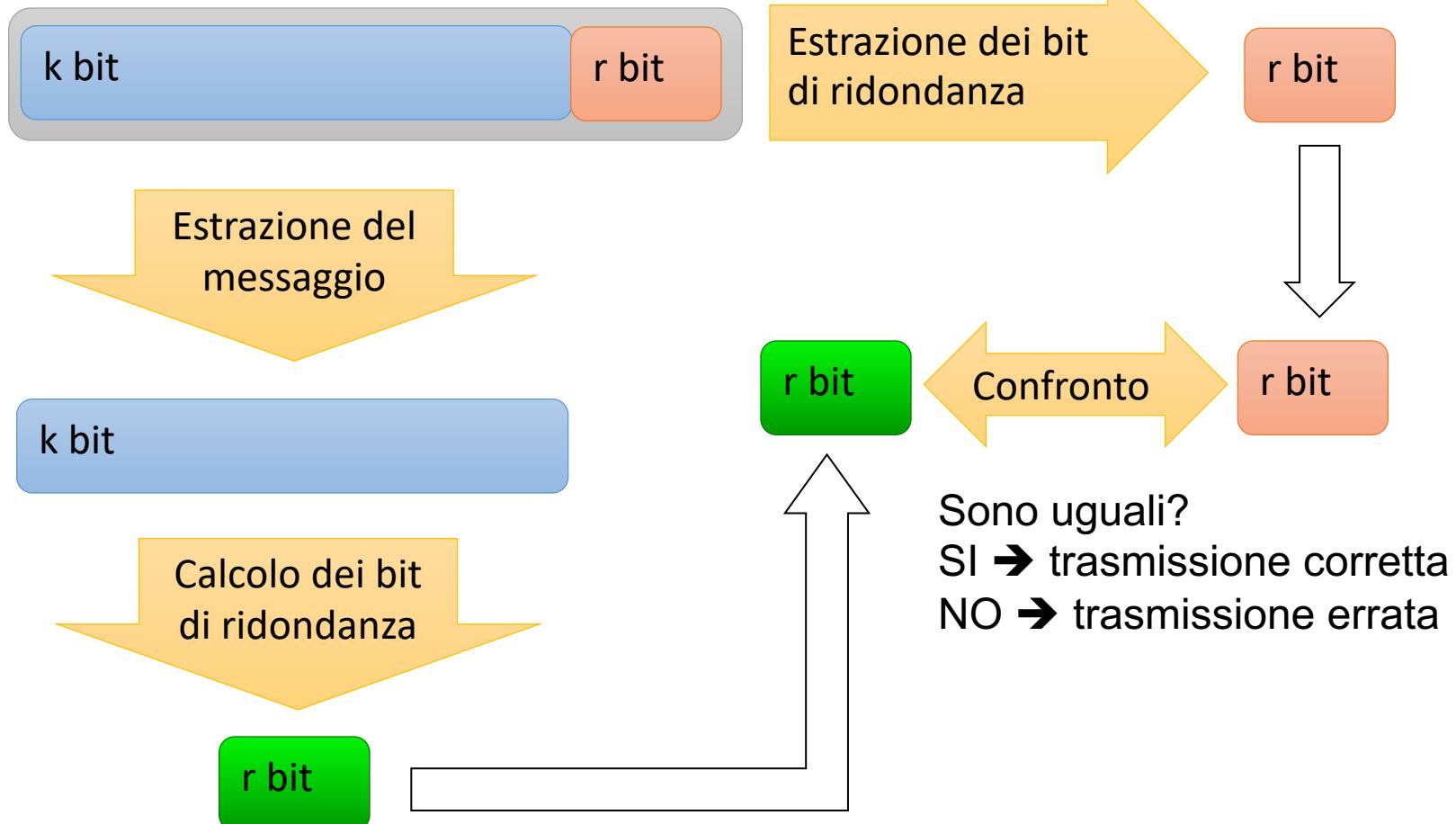
- Rivelazione d'errore
 - Codice a blocco sistematico





Uso del codice: in ricezione

Parola di codice





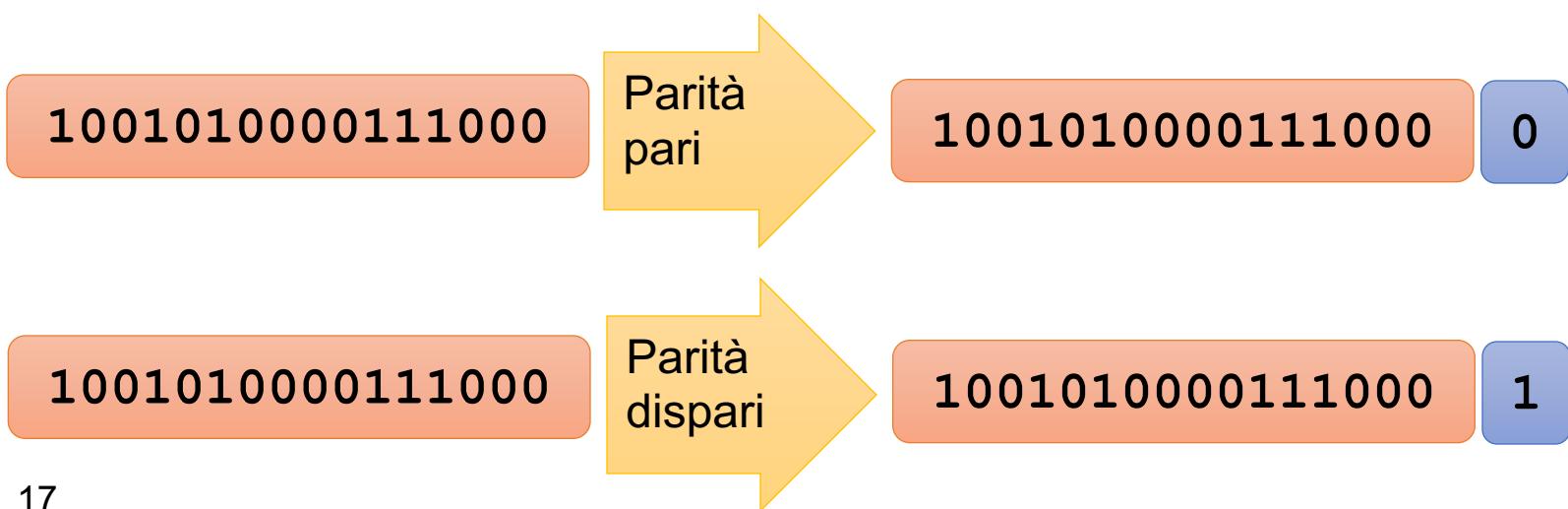
Il bit di parità

- Dati k bit di informazione b_0, b_1, \dots, b_{k-1}

$$b_k = b_0 \oplus b_1 \oplus b_2 \oplus \dots \oplus b_{k-1} : \text{parità pari}$$

$$b_k = \text{NOT} [b_0 \oplus b_1 \oplus \dots \oplus b_{k-1}] : \text{parità dispari}$$

- Dove \oplus è l'operazione di OR esclusivo
- $r = 1$ un solo bit di ridondanza per qualunque dimensione del blocco dati k





Proprietà

- Rivela sempre un numero dispari di errori
- Fallisce con un numero pari di errori

Da trasmettere

1001010000111000

Trasmesso

1001010000111000

Errore singolo

Ricevuto

100101 $\textcolor{red}{1}$ 000111000

Calcolo del bit
di parità

1 \neq 0

RICEZIONE ERRATA

Errore doppio

Ricevuto

100101 $\textcolor{red}{1}$ 000 $\textcolor{red}{0}$ 111000

Calcolo del bit
di parità

0 = 0

RICEZIONE CORRETTA
RIVELAZIONE ERRATA!



Internet checksum

- Nei protocolli di Internet vengono solitamente utilizzati codici a blocchi sistematici
- Sono estensioni del bit di parità, volte ad estenderne le prestazioni
- Si applica su parole di 16 bit, indipendente dalla lunghezza complessiva del blocco dati

RFC1071 - 1988

In outline, the Internet checksum algorithm is very simple:

(1) Adjacent octets to be checksummed are paired to form 16-bit integers, and the 1's complement sum of these 16-bit integers is formed.

(2) To generate a checksum, the checksum field itself is cleared, the 16-bit 1's complement sum is computed over the octets concerned, and the 1's complement of this sum is placed in the checksum field.

(3) To check a checksum, the 1's complement sum is computed over the same set of octets, including the checksum field. If the result is all 1 bits (-0 in 1's complement arithmetic), the check succeeds.



Somma complemento a 1

- La somma complemento a 1 è simile al calcolo binario intero senza segno (somma complemento a 2) ma differisce per l'uso dei riporti
- Se una somma genera un riporto questo viene aggiunto al risultato

Somma complemento a 1

$$\begin{array}{r} 11110010 \\ + 11110100 \\ \hline 111100110 \\ \hline \end{array}$$

1

11100111



Proprietà

- Blocco dati fatto di byte A, B, C, D, E, F, G, ...
- Parole di 16 bit [A,B], [C,D], [E,F], [G,H]
- Proprietà commutativa e associativa
 - $[A,B] + [C,D] = [C,D] + [A,B]$
 - $([A,B] + [C,D]) + [E,F] = [A,B] + ([C,D] + [E,F])$
- Indipendenza dall'ordine dei byte
 - $[A,B] + [C,D] = [X,Y]$ allora $[B,A] + [D,C] = [Y,X]$
 - Questa proprietà è molto importante perché rende il calcolo indipendente dalla rappresentazione del numero a livello di sistema hardware "big-endian" o "little-endian"



Esempio

- Devo calcolare il checksum di 64 bit, raggruppabili in 4 parole da 16 bit

00000000 00000001
11110010 00000011
11110100 11110101
11110110 11110111

Calcolo
checksum

Creo la parola di codice

00000000 00000001
11110010 00000011
11110100 11110101
11110110 11110111
00100010 00001101

00100010
00001101

00000000 00000001
11110010 00000011

11110010 00000100
11110100 11110101

111100110 11111001
11110110 11110111
1

111011100 111110001
11011100 11110001
1 1

11011101 11110010



In ricezione

00000000	00000001
11110010	00000011
11110100	11110101
11110110	11110111
00100010	00001101

Eseguo la somma

00000000	00000001
11110010	00000011

11110010	00000100
11110100	11110101

111100110	11111001
11110110	11110111

10 11011100	11111000
00100010	00001101

10 11111110	111111101
--------------------	------------------

11111110	111111101
1	10

11111111	11111111
-----------------	-----------------

Checksum giusto
Ricezione corretta



Errore

00000000	00000001
11110010	00000011
11110100	11110101
11110110	111 01 111
00100010	00001101

Eseguo la somma

00000000	00000001
11110010	00000011

11110010	00000100
11110100	11110101

111100110	11111001
11110110	11101111

10 11011100	11111100
00100010	00001101

10 11111110	1000000101
--------------------	-------------------

11111110	000000101
10	10

100000000	00000111
------------------	-----------------

Checksum inesatto
Ricezione errata



Algebra binaria e codici polinomiali

- L'algebra si costruisce sull'insieme delle cifre binarie 0 e 1
 - $\mathbb{A} = \{0, 1\}$
- Operazioni:
 - Or esclusivo \oplus (somma e sottrazione)
 - $a \oplus (b \oplus c) = (a \oplus b) \oplus c$ infatti ad esempio $1 \oplus (1 \oplus 0) = (1 \oplus 1) \oplus 0 = 0$
 - Ha elemento neutro ed opposto
 - 0 elemento neutro
 - 1 oppostoinfatti $1 \oplus 0 = 1$ e $0 \oplus 0 = 0$
 - Vale la proprietà commutativa
 - Infatti $1 \oplus 0 = 0 \oplus 1 = 1$
 - Genera un gruppo abeliano
 - Moltiplicazione
 - Esiste l'elemento neutro
 - 1 infatti $0 \times 1 = 0$ e $1 \times 1 = 1$
 - Vale la proprietà commutativa
 - Vale la proprietà distributivainfatti $1 \times 0 = 0 \times 1 = 0$
infatti $1 \times (1 \oplus 0) = 1 \times 1 \oplus 1 \times 0 = 1$
 - Si genera un anello
- Ne risulta un'algebra analoga a quella ordinaria ma limitata alle cifre binarie
- L'insieme polinomi con variabile x e coefficienti in un anello formano un anello ed ereditano le operazioni e le loro proprietà



Codici polinomiali

- Basati sull'uso di polinomi in un'algebra binaria
 - k bit vengono posti in corrispondenza con un polinomio di grado $k-1$ nella variabile binaria x :
$$P_{k-1}(x) = b_0 + b_1x + b_2x^2 + \dots + b_{k-1}x^{k-1}$$
- Vengono calcolati i bit di ridondanza utilizzando operazioni sui polinomi
- Polinomio generatore
 - Viene stabilito un polinomio di grado r noto a trasmettitore e ricevitore
 - $G_r(x)$ determina le proprietà di rivelazione del codice



Polinomio trasmesso

- Per calcolare polinomio $T_{n-1}(x)$ da trasmettere:
 - Si moltiplica il polinomio $P_{k-1}(x)$ per x^r
 - r bit a zero posti in coda
 - Si esegue la divisione polinomiale fra $P_{k-1}(x) x^r$ e $G_r(x)$ ottenendo un quoziente ed un resto

$$P_{k-1}(x) x^r = G_r(x) Q_{k-1}(x) \oplus R_{r-1}(x)$$

- Notando che nell'algebra adottata somma e sottrazione coincidono, si trasmette

$$T_{n-1}(x) = P_{k-1}(x) x^r \oplus R_{r-1}(x) = G_r(x) Q_{k-1}(x)$$

- Proprietà di $T_{n-1}(x)$
 - Realizza una codifica di tipo sistematico perché i bit di resto, al più r bit, vanno a sovrapporsi agli r zeri in coda
 - È multiplo di $G_r(x)$



Polinomio ricevuto

- Il ricevitore riceve una sequenza di n bit che corrisponde al polinomio ricevuto

$$T'_{n-1}(x)$$

- Se si verifica un errore di trasmissione

$$T'_{n-1}(x) \neq T_{n-1}(x)$$

- Esisterà un polinomio $E(x)$ tale che

$$T'_{n-1}(x) = T_{n-1}(x) + E(x)$$

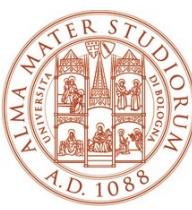
- $E(x)$

- ha coefficienti non nulli in corrispondenza dei bit in cui $T'_{n-1}(x)$ differisce da $T_{n-1}(x)$
- rappresenta in forma polinomiale gli eventuali errori e per questo si dice *polinomio errore*



Rivelazione dell' errore

- Il ricevitore esegue la divisione:
$$T'_{n-1}(x)/G_r(x) = (T_{n-1}(x) + E(x))/G_r(x) = T_{n-1}(x)/G_r(x) + E(x)/G_r(x)$$
- Il primo di questi termini ha sempre resto 0
- Se $E(x) \neq 0$ e $E(x)$ non è divisibile per $G_r(x)$, allora il resto della divisione precedente risulta diverso da 0 e viene quindi rilevato l'errore
- Per rilevare gli errori si deve quindi evitare che:
$$\text{Resto}[E(x)/G_r(x)] = 0$$
- $G_r(x)$ va scelto per minimizzare la probabilità di non rivelare un errore
 - affinché due polinomi siano divisibili è comunque necessario che il grado del numeratore sia maggiore o uguale al grado del denominatore



Capacità del codice e scelta di $G_r(x)$

- Un singolo errore
 - $E(x) = x^i$
 - è sufficiente che in $G_r(x)$ vi siano almeno due bit a 1
- Numero dispari di errori
 - Se $G_r(x)$ è multiplo di $(1+x)$ non divide mai un polinomio con numero dispari di termini
 - Se si sceglie $G_r(x) = 1+x$, il codice polinomiale fornisce 1 singolo bit di ridondanza eguale al bit di parità
- 2 errori
 - $E(x) = x^i + x^j = x^j(x^h + 1)$. Esistono diversi polinomi che non dividono mai $(x^h + 1)$. ITU ha proposto il seguente polinomio :
$$G_{16}(x) = x^{16} + x^{12} + x^5 + 1$$



Errori a burst

- Nelle reti di telecomunicazione è frequente una distribuzione non uniforme degli errori, con concentrazione degli stessi in certi intervalli
 - filotto (burst) di bit lungo k cui bit intermedi sono inaffidabili (supponiamo abbiano una probabilità di essere errati pari al 50%)
 - rappresentato da un polinomio di grado $k - 1$
- Si possono avere i seguenti casi:
 - $k - 1 < r$: l' errore viene sempre rilevato;
 - $k - 1 = r$: si ha resto nullo se $E(x) = G_r(x)$
 - questo evento può verificarsi con probabilità = $1/2^{r-1}$
 - $k - 1 > r$: il resto ha valore casuale e l' errore sfugge se il resto è nullo (r bit a 0)
 - questo evento può con probabilità = $1/2^r$



Esercizo 1- (a)

- Si vuole trasmettere da S a D la seguente stringa di bit utilizzando un codice polinomiale per verificare che la trasmissione avvenga senza errori.
- Informazione Iniziale: 101101000101
- Il polinomio di grado $k-1$, $P_{k-1}(x)$ nella variabile binaria x è il seguente:

$$P(x) = x^{11} + 0 + x^9 + x^8 + 0 + x^6 + 0 + 0 + 0 + x^2 + 0 + 1$$

- Il polinomio generatore è il seguente:

$$G(x) = x^2 + x + 1$$

Quindi $r = 2$



Esercizio 1- (b)

- Determiniamo ora gli n bit realmente trasmessi e verifichiamo la resistenza agli errori di trasmissione che si ottiene attraverso il metodo dei codici polinomiali.
- $G(x)$ è un polinomio di secondo grado per cui occorre moltiplicare il polinomio $P(x)$ per x^2 :

$$P(x)x^2 = x^{13} + 0 + x^{11} + x^{10} + 0 + x^8 + 0 + 0 + 0 + x^4 + 0 + x^2 + 0 + 0$$

- Si devono calcolare gli n bit che verranno trasmessi



Esercizio 1- (c)

- Si divide $P(x) / x^2$ per $G(x)$, per ottenere $T_{n-1}(x)$:

$$x^{13} + 0 + x^{11} + x^{10} + 0 + x^8 + 0 + 0 + 0 + x^4 + 0 + x^2 + 0 + 0$$

$$x^{13} + x^{12} + x^{11}$$

$$/ \quad x^{12} + 0 + x^{10} + 0 + x^8 + 0 + 0 + 0 + x^4 + 0 + x^2 + 0 + 0$$

$$x^{12} + x^{11} + x^{10}$$

$$/ \quad x^{11} + 0 + 0 + x^8 + 0 + 0 + 0 + x^4 + 0 + x^2 + 0 + 0$$

$$x^{11} + x^{10} + x^9$$

$$/ \quad x^{10} + x^9 + x^8 + 0 + 0 + 0 + 0 + x^4 + 0 + x^2 + 0 + 0$$

$$x^{10} + x^9 + x^8$$

$$/ \quad / \quad / \quad x^4 + 0 + x^2 + 0 + 0$$

$$x^4 + x^3 + x^2$$

$$/ \quad x^3 + 0 + 0 + 0$$

$$x^3 + x^2 + x$$

$$/ \quad x^2 + x + 0$$

$$x^2 + x + 1$$

$$/ \quad / \quad 1$$

$$\begin{array}{r} x^2 + x + 1 \\ \hline x^{11} + x^{10} + x^9 + x^8 + x^2 + x + 1 \end{array}$$



Esercizio 1- (d)

- Dalla divisione si ottiene:
 - $R(x) = 1$
 - $Q(x) = x^{11} + x^{10} + x^9 + x^8 + x^2 + x + 1$
- quindi il polinomio da trasmettere è
 - $T_{n-1}(x) = x^{13} + 0 + x^{11} + x^{10} + 0 + x^8 + 0 + 0 + 0 + x^4 + 0 + x^2 + 0 + 1$
- In conclusione la sequenza di bit da trasmettere è dunque:

10110100010101



Esercizio 1- (e)

- Il polinomio ricevuto alla destinazione è dato da:
 - $T'_{n-1}(x) = T_{n-1}(x) + E(x)$;
 - $E(x)$ rappresenta il polinomio errore.
 - $E(x)$ ha coefficienti diversi da 0 in corrispondenza dei bit $T_{n-1}(x)$ che vengono corrotti dall'errore.
- Il ricevitore verifica la correttezza dei dati ricevuti eseguendo la seguente divisione:

$$\frac{T'_{n-1}(x)}{G(x)} = \frac{T_{n-1}(x) + E(x)}{G(x)}$$



Esercizio 1- (f)

- Consideriamo i seguenti $E(x)$
 - $E(x)=x^9+x^8$
 - $E(x)=x^4+x^3+x^2$
- Considerando il primo polinomio $E(x)$:
 - $T_{n-1}(x) = x^{13}+0+x^{11}+x^{10}+0+x^8+0+0+0+x^4+0+x^2+0+1$
 - $T'_{n-1}(x) = T_{n-1}(x)+E(x) = x^{13}+0+x^{11}+x^{10}+\textcolor{orange}{x}$
 $\textcolor{orange}{9}+\textcolor{orange}{0}+0+0+0+x^4+0+x^2+0+1$
 - $T'_{n-1}(x) = \boxed{1011\textcolor{orange}{10}000101}\boxed{01}$
- Per verificare se l' errore viene rilevato si deve dividere $T'_{n-1}(x)$ per $G(x) = x^2+x+1$



Esercizio 1- (g)

- $T_{n-1}^1(x)$ diviso $G(x)$:

$$x^{13} + 0 + x^{11} + x^{10} + x^9 + 0 + 0 + 0 + 0 + x^4 + 0 + x^2 + 0 + 1$$

$$x^{13} + x^{12} + x^{11}$$

$$/ \quad x^{12} + 0 + x^{10}$$

$$x^{12} + x^{11} + x^{10}$$

$$/ \quad x^{11} + 0 + x^9$$

$$x^{11} + x^{10} + x^9$$

$$/ \quad x^{10} + 0 + 0$$

$$x^{10} + x^9 + x^8$$

$$/ \quad x^9 + x^8 + 0$$

$$x^9 + x^8 + x^7$$

$$/ \quad / \quad x^7 + 0 + 0$$

$$x^7 + x^6 + x^5$$

$$/ \quad x^6 + x^5 + x^4$$

$$x^6 + x^5 + x^4$$

$$/ \quad / \quad / \quad 0 + x^2 + 0 + 1$$

$$x^2 + x + 1$$

$$/ + x + 0$$

$$x^2 + x + 1$$

$$x^{11} + x^{10} + x^9 + x^8 + x^7 + x^5 + x^4 + 1$$



Esercizio 1- (h)

- Eseguiamo la divisione per il secondo caso:

- $E(x) = x^4 + x^3 + x^2$

$$x^{13} + 0 + x^{11} + x^{10} + 0 + x^8 + 0 + 0 + 0 + x^3 + 0 + 0 + 1$$

$$x^{13} + x^{12} + x^{11}$$

$$/ \quad x^{12} + 0 + x^{10}$$

$$x^{12} + x^{11} + x^{10}$$

$$/ \quad x^{11} + 0 + 0$$

$$x^{11} + x^{10} + x^9$$

$$/ \quad x^{10} + x^9 + x^8$$

$$x^{10} + x^9 + x^8$$

$$/ \quad / \quad /$$

$$x^2 + x + 1$$

$$x^{11} + x^{10} + x^9 + x^8 + x + 1$$

$$x^3 + 0 + 0$$

$$x^3 + x^2 + x$$

$$/ \quad x^2 + x + 1$$

$$x^2 + x + 1$$

$$/ \quad / \quad /$$

- In questo caso l' errore **NON VIENE RIVELATO**



Authomatic Repeat Request

- I protocolli ARQ vengono utilizzati nello strato di linea ed in quello di trasporto in sinergia con una codifica a rivelazione di errore
- Obiettivo:
 - Rendere affidabile il canale di comunicazione
 - Affidabile?
 - Identifica errori di trasmissione e innesca la ritrasmissione
 - Riconosce perdita di informazioni
 - Riconosce perdite di sequenza
- Il canale tipicamente è:
 - Singolo collegamento seriale nello strato di linea
 - Flusso seriale di bit
 - Connessione end-to-end nello strato di trasporto
 - Cascata di nodi e collegamenti con diverse caratteristiche e prestazioni
- La diversità del canale rende le problematiche dei protocolli di trasporto più complesse ma esistono molti elementi in comune



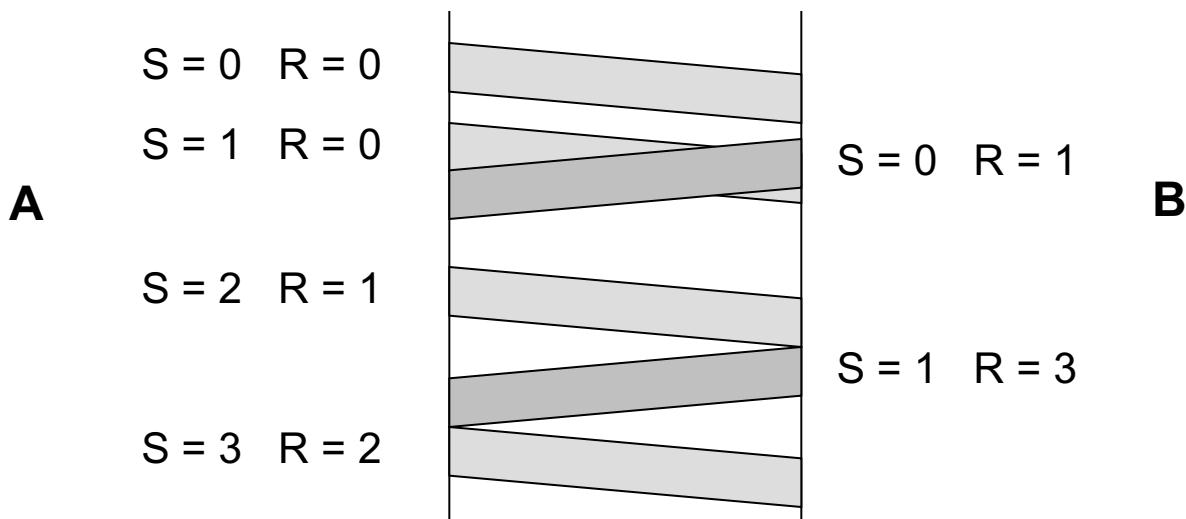
Controllo degli errori

- Alle PDU viene applicata una codifica di canale
- Il ricevitore
 - Verifica la correttezza delle PDU ricevute grazie a rivelazione di errore
 - Ignora le PDU errate
 - Può far partire le **procedure di ritrasmissione**
- Il trasmettitore
 - Ritrasmette le trame non correttamente ricevute
 - Su indicazione del ricevitore
 - Alla scadenza del time-out



Numerazione

- I protocolli ARQ numerano sequenzialmente le unità informative (UI) da consegnare ai protocolli superiori
- Cosa numerare?
 - PDU
 - Unità informative standard (bit, byte ...)
- Trasmettitore e ricevitore mantengono due contatori:
 - **S** conta in modo sequenziale le unità informative **inviate**
 - **R** conta le unità informative **ricevute** in modo corretto
- S permette il “posizionamento” nel flusso
- R permette la confermare di ricezione





Conferma (Acknowledge)

- La corretta ricezione viene confermata dal ricevitore inviando al trasmettitore il proprio valore di **R**
 - Le PDU ricevute in modo corretto fanno aumentare R
 - Quando una PDU viene ricevuta in modo non corretto *viene ignorata ed R non viene modificato*
- La conferma della corretta ricezione può essere
 - **Esplicita**
 - Ogni PDU ricevuta correttamente genera una conferma
 - **Implicita** (cumulativa)
 - Una PDU di conferma con **R = n** conferma la ricezione fino a **n-1**
 - In **piggybacking**
 - Viaggia inserita (a “cavalluccio”) in una PDU contenente dati utili



Gli ACK

- Gli acknowledge o ACK
 - Sono PDU specializzate che non portano dati di utente ma solamente informazioni di controllo per il protocollo
- Servono qualora
 - Il protocollo ARQ non possa usare il piggybacking
 - Il ricevitore non abbia dati da trasmettere
- Non è necessario numerare gli ACK
 - I protocolli ARQ tipicamente
 - confermano la ricezione delle PDU che portano dati d' utente
 - non confermano la ricezione degli ACK (conferma della conferma)
 - Non si ritiene necessario controllare la sequenza degli ACK

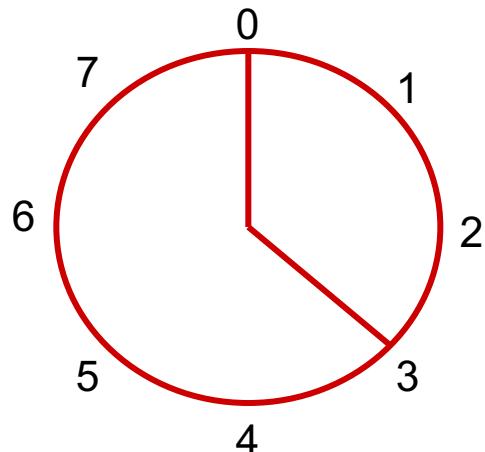


Finestra scorrevole

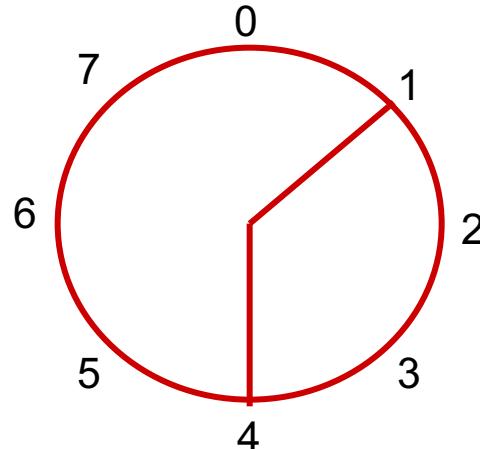
- Le funzioni di controllo
 - Dell'errore
 - Di flusso
 - Di sequenza
- Possono essere implementate con l'uso sinergico di
 - Codici di canale
 - Numerazione delle unità informative
 - Conferma di ricezione
- Il meccanismo utilizzato è quello della trasmissione a finestra scorrevole

Finestra di trasmissione

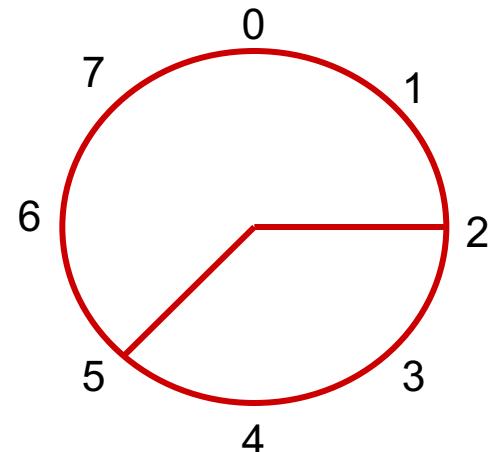
- $W_T = \text{numero massimo di trame}$ che il trasmettitore può inviare **senza ricevere alcuna conferma**
- La numerazione delle trame viene effettuata **modulo M**
 - $M = 2^n$ dove n è il numero di bit utilizzati per la numerazione
- Si può procedere con la trasmissione di nuove trame solo al ricevimento delle conferme
 - La numerazione delle trame trasmesse **scorre nel tempo (sliding window)**



$W_T = 4$
46 Trasmetto 0, 1, 2, 3



Ricevuto ACK 1
Trasmetto 4

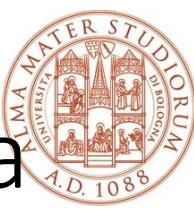


Ricevuto ACK 2
Trasmetto 5



Dimensione della finestra

- Per quale motivo imporre W finito e sospendere la trasmissione delle trame?
 - Garantire unicità di numerazione delle trame
 - Lo spazio di numerazione
 - Dipende dal numero di bit dedicati alla numerazione nell'intestazione
 - Ha necessariamente dimensioni limitate
 - Se si continuasse a trasmettere all'infinito non si avrebbe più una corrispondenza biunivoca trame-numero
 - Le trame con uguale numerazione sono indistinguibili

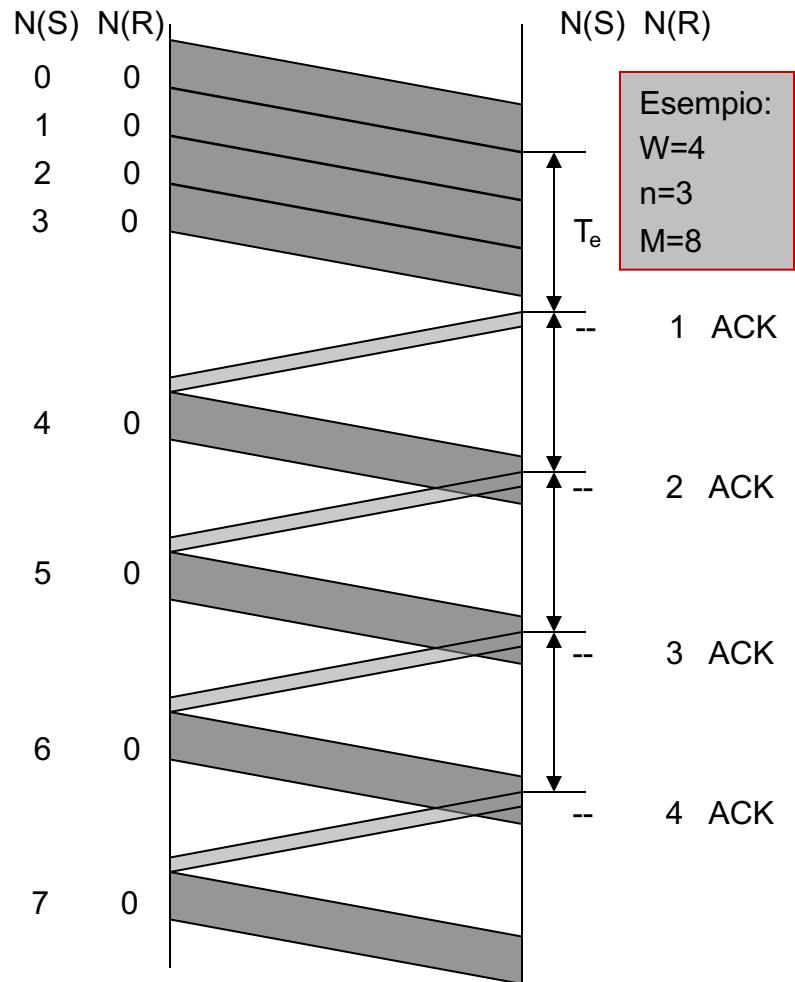


Efficacia della numerazione a finestra

- Permettere la gestione automatica del controllo di flusso
 - Il ricevitore deve poter ricevere un' intera finestra, dopodichè “pilota” il trasmettitore con gli ACK
- Permette di riconoscere l' errata ricezione o la perdita di dati
 - Il ricevitore vede arrivare una trama (segmento) fuori sequenza
- Permette di ricostruire in ricezione la corretta sequenza dei dati

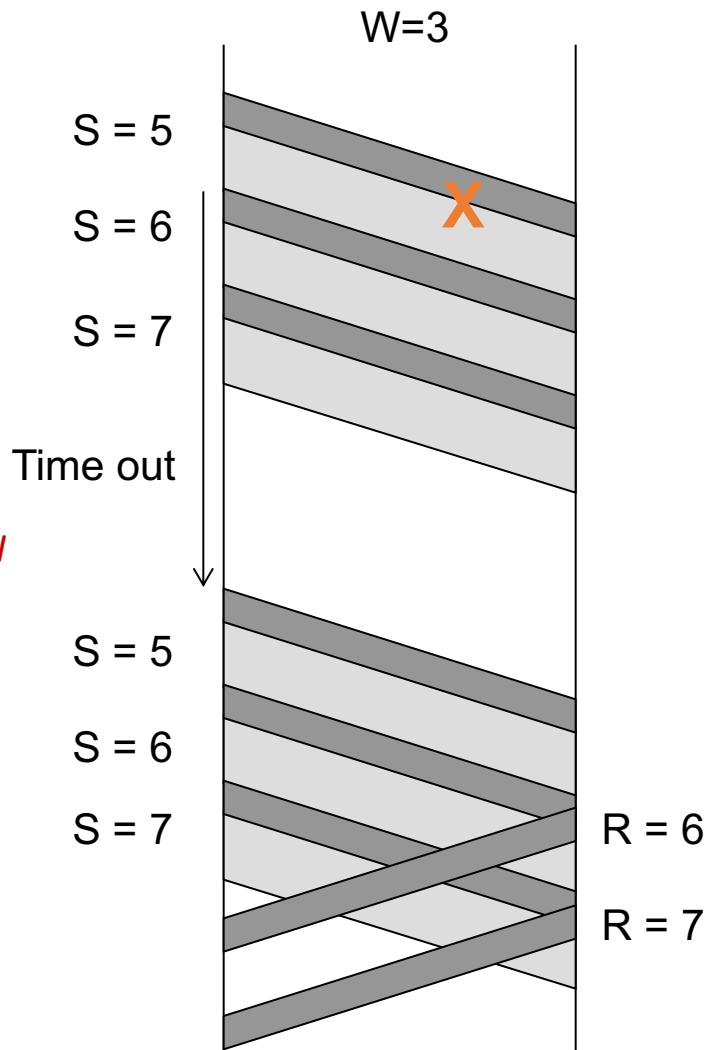
Controllo di flusso

- Accorda la velocità del trasmettitore alla capacità del ricevitore (e della rete)
- Il ricevitore
 - Deve essere in grado di gestire un' intera finestra
 - Memorizzazione ed elaborazione di W trame
 - Accorda il flusso di trame in arrivo tramite le conferme
- A regime un nuova trama ogni T_e
 - T_e = tempo necessario per elaborare una trama



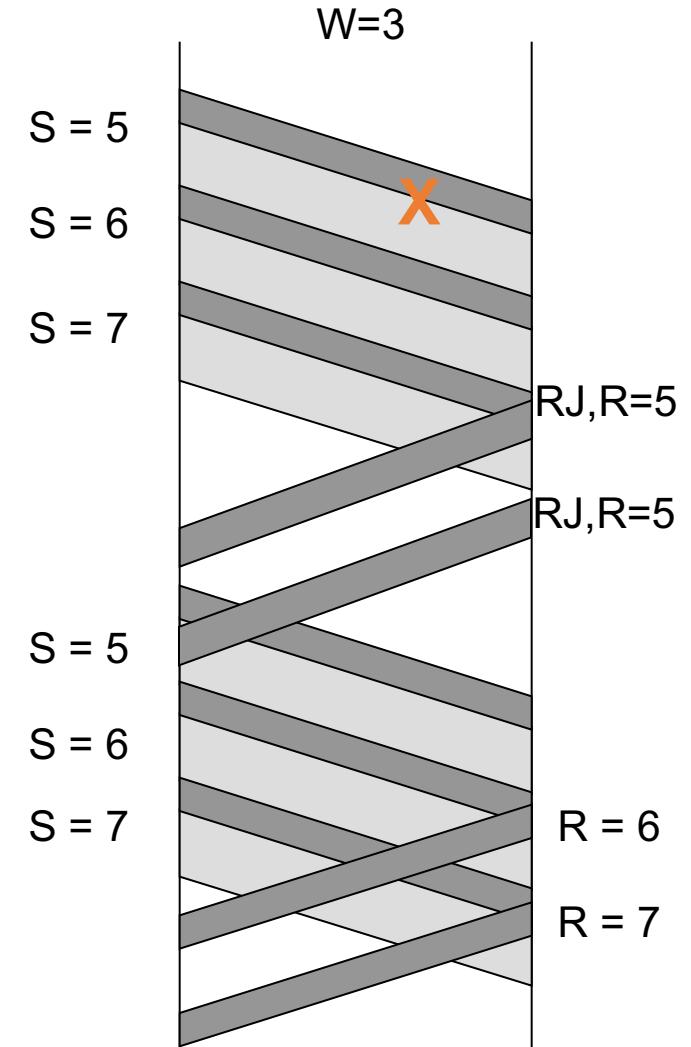
Recupero dell'errore: go-back-n ARQ

- Viene persa la trama N
- Il ricevitore
 - **Scarta** tutte le trame successive a quella errata
 - A seconda dell' implementazione
 - **Segnale** al trasmettitore la mancata ricezione della trama N
 - **Rimane in silenzio** senza inviare alcuna trama di segnalazione
- Il trasmettitore
 - **Ritrasmette** tutte la trame a partire dalla numero N
- Vantaggi
 - Semplicità operativa
 - Ridotta complessità nel ricevitore
- Svantaggi
 - Inefficienza
 - Si ritrasmettono trame senza che questo sia strettamente necessario



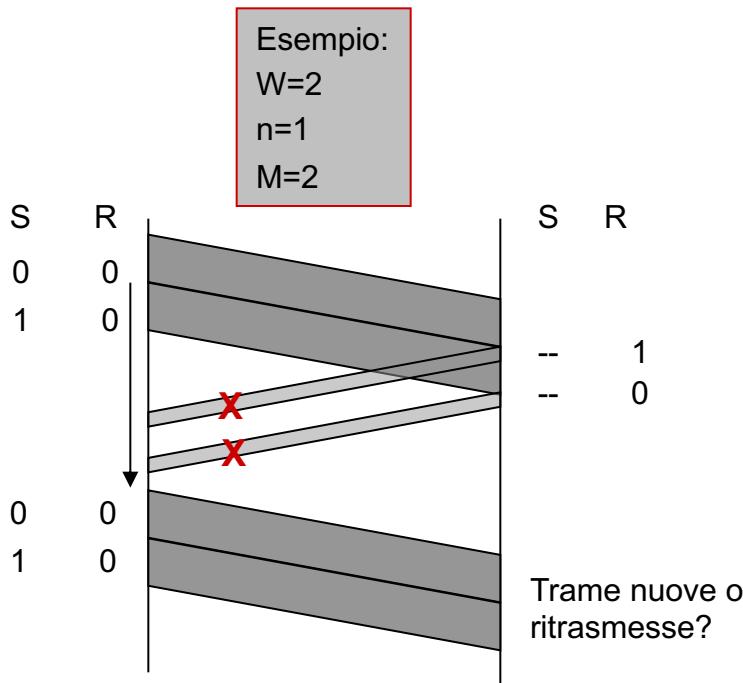
Recupero dell'errore: go-back-n ARQ

- Viene persa la trama N
- Il ricevitore
 - Scarta tutte le trame successive a quella errata
 - A seconda dell' implementazione
 - Segnale al trasmettitore la mancata ricezione della trama N
 - Rimane in silenzio senza inviare alcuna trama di segnalazione
- Il trasmettitore
 - Ritrasmette tutte la trame a partire dalla numero N
- Vantaggi
 - Semplicità operativa
 - Ridotta complessità nel ricevitore
- Svantaggi
 - Inefficienza
 - Si ritrasmettono trame senza che questo sia strettamente necessario

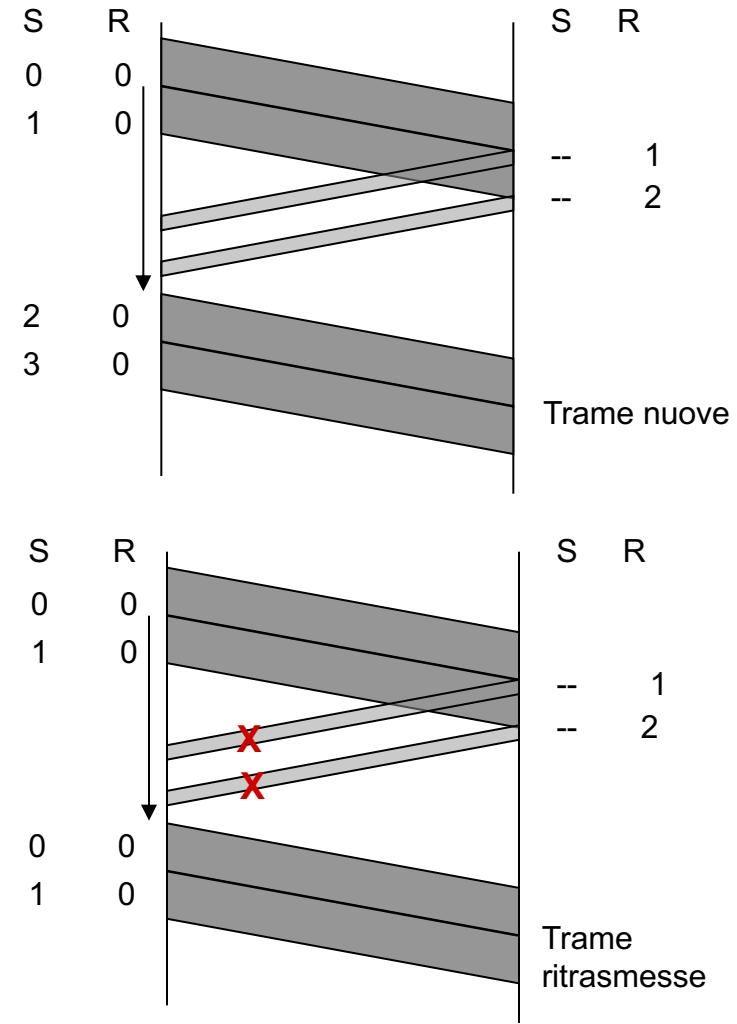


Finestra e numerazione

- Campo di numerazione finito (n bit $\rightarrow M = 2^n$ diversi numeri di sequenza)
 - deve essere $W_T \leq M-1$

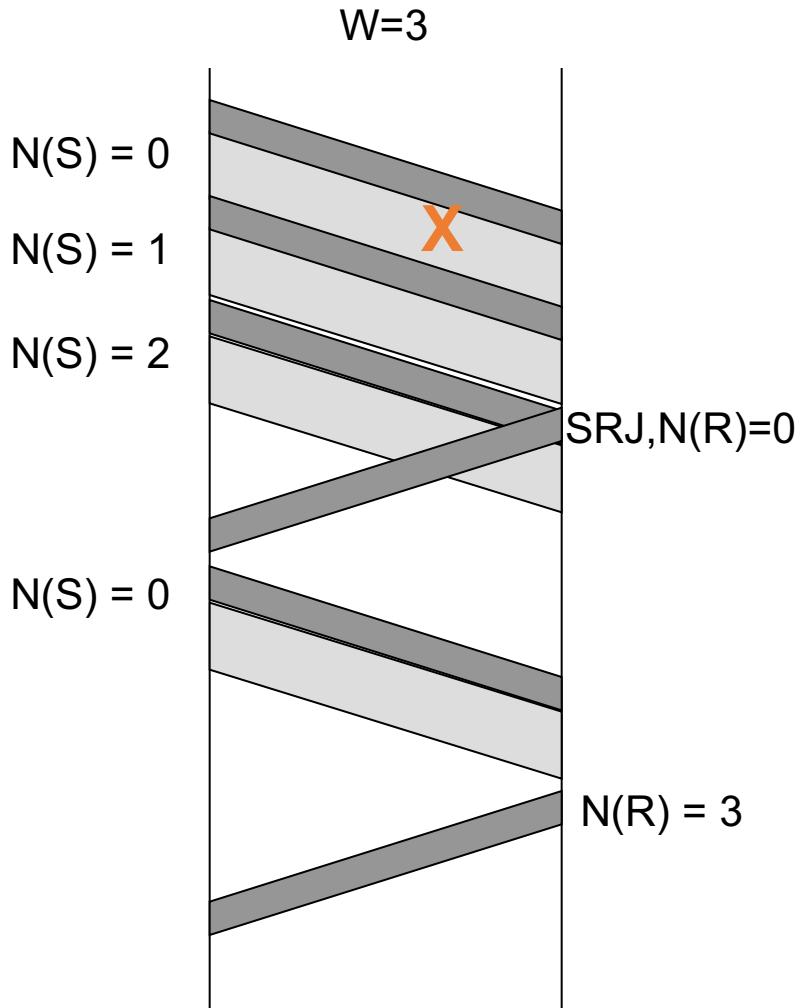


Esempio:
 $W=2$
 $n=2$
 $M=4$



Selective repeat ARQ

- Viene persa la trame N
- Il ricevitore
 - Scarta solamente la trama errata
 - Segnala la mancata ricezione della trama N
- Il trasmettitore
 - Ritrasmette solamente la trama N
- Il ricevitore
 - Riordina le trame nella memoria di ricezione
- Vantaggi
 - Maggiore efficienza
- Svantaggi
 - Complessità del ricevitore
 - Deve tenere in memoria le trame correttamente ricevute fintanto che non può consegnarle allo strato superiore nella giusta sequenza



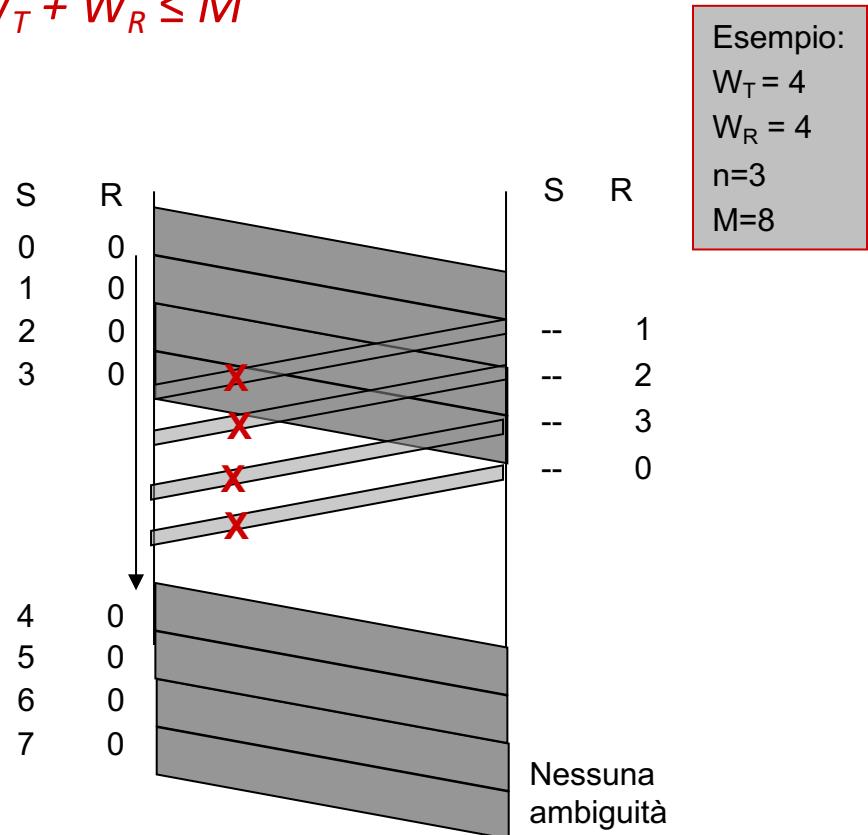
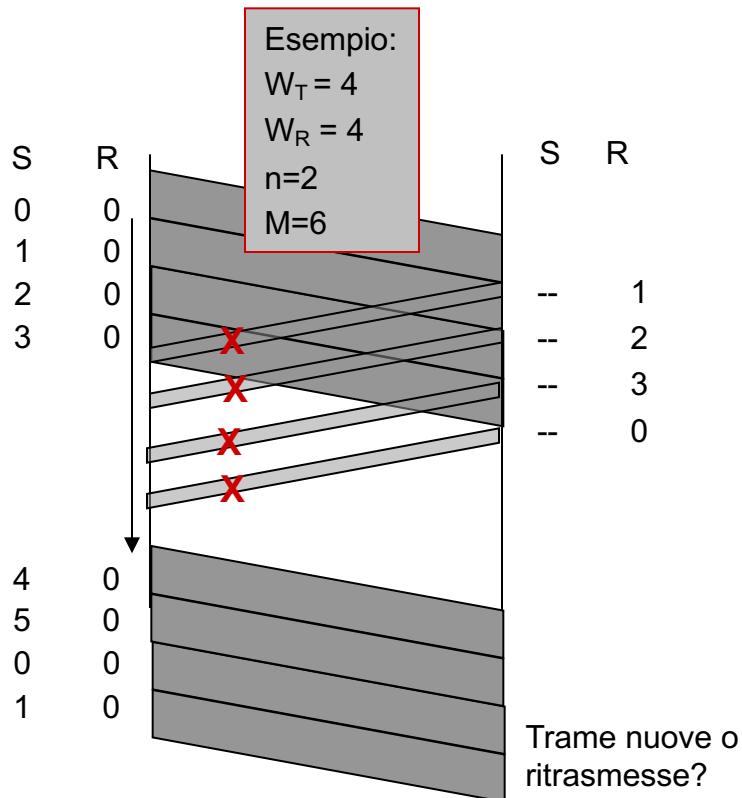


Finestra di ricezione

- Il concetto di finestra si può applica anche in ricezione
- Finestra di ricezione
 - W_R = massimo numero di PDU che Rx può memorizzare prima di consegnare i dati allo strato superiore
 - Nel protocollo go-back-N tipicamente $W_R = 1$
 - In un protocollo Selective Repeat deve essere $W_R > 1$

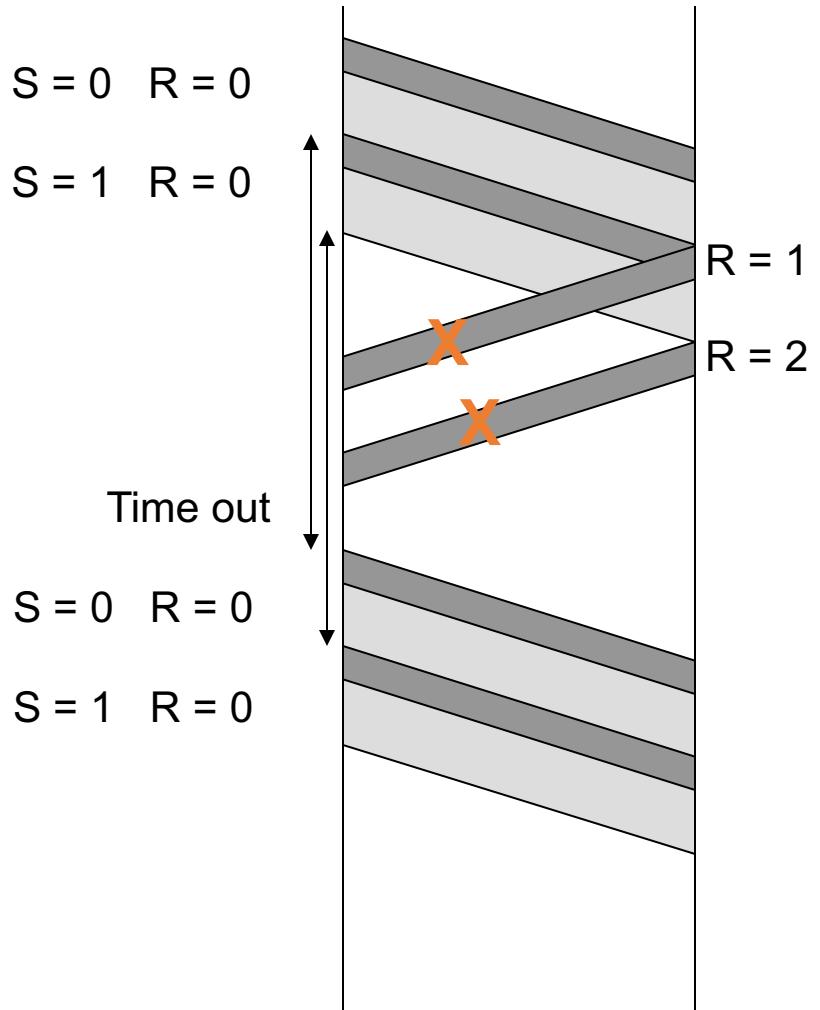
Finestra e numerazione

- Campo di numerazione finito (n bit -> $M = 2^n$ diversi numeri di sequenza)
 - Se $W_T = W_R$ deve essere $W_T + W_R \leq M$



Time out

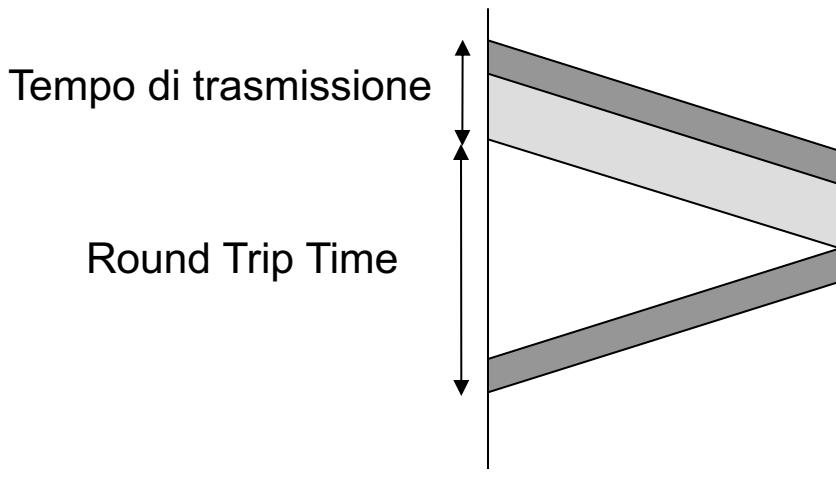
- Il protocollo può entrare in stallo (**deadlock**)
 - Se le trame informative sono perdute
 - Se gli ACK sono perduti
- È necessario un **time out** per riprendere il dialogo
 - Un orologio parte al termine della trasmissione di ciascuna trama
 - Se si raggiunge il time out senza avere conferma si ritrasmette la trama





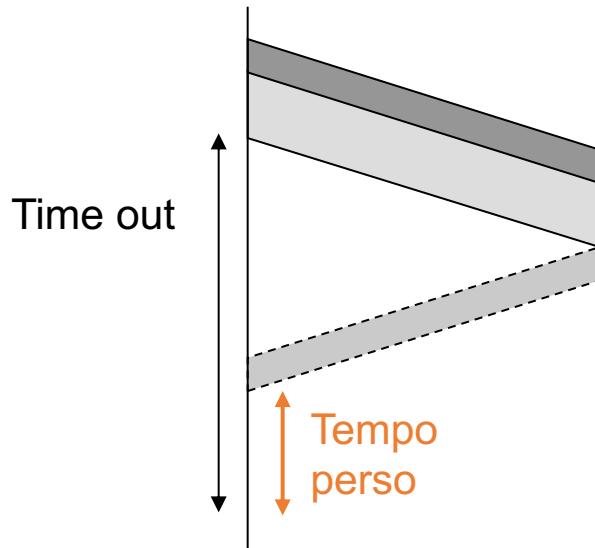
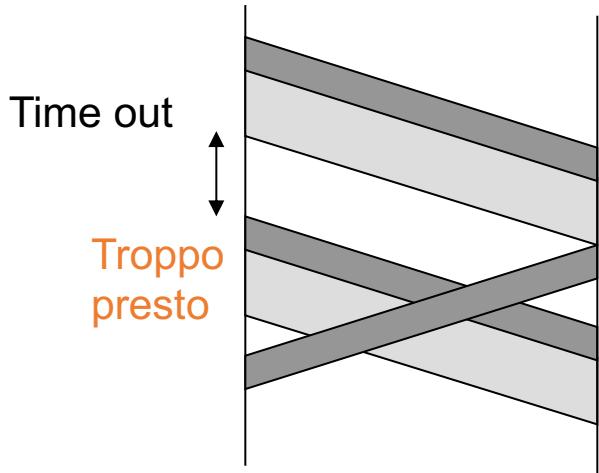
Il round trip time (RTT)

- RTT = tempo necessario per effettuare un' andata e ritorno sul canale
 - Tempo intercorso fra la partenza dell' ultimo bit di una trama e la ricezione del relativo ACK
- Variabilità di RTT
 - RTT è praticamente deterministico per lo strato 2
 - RTT può variare da segmento a segmento per lo strato 4



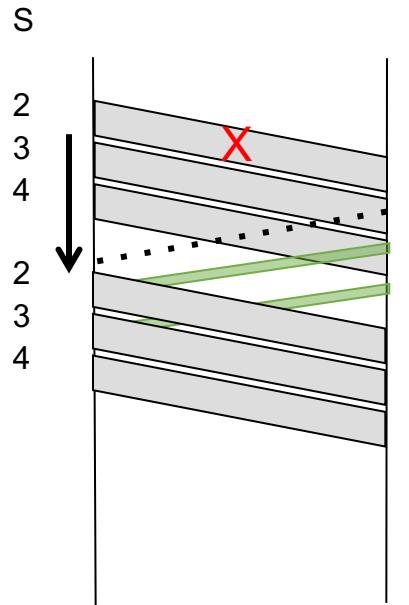
Dimensione del Time out?

- Il time out va relazionato al RTT
- Time out troppo breve
 - Non si attende l'arrivo dell'ACK
 - Invio non necessario di trame duplicate
- Time out troppo lungo
 - Inutile attesa prima di ritrasmettere le trame errate

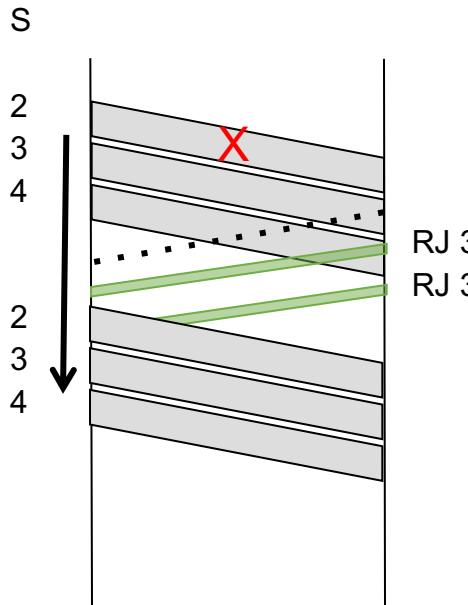


- In entrambi i casi
 - Si spreca capacità di trasmissione (banda)
 - Degradano le prestazioni

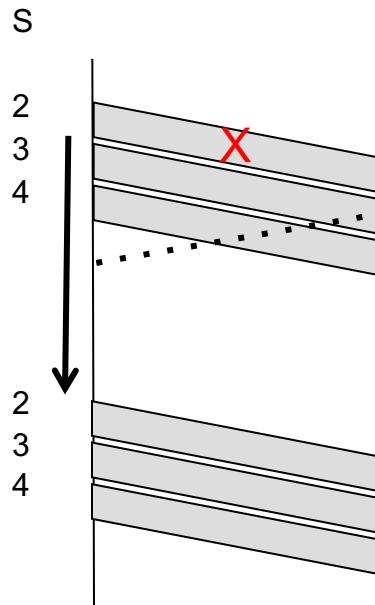
Esempio: $W_T = 3$



Time out correttamente dimensionato: equivalente con o senza Reject



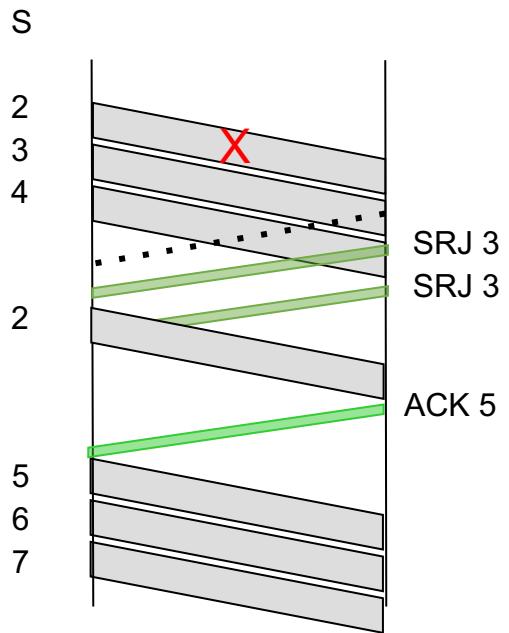
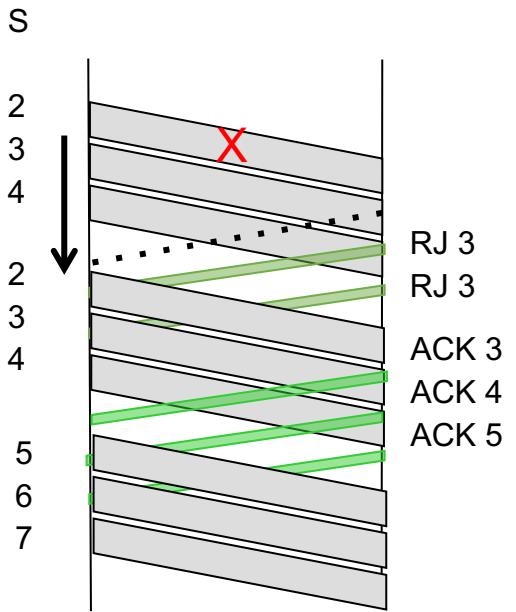
Time out mal dimensionato:
Reject permette di reagire
prima alla perdita



Time out mal dimensionato:
l'assenza di Reject fa
perdere tempo

- Go-Back-N con e senza ACK negativo (Reject)
 - Reject utile in caso di timeout mal dimensionato

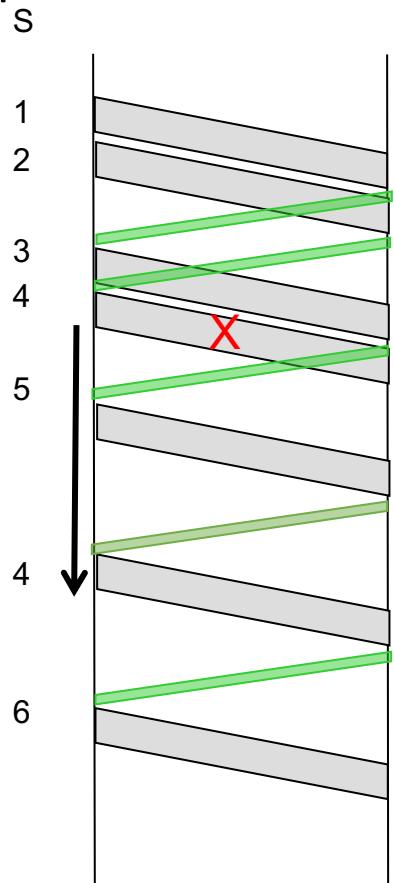
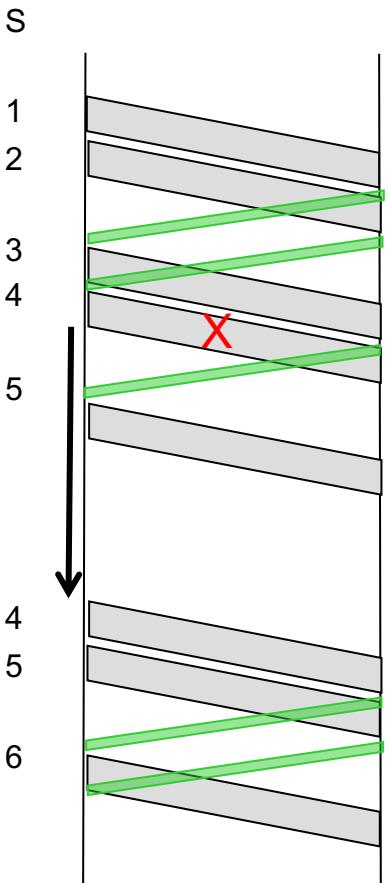
Esempio: $W_T = 3$



La dimensione della finestra è prossima al RTT

- Go-Back-N a confronto con Selective Repeat
 - Una perdita singola non determina particolare differenza

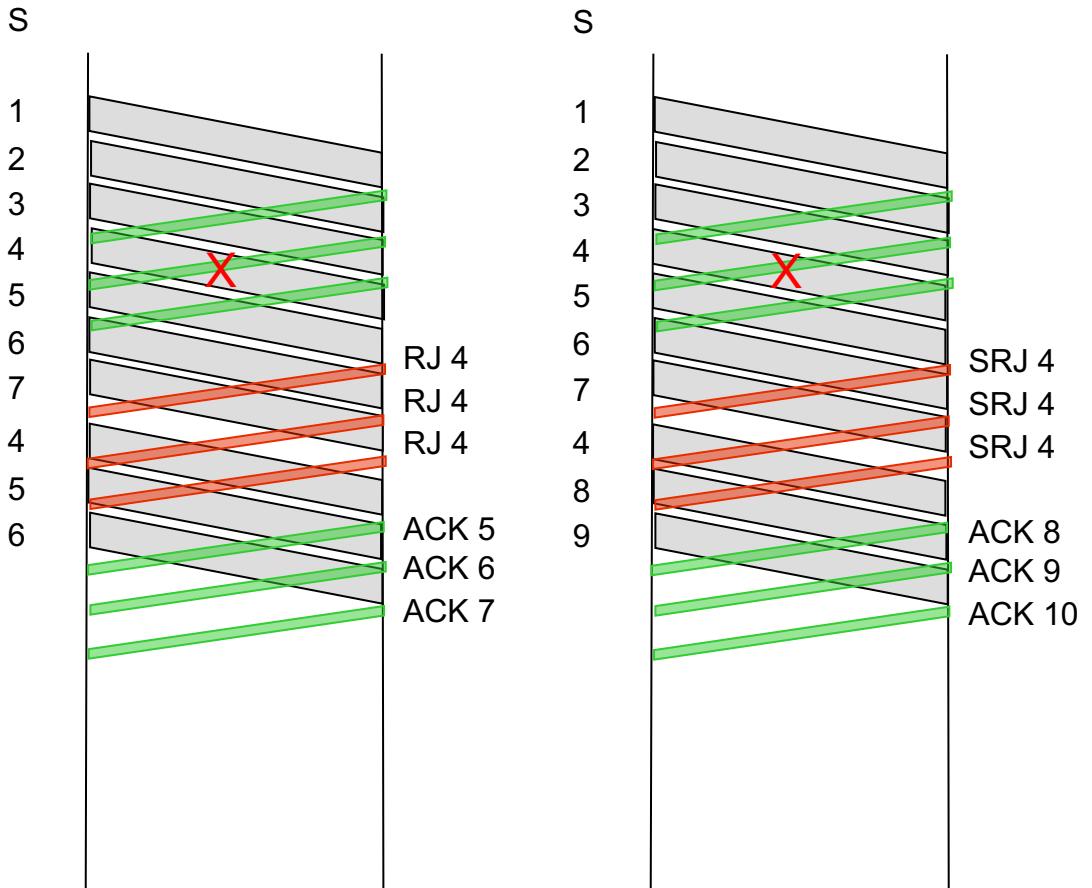
Esempio: $W_T = 2$



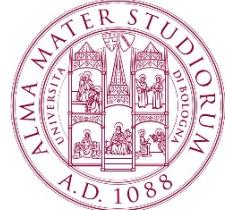
Il time out è sovradimensionato
 $W_T < RTT$

- Selective Repeat ha un vantaggio

Esempio: $W_T = 6$



- Selective Repeat ha un vantaggio
 - Una perdita singola non determina particolare differenza



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Prestazioni ed efficienza dei protocolli di strato 2

Franco CALLEGATI

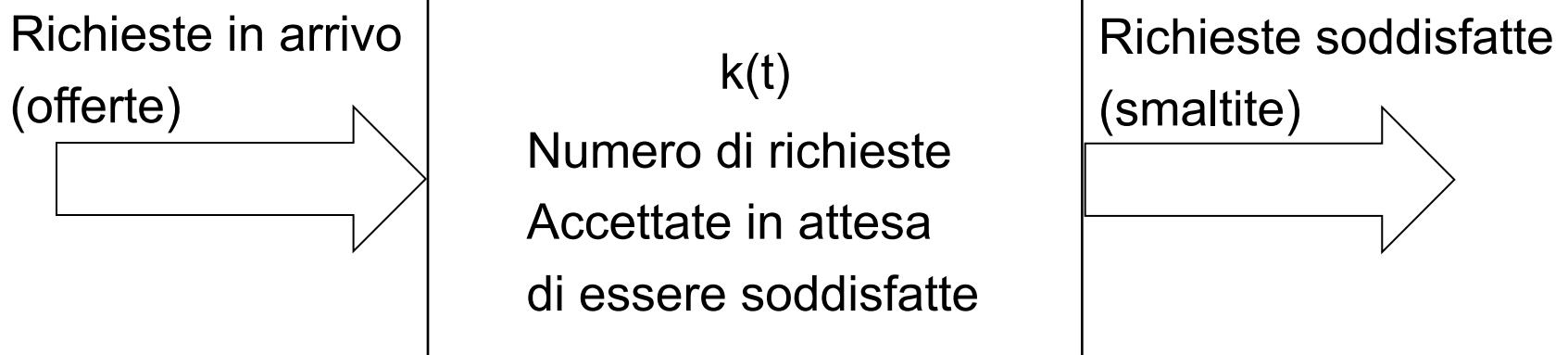
Dipartimento di Informatica: Scienza e Ingegneria

A.A. 2021-2022



Di cosa parliamo?

- Un *sistema* deve *smaltire* del *lavoro* che gli viene *offerto* dall'esterno
- Esempio nel caso specifico delle reti di tlc
 - Livello N+1 invia PDU al livello N tramite la relativa interfaccia (e un opportuno SAP)
 - Livello N impiega un certo tempo per soddisfare la richiesta



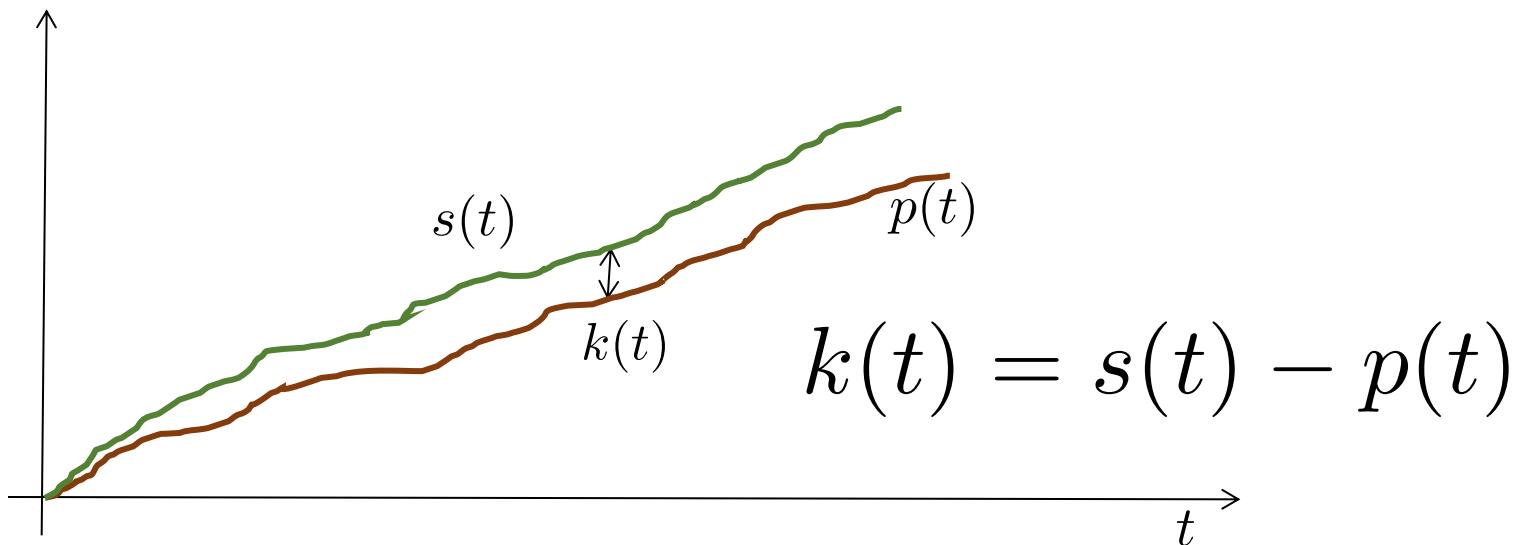


Arrivi e partenze

$a(t)$ • Numero di richieste di servizio giunte al tempo t

$s(t)$ • Numero di richieste accettate al tempo t

$p(t)$ • Numero di partenze dal sistema al tempo t





Richieste offerte e smaltite

- Frequenza media delle richieste offerte

$$\lambda = \lim_{t \rightarrow \infty} \frac{a(t)}{t}$$

- Frequenza media delle richieste smaltite

$$\lambda_s = \lim_{t \rightarrow \infty} \frac{p(t)}{t}$$

- Se il sistema in oggetto non produce lavoro ma lo riceve solamente dall'esterno

$$\lambda_s \leq \lambda$$



Richieste percate

$$\lambda_s = \lambda \quad \text{Implica} \quad s(t) = a(t)$$

- Tutte le richieste vengono accettate dal sistema e prima o poi soddisfatte

$$\lambda_s < \lambda \quad \text{implica} \quad r(t) = a(t) - s(t)$$

- Dove $r(t)$ rappresenta le richieste che non vengono accettate e sono *rifiutate o perdute* dal sistema



Analogamente

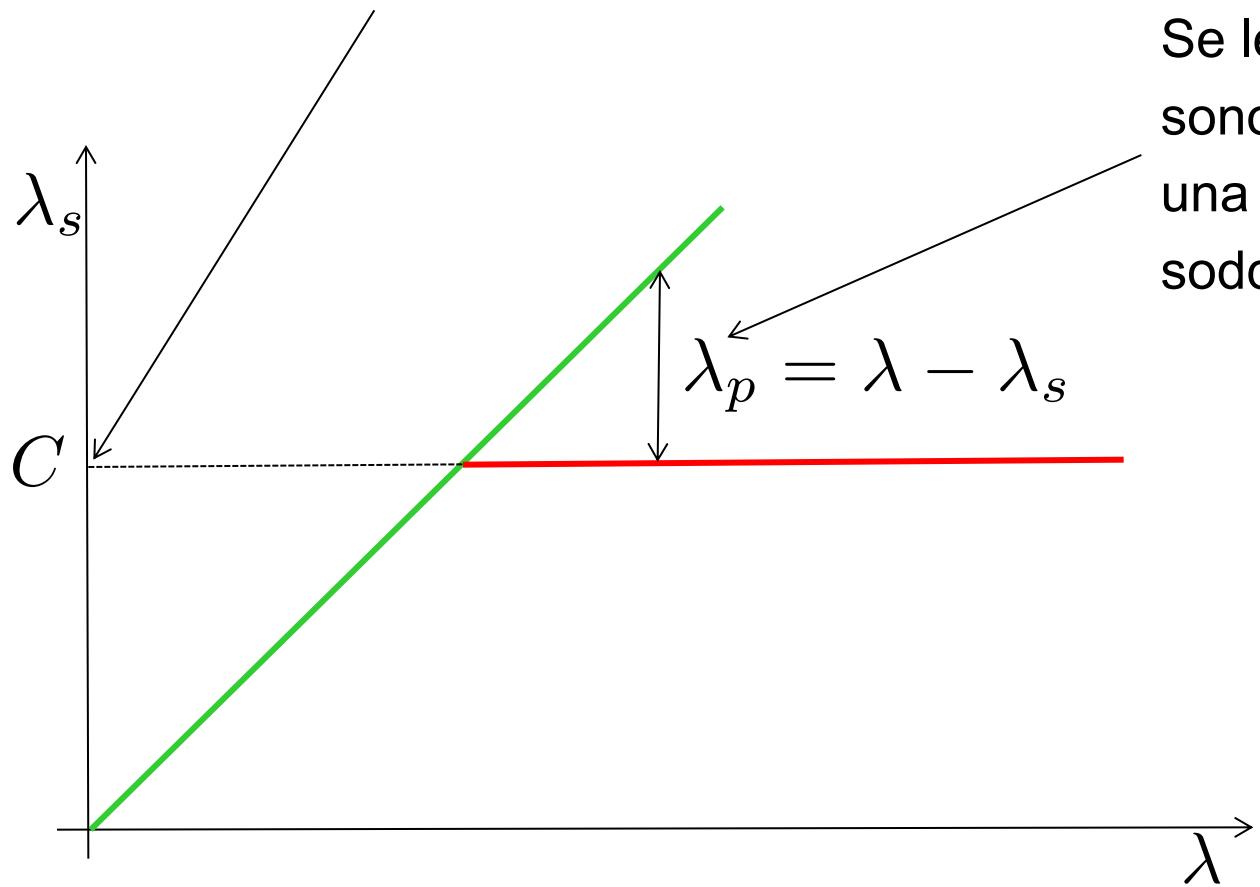
- Posso definire

$$\lambda_p = \lim_{t \rightarrow \infty} \frac{r(t)}{t}$$

$$\lambda = \lambda_s + \lambda_p$$

In un sistema ideale

Il sistema ha una capacità massima finita
(dipende dalle condizioni in cui opera)



Se le richieste offerte
sono eccessive
una parte non può essere
soddisfatta



Capacità massima ed efficienza

- In un protocollo di strato due, qual è la capacità massima?
 - Poiché il protocollo invia i bit sul canale la sua capacità massima teorica è la velocità del canale C
- Ma il protocollo di strato 2 utilizza parte della capacità del canale per suoi scopi
 - PCI necessarie per la segnalazione
 - Tempi morti legati alle sue dinamiche
- La capacità effettivamente disponibile all'utente

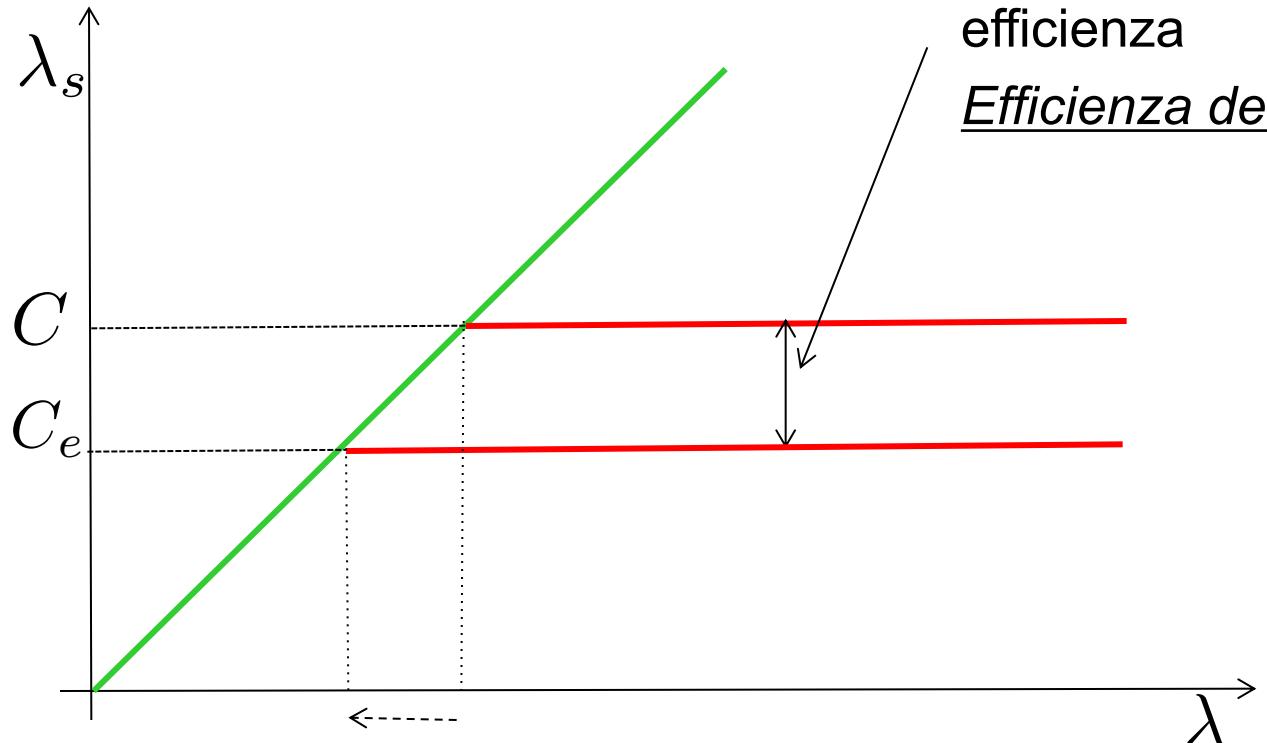
$$C_e \leq C$$

In un sistema ideale

La riduzione di capacità si interpreta come perdita di efficienza

Efficienza del protocollo

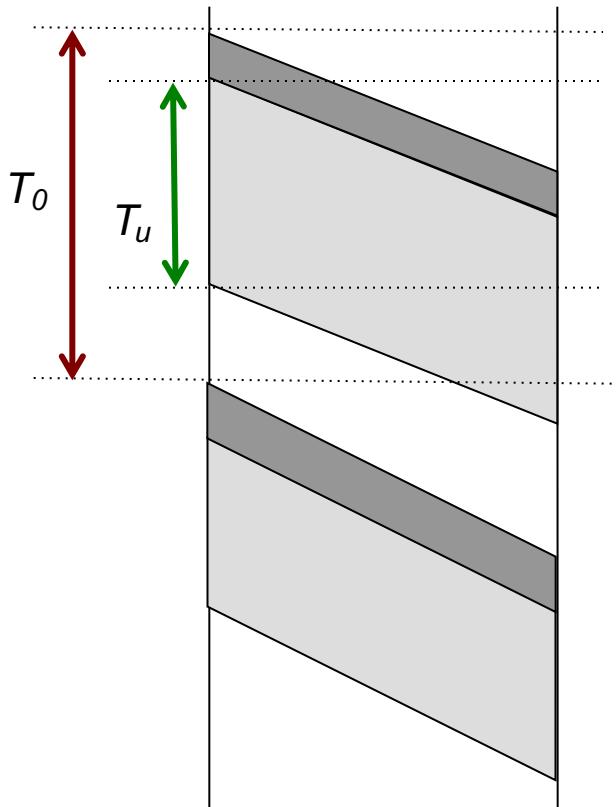
$$\eta = \frac{C_e}{C} \leq 1$$



La riduzione di capacità obbliga il livello superiore a limitare la proprie richieste pena la perdita dei dati

Valutazione efficienza

- Per valutare l'efficienza di solito si fa riferimento alla PDU
- Si confronta:
 - La quantità di tempo strettamente utilizzato per inviare i soli dati d'utente (SDU)
 - La quantità di tempo utilizzato complessivamente per completare correttamente l'invio della PDU
 - In funzione delle regole del protocollo
- L'efficienza è data dal rapporto fra queste due quantità



$$\eta = \frac{T_u}{T_0}$$



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

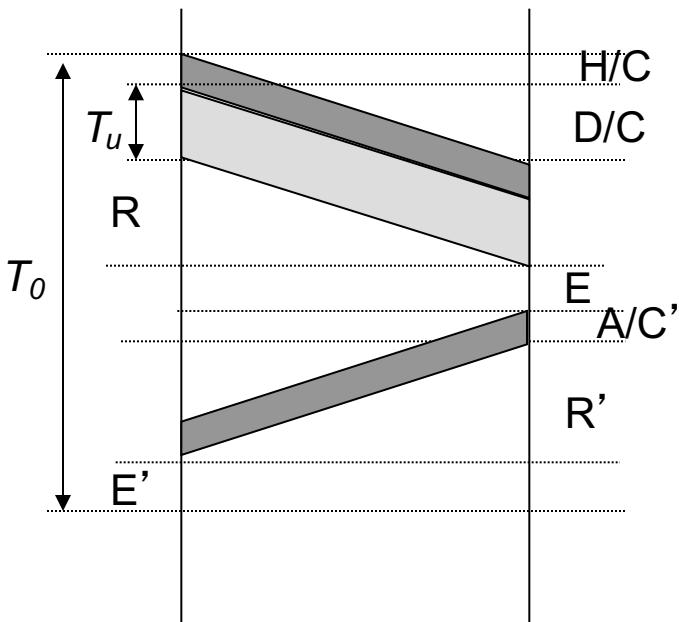
Prestazioni dei protocolli ARQ

Prof. Franco Callegati

<http://deisnet.deis.unibo.it>

Prestazioni Stop-and-Wait

- Qual è l'efficienza del protocollo ARQ stop and wait
 - Stop and wait equivale ad un protocollo a finestra scorrevole con finestra unitaria
- D : dimensione campo dati in bit
- H : dimensione dell' header (PCI) in bit,
- F=D+H : lunghezza totale del frame,
- A : lunghezza dell'ACK,
- E, E' : tempi di elaborazione per il controllo del frame in arrivo e per la preparazione del frame in partenza
- R : tempo di propagazione del segnale da un capo all'altro del collegamento,
- I = E+R ; I' = E' +R'
- C, C' : velocità dei canali di trasmissione
- In generale il canale di andata e di ritorno possono essere diversi





Efficienza

- Tempo intercorso fra l' invio di due frame successivi:

$$T_0 = F/C + I + A/C' + I'$$

- Il tempo strettamente necessario per la trasmissione dei dati di utente è

$$T_d = D/C$$

- Efficienza:

$$\eta = T_d / T_0 = D / (C T_0) = D / (D + H + IC + I'C + AC/C')$$

- Per semplicità poniamo $I = I'$, $C = C'$, inoltre l'ACK è praticamente composto dalla sola PCI, e quindi $A \approx H$

$$\eta = D / (D + H + A + 2IC) = D / (D + 2H + 2IC) = D / (D + O)$$

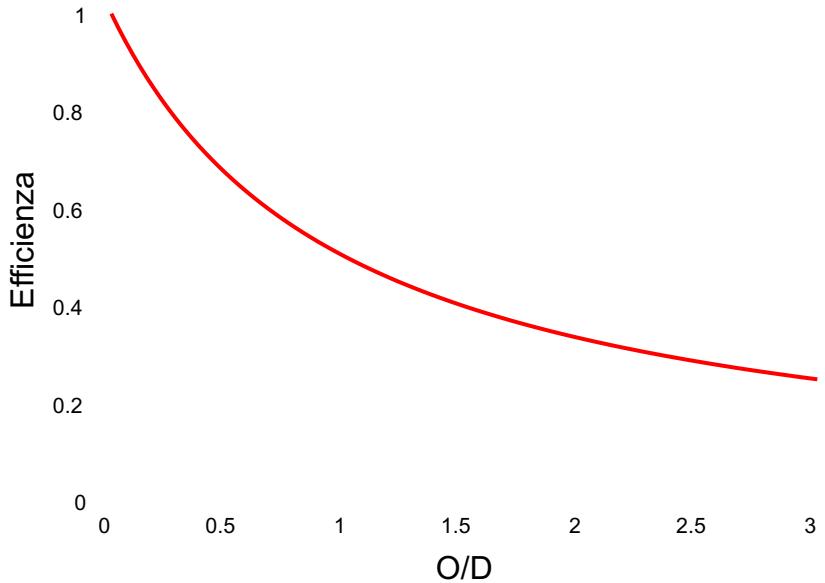
- Overhead

$$O = 2H + 2IC$$



Overhead

- Rappresenta la quantità di dati aggiuntivi introdotti dal protocollo
 - O è una grandezza in bit
 - L' efficienza diminuisce al crescere di O
- O tiene conto di
 - bit aggiuntivi di controllo (PCI)
 - tempo non utilizzato dalla trasmissione a causa del protocollo ARQ

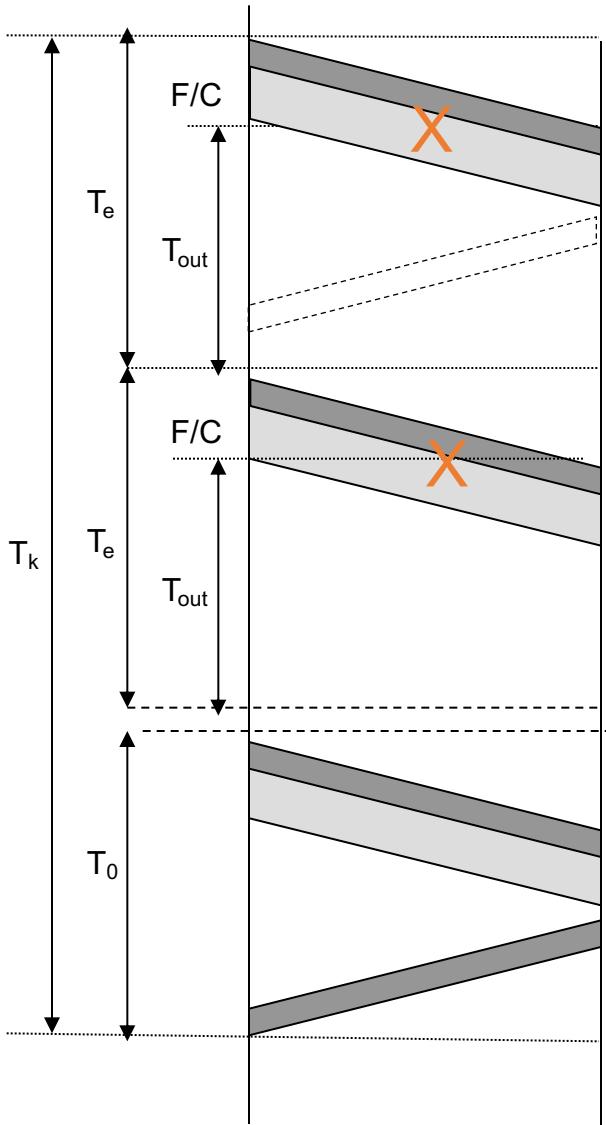


- L' efficienza **diminuisce**
 - Al **crescere di H**
 - Molti bit per PCI
 - All' **aumentare di C**
 - Linea molto veloce
 - All' **aumentare di I**
 - Grandi distanze

Caso con errore

- Prima di trasmettere correttamente una trama possono avvenire k errori
 $k \geq 0$
 - Il tempo necessario a trasmettere la trama, dati k errori, vale
- $$T_k = k T_e + T_o$$
- Dove
- $$T_o = (D + O)/C$$
- $$T_e = D/C + H/C + T_{out}$$
- Se P_k è la probabilità di avere k errori, il tempo medio per trasmettere una trama vale

$$E[T_k] = \sum_{k=0...∞} T_k P_k = \sum_{k=0...∞} (k T_e + T_o) P_k$$





Determinare P_k

- Ipotesi
 - Errore sulle trame i.i.d. (indipendenti ed identicamente distribuite, ossia di uguale probabilità) P_F : *prob. di ricevere una trama errata*

$$P_k = \text{prob}\{k \text{ trame errate seguita da 1 corretta}\} = P_F^k (1 - P_F)$$

- Il numero medio di errori consecutivi risulta

$$E[k] = P_F / (1 - P_F)$$



P_k

- Ipotesi
 - Trame errate i.i.d. (indipendenti e di uguale probabilità)
 - P_F = probabilità di errore per trama
- Una trama per essere ricevuta deve essere trasmessa senza errori
 - K transmissioni errate consecutive con probabilità P_F
 - seguite da
 - 1 trasmissione corretta con probabilità $(1 - P_F)$

$$P_k = \text{prob}\{k \text{ trame errate seguita da 1 corretta}\} = P_F^k (1 - P_F)$$

- Il numero medio di errori consecutivi risulta

$$E[k] = P_F / (1 - P_F)$$



Determinare P_F

- Ipotesi
 - Errore sui bit i.i.d.
 - P_e : probabilità di errore per bit
- È possibile calcolare P_k
 - Ipotesi: errori sui bit *indipendenti*
 - $P_F = 1 - \text{prob}\{\text{trama corretta}\} = 1 - (1 - P_e)^F \simeq F P_e \simeq D P_e$
 - Nelle reti telefoniche gli errori sono a **burst** e vale la formula approssimata
 - $P_F = \alpha F^\beta$, con $\beta > 1$
 - Se però $\beta \simeq 1$ si può usare la formula dei bit indipendenti



Il valor medio di T_k

- In teoria possono essere errate le trame ma anche le conferme
- Se vale la formula $P_F = F P_e$
 - Se P_e è circa costante
 - P_F dipende solamente da F
 - La probabilità di sbagliare trame corte è trascurabile rispetto a quella di sbagliare trame lunghe

$$\begin{aligned} E[T_k] &= \sum_{k=0}^{\infty} (kT_e + T_0)P_F^k(1 - P_F) = (1 - P_F) \sum_{k=0}^{\infty} (kT_e P_F^k + T_0 P_F^k) \\ &= (1 - P_F) \left(T_e \sum_{k=0}^{\infty} kP_F^k + T_0 \sum_{k=0}^{\infty} P_F^k \right) = (1 - P_F) \left(T_e \frac{P_F}{(1 - P_F)^2} + T_0 \frac{1}{1 - P_F} \right) \end{aligned}$$

$$E[T_k] = T_0 + T_e \frac{P_F}{1 - P_F}$$



In conclusione

- Efficienza: $\eta = (D/C) / E[T_k]$
- $T_o = (D + O)/C$
- $T_{out} = I + H/C + I = 2I + H/C$
- $T_e = D/C + T_{out} + H/C = D/C + 2I + 2H/C = D/C + O/C$
- L' efficienza massima

si ha con T_{out} minimo
e T_e minimo :

$$\eta_{Max} = \frac{D}{\left((D+O) + (D+O) \frac{FP_e}{(1-FP_e)} \right)}$$

- Assumendo che:

- $FP_e \ll 1$
- $O \ll D$

Risulta: $\eta_{Max} \cong \frac{D}{D + O + D^2 P_e}$



Efficienza ottima

- Derivando ed uguagliando a zero l'espressione dell'efficienza *massima*, si ottiene il valore ottimo per D :

$$D_{ott} = \sqrt{\frac{O}{P_e}}$$

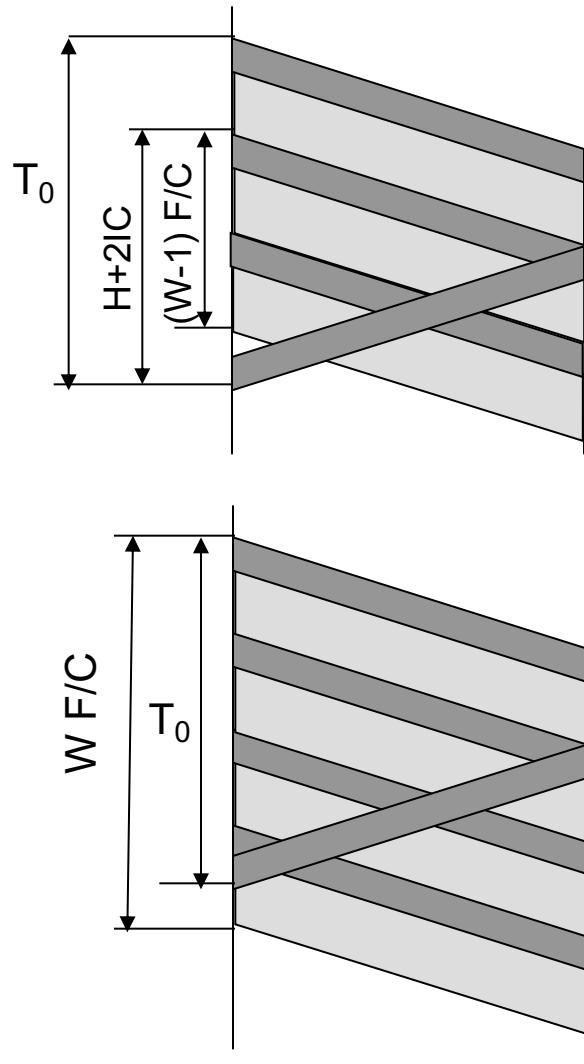
- Sostituendo il valore D_{ott} nell'espressione di η_{max} si ottiene:

$$\eta_{ott} = \frac{D_{ott}}{D_{ott} + 2O}$$

Finestra $W > 1$: in assenza di errori

- È necessario distinguere due casi
 - $(W-1) F/C < H + 2IC$, cioè $WF < CT_0$
 - Si trasmettono W trame in un tempo pari a T_0 per cui
$$\eta = (WD)/(D+2H+2IC)$$
 - $WF \geq CT_0$
 - il trasmettitore non interrompe mai la trasmissione delle trame per cui risulta:
$$\eta = D/(D+H) \text{ oppure}$$

$$\eta = D/(D+O) \text{ , con } O = H$$
- Il secondo caso è il migliore per l'efficienza : in trasmissione non ci sono tempi morti



Finestra $W > 1$: caso con errore

- Supponiamo di metterci nel caso $W F > C T_0$
 - Assenza di tempi morti
- Distinguiamo due casi:
 - Caso Selective Repeat ARQ
 - L' errore sulla trama comporta la sola ritrasmissione della trama stessa
 - $C T_{medio} = F + F [P_F / (1-P_F)] \simeq F + F P_F \simeq D + H + D^2 P_e$

$$\eta = \frac{D}{D + H + D^2 P_e}$$

$$D_{ott} = \sqrt{\frac{H}{P_e}}$$

$$\eta_{ott} = \frac{D_{ott}}{D_{ott} + 2H}$$

- Caso Go-back-n
 - L' errore comporta in media la ritrasmissione di $w = W/2$ trame (l' errore si colloca casualmente all' interno della finestra)
 - $C T_{medio} = F + W/2 F P_F \simeq D + H + W/2 D^2 P_e$

$$\eta = \frac{D}{D + H + w D^2 P_e}$$

$$D_{ott} = \sqrt{\frac{H}{w P_e}}$$

$$\eta_{ott} = \frac{D_{ott}}{D_{ott} + 2H}$$



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Alcuni Esercizi



Esercizio – 1 (a)

- Protocollo Stop and Wait
 - Velocità della linea: $C = 4 \text{ Kbit/s}$
 - Ritardo di elaborazione e propagazione: $I = 20 \text{ ms}$
 - $H \approx A \approx 0$
- Determinare la dimensione della trama tale che l'efficienza $\eta > 50\%$
- Formula dell'efficienza:

$$\eta = \frac{D}{T_0 C};$$

$$T_0 = \frac{F}{C} + I + \frac{A}{C} + I;$$



Esercizio – 1 (b)

- Sviluppando i termini:

$$\eta = \frac{D}{F + 2IC + A} = \frac{D}{D + H + 2IC + A};$$

- Nell'ipotesi che $A \simeq H$:

$$\eta = \frac{D}{D + \underbrace{2H + 2IC}_O};$$

- Con O si indica l'overhead, cioè la quantità di dati aggiuntivi introdotti dal protocollo



Esercizio – 1 (c)

- Sostituendo i dati di progetto forniti dal testo:

$$\eta = \frac{D}{D + 2H + 2IC} \cong \frac{D}{D + 2IC} \geq 0.5$$

$$\Rightarrow D \geq 0.5D + IC \Rightarrow 0.5D \geq IC$$

$$\Rightarrow D \geq 2IC = 2 \cdot 4 \cdot 10^3 \cdot 20 \cdot 10^{-3} = 8 \cdot 20 = 160$$

- Affinché sia soddisfatto il vincolo richiesto sull'efficienza ($\eta > 50\%$), la trama deve essere lunga almeno 160 Bit



Con errori di trasmissione

- $2IC = 160$ bit
- $P_e = 10^{-3}$
- $O/Pe = 160000$
- $D_{ott} = \text{Sqrt}(O/Pe) = 400$
- $\eta = 400/(400+320) = 0,56$
- $P_e = 10^{-2}$
- $O/Pe = 16000$
- $D_{ott} = \text{Sqrt}(O/Pe) = 126$
- $\eta = 126/(126+320) = 0,28$



Esercizio – 2 (a)

- Protocollo a Finestra
- Si supponga di dover trasferire dei dati fra 2 siti con una linea con tali requisiti:
 - $d = 100 \text{ Km}$ (distanza tra i 2 siti)
 - $C = 64 \text{ Kbit/s}$ (velocità della linea)
 - ritardo di propagazione $t = 6 \text{ usec/km}$
 - Ritardo di Elaborazione pressoché;
 - $D = 200 \text{ bit}$; $H \approx A = 20 \text{ bit}$
 - $I = 6 \cdot 10^{-6} * 100 = 600 \text{ usec.}$
- Determinare la dimensione della finestra W in grado di garantire efficienza massima



Esercizio – 2 (b)

- Calcolo del tempo totale di trasmissione:

$$T_0 = \frac{F}{C} + I + \frac{A}{C} + I;$$

$$\begin{aligned} T_0 &= \frac{D+H}{C} + \frac{A}{C} + 2I = \frac{220+20}{C} + 2I = \frac{240}{64 \cdot 10^3} + 1200 \cdot 10^{-6} = \\ &= 3.75 \cdot 10^{-3} + 1.2 \cdot 10^{-3} = 4.95 \cdot 10^{-3} \text{ sec} \cong 5 \cdot 10^{-3} \text{ sec} \end{aligned}$$

- Ricordando la condizione sulla dimensione della finestra:

$$\frac{WF}{C} > T_0$$

$$W \geq \frac{C \cdot T_0}{F} \geq \frac{64 \cdot 10^3 \cdot 5 \cdot 10^{-3}}{220} \geq \frac{320}{220} \geq 1.45 \Rightarrow W = 2;$$



Esercizio – 2 (c)

- L' efficienza che si ottiene con una finestra di dimensione W=2 è dunque pari a:

$$\eta = \frac{D}{D + H} = \frac{200}{220} = 0.91 = 91\%;$$

- Si ricorda che deve essere:

$$\frac{WF}{C} > T_0$$



Esercizio – 3 (a)

- Protocollo a Finestra Scorrevole
- Caratteristiche:
 - D= 1000 bit
 - H= 100 bit
 - C= 128 Kbit/s
 - I= 10 ms
- Calcolo del tempo totale di trasmissione:

$$T_0 = \frac{F}{C} + I + \frac{A}{C} + I;$$

$$T_0 = \frac{D + H + A}{C} + 2I = \frac{1200}{128 \cdot 10^3} + 20 \cdot 10^{-3} = 29.375 \cdot 10^{-3} \text{ sec}$$



Esercizio – 3 (b)

- Confrontare l'efficienza massima del protocollo con quella del protocollo Stop and Wait.

$$\eta_{sw} = \frac{D}{T_0 C} = \frac{1000}{29.4 \cdot 10^{-3} \cdot 128 \cdot 10^3} = \frac{1000}{3763} = 0.26 = 26\%;$$

- Calcoliamo ora l'efficienza massima per il protocollo a finestre scorrevoli:
 - Se $W=2$ $\eta = 2 * 0.26 = 0.52$
 - Se $W=3$ $\eta = 3 * 0.26 = 0.78$
 - Se $W=4$ $\eta = 4 * 0.26 = 1.04$
- $\eta = 1.04$ è un risultato impossibile indica solo che 4 è la dimensione di W che produce maggiore efficienza



Esercizio – 3 (c)

- Calcoliamo ora l'efficienza per W=4:

$$\eta = \frac{D}{D + H} = \frac{1000}{1100} = 0.91 = 91\%;$$

- Si poteva calcolare W anche utilizzando la formula nota:

$$\frac{WF}{C} > T_0$$

$$W \geq \frac{C \cdot T_0}{F} \geq \frac{3763}{1100} \geq 3.42 \Rightarrow W = 4$$



Esercizio 4 - (a)

- Un protocollo ARQ a finestra scorrevole utilizzato per il controllo di un collegamento satellitare opera nelle seguenti condizioni:
 - $F \approx D = 1 \text{ Kbyte} = 1024 * 8 \text{ bit} = 8192 \text{ bit}$
 - $I = 125 \text{ ms}$
 - $C = 2,048 \text{ Mbit/s}$
 - $H = A$ trascurabili
- Valutare l'efficienza nell'ipotesi di una numerazione della finestra a 3 e a 7 bit



Esercizio 4 - (b)

- $F/C = 4 \text{ ms}$
- $T_0 = D/C + 2H/C + 2I \approx D/C + 2I = 254 \text{ ms}$
- Caso finestra 7
 - $W F/C = 28 \text{ ms}$
 - Efficienza = $28/254 = 0,11$
- Caso finestra 127
 - $W F/C = 0,51 \text{ s}$
 - Efficienza ≈ 1



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Protocolli a contesa: prestazioni e funzionalità



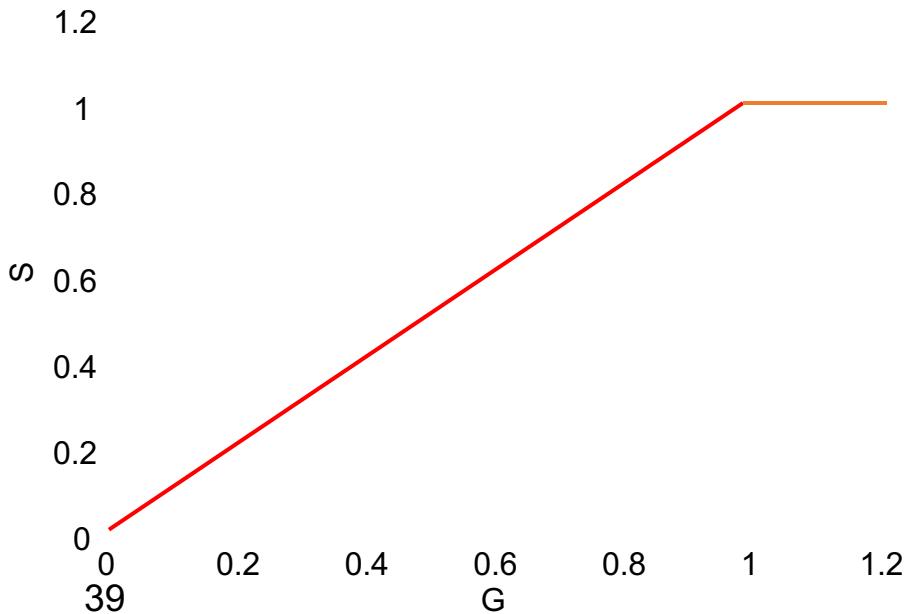
Parametri caratterizzanti la LAN

- F : lunghezza massima della trama
 - Tutte le trame sono della dimensione massima
 - C : velocità di trasmissione sul mezzo
 - d : massima distanza fra due stazioni della LAN
 - v : velocità di propagazione del segnale
-
- $T = F/C$: tempo di trasmissione di una trama
 - d/v : tempo di propagazione di un singolo bit sulla LAN
 - Cd/v : massimo numero di bit che possono essere presenti contemporaneamente sulla LAN



LAN ideale

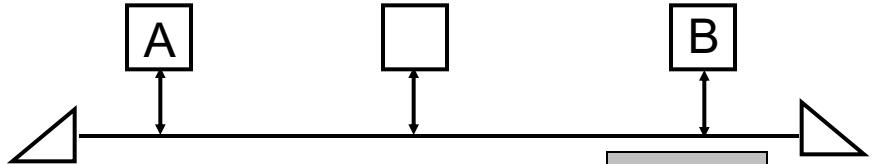
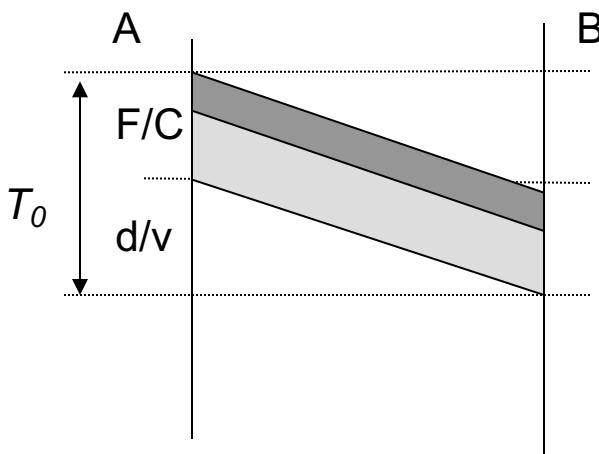
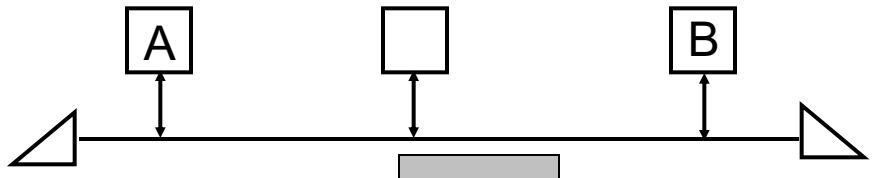
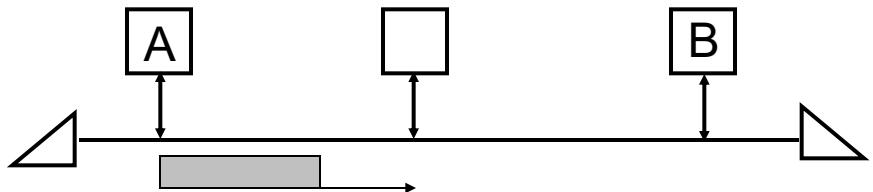
- Utilizza una CAP ideale
 - Coordina le stazioni per evitare accessi contemporanei al canale di trasmissione
 - Tutte le trame in arrivo vengono trasmesse con successo, quindi $G = A_0$
- Il tempo di propagazione della trama è nullo
- È possibile trasmettere le trame una di seguito all' altra
 - Il canale di trasmissione della LAN può essere utilizzato al 100%



Se $A_0 < 1$ allora $S = G = A_0$
Se $A_0 \geq 1$ allora $S = 1$
La LAN ideale permette di smaltire tutto il traffico offerto, fino alla saturazione del canale

Propagazione reale (topologia bus)

- La trama impiega un tempo non nullo per attraversare la LAN
 - t : A inizia la trasmissione
 - $t + F/C$: A termina la trasmissione
 - $t + d/v$: B riceve il primo bit
 - $t + F/C + d/v$: B riceve l'ultimo bit



Efficienza con MAC ideale

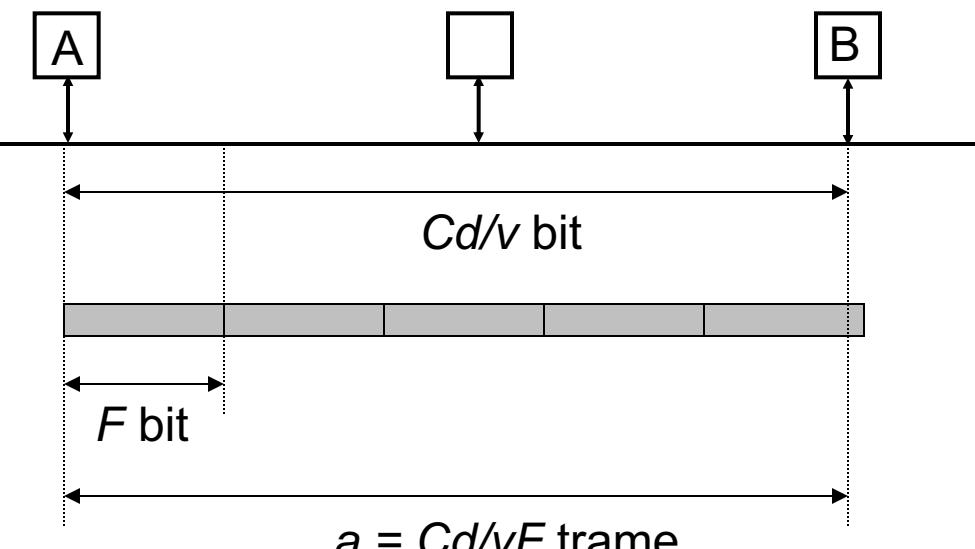
- Una trama tiene impegnata la LAN per T_0
- Il canale di trasmissione non può più essere usato al 100%
- Al massimo viene utilizzato per T secondi ogni T_0
- **Efficienza** del MAC

$$\eta = T/T_0 = (F/C)/(F/C + d/v) = 1/(1+a)$$

- L' efficienza pone un limite superiore al massimo traffico smaltito S

$a = Cd / vF$

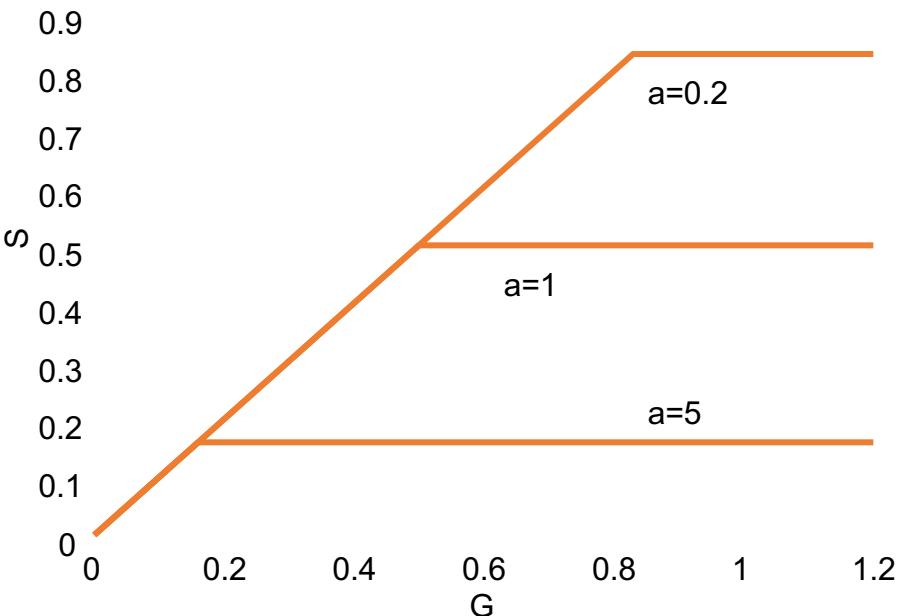
è interpretabile come la lunghezza della LAN misurata in trame MAC



Traffico smaltito dalla LAN

- $G < 1/(1+a)$
 - Tutte le trame in arrivo vengono trasmesse
 - $S = G = A_0$

- $G \geq 1/(1+a)$
 - Il MAC non permette la trasmissione di tutte le trame
 - Parte delle trame viene accodata all' infinito
 - $S = \eta = 1/(1+a)$





Esempio numerico

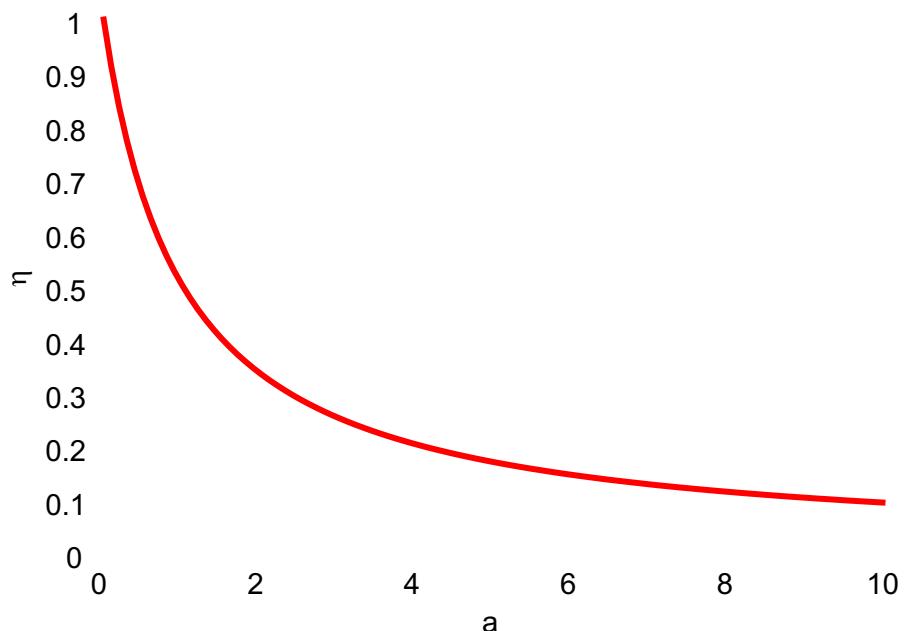
- $C=10 \text{ Mbit/s}$ $d=1 \text{ km}$ $v=200000 \text{ km/s}$ $F=100$
- $d/v=5 \cdot 10^{-6}$
- $C d/v = 10 \cdot 10^6 \cdot 5 \cdot 10^{-6} = 50$
- $a = 50 / 100 = 0,5$ $\eta = 1/1,5 = 0,67$

- $C=100 \text{ Mbit/s}$
- $C d/v = 500$
- $a=500/100=5$ $\eta = 1/6$

- $C=1 \text{ Gbit/s}$
- $C d/v = 5000$
- $a=5000/100 = 50$ $\eta = 1/51$

Quale efficienza per le LAN

- a determina le prestazioni della LAN
- Maggiore è la lunghezza del canale in trame, minore risulta il traffico massimo smaltibile (massimo throughput)
 - I protocolli ad accesso multiplo sono efficienti quando le distanze e le velocità di trasmissione sono abbastanza limitati





Protocollo a contesa: ALOHA

- È nato nel 1970 per collegare tra loro le università delle isole Hawaii.
- Prevede stazioni a terra ed un satellite geostazionario
 - Le stazioni trasmettono tutte sul medesimo canale radio (uplink)
 - Il satellite ritrasmette a terra amplificati i dati su un canale diverso (downlink)
- **CAP**
 - Quando un trasmettitore ha una trama da trasmettere la trasmette senza alcun verifica preventiva
 - La trama viene ritrasmessa dal satellite verso tutte le stazioni
 - La stazione trasmittente riceve la propria trama ed ha quindi conferma della corretta trasmissione
- **CRA**
 - Quando due stazioni trasmettono contemporaneamente i segnali collidono e si interferiscono sull'uplink
 - Il satellite scarta le trame non correttamente ricevute
 - La stazione non riceve la propria trama sul downlink e quindi identifica una collisione
 - Non ritrasmette subito ma fa partire l'algoritmo di **back-off**
 - Sceglie l'istante per la ritrasmissione in modo aleatorio all'interno di un intervallo di lunghezza prefissata T_b (tempo di back-off)



Aloha: prestazioni

- Assumiamo che i pacchetti generati dalle sorgenti di traffico (applicazioni) determinino gli arrivi di trame alle stazioni secondo un **processo di Poisson** con frequenza media di arrivo λ
 - Tenendo conto delle ritrasmissioni, il numero medio di pacchetti trasmessi in effetti al satellite nell'unità di tempo è $\lambda_r > \lambda$
 - Le collisioni con successive ritrasmissioni generano delle **correlazioni** fra gli arrivi, ma se l'intervallo di back off è abbastanza lungo rispetto a T ($T_b \gg T$), anche il traffico verso il satellite è **approssimativamente di Poisson**



Traffico offerto e smaltito

- Ipotesi:
 - Trame tutte uguali di lunghezza pari a T
- Traffico offerto dalle applicazioni

$$A_0 = \lambda T$$

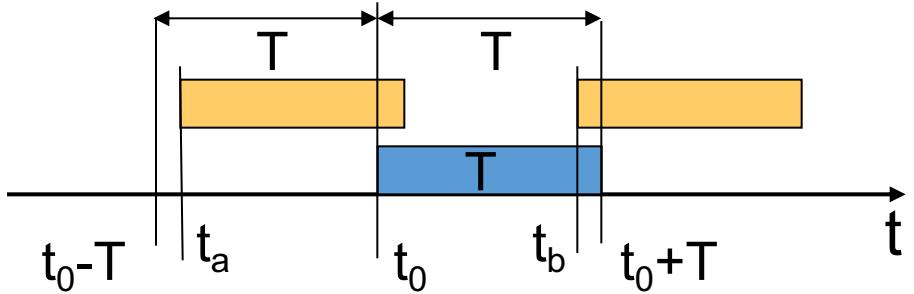
- Traffico offerto al MAC

$$G = \lambda_r T$$

- A causa delle collisioni $\lambda_r \geq \lambda$
- Il traffico smaltito è pari al traffico offerto che viene trasmesso senza collidere
 - Una trama viene trasmessa senza collidere con probabilità P_0

$$S = G P_0$$

Intervallo di vulnerabilità



- Si definisce **intervallo di vulnerabilità** T_v l'intervallo all'interno del quale una trasmissione può dar luogo a collisione
- Nel caso di ALOHA vale $T_v = 2 T$
 - La trama considerata inizia in t_0 e finisce in $t_0 + T$
 - Si ha collisione se
 - il primo bit della trama considerata si sovrapponga all'ultimo bit di una trama precedente
 - Il primo bit di una nuova trama si sovrapponga all'ultimo bit della trama considerata
 - Nessuna trama deve essere trasmessa per un tempo T prima di t_0 e per un tempo T successivo a t_0



Calcolo del Throughput

- La probabilità di non avere una trasmissione in $2T$ (probabilità di non collisione) è

$$P_0 = e^{-2\lambda_r T} = e^{-2G}$$

- Quindi il numero medio di trasmissioni aventi successo (traffico smaltito S) è pari a

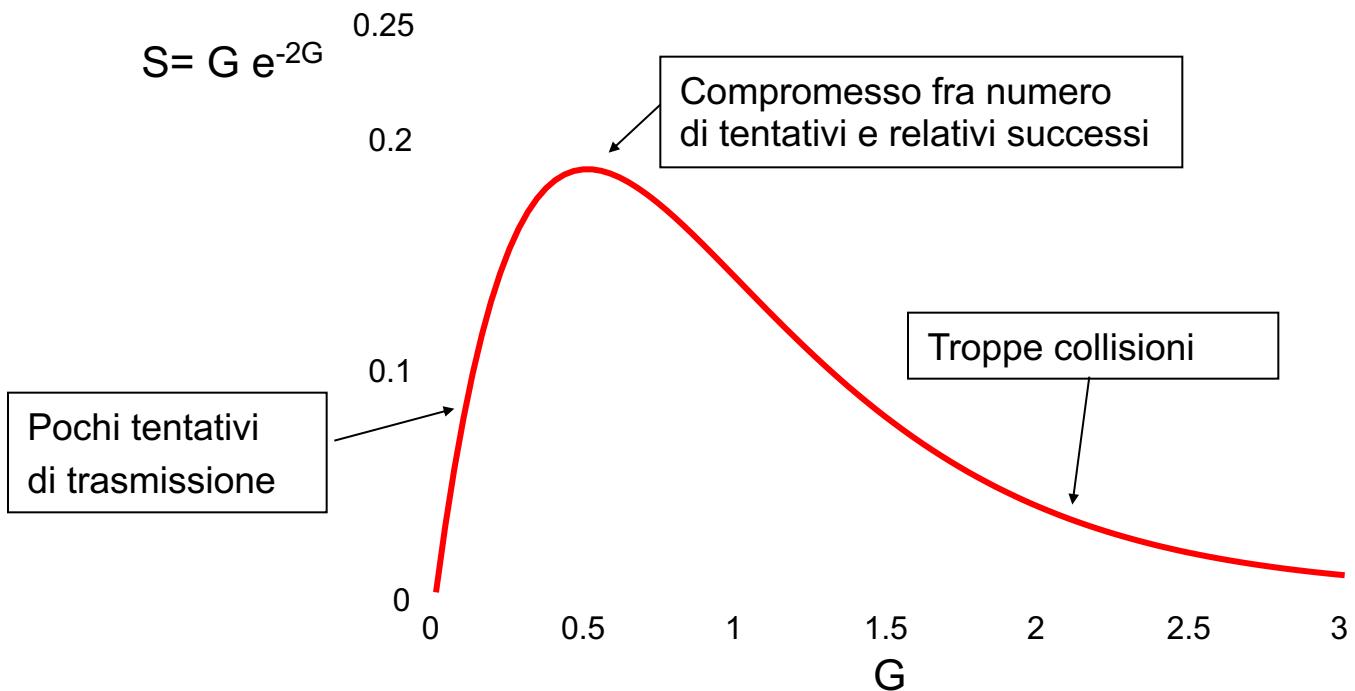
$$S = G e^{-2G}$$

- Valore massimo di S

$$S_{\max} = 1/(2e) \approx 0.18 \text{ per } G_{\max} = 0.5$$

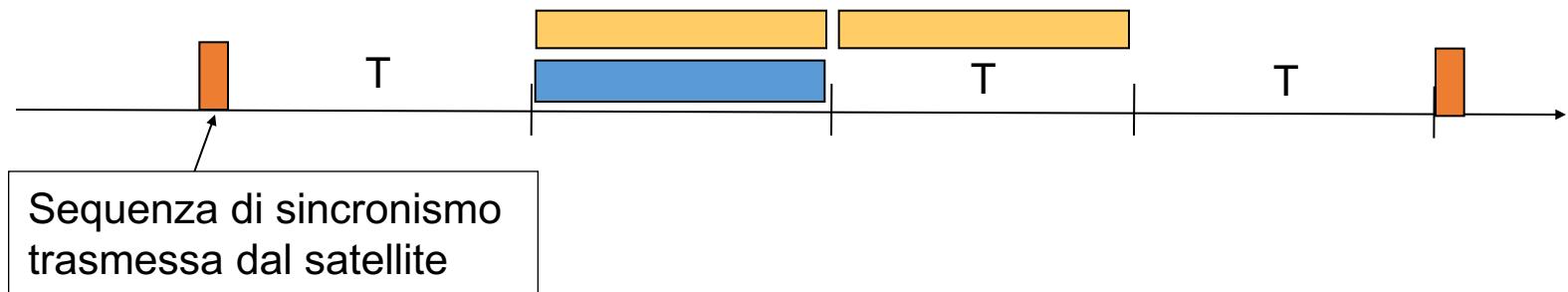
Aloha: throughput

- $S \approx G$ per piccoli valori di G
- $S \rightarrow 0$ per grandi valori di G



Slotted Aloha

- Un possibile miglioramento: **SLOTTED ALOHA**.
 - Il sistema lavora in modo **sincrono**: l'asse dei tempi viene diviso in intervalli (slot) di lunghezza T
 - Le trame vengono trasmesse in corrispondenza di istanti predefiniti



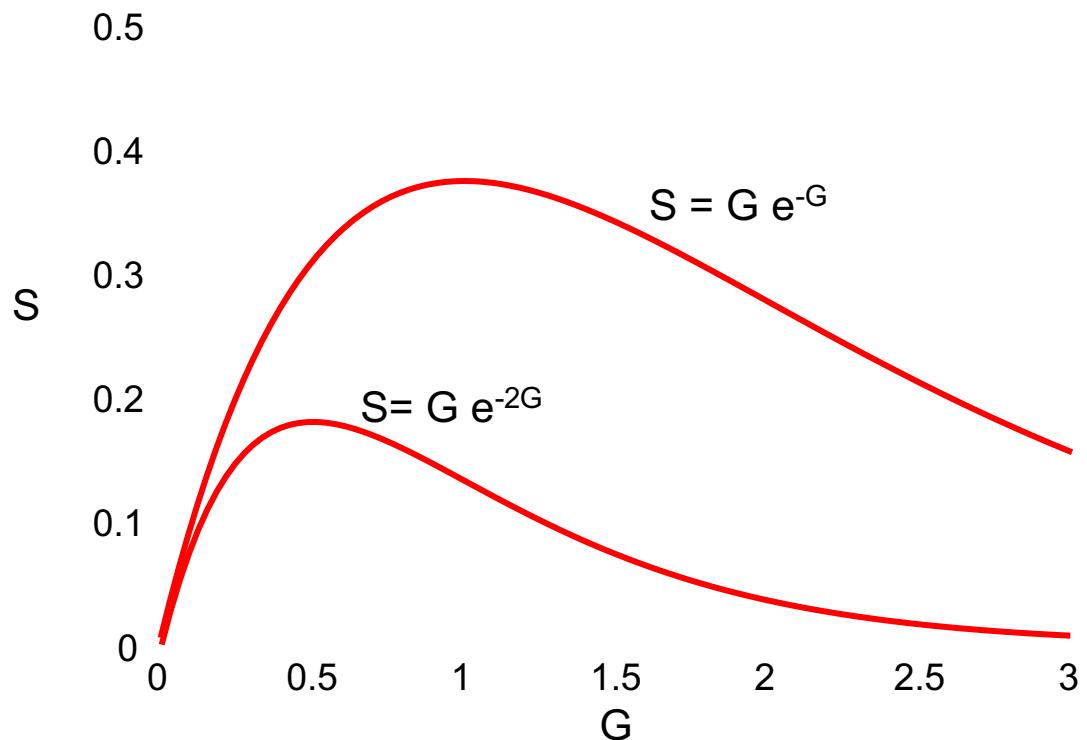
Prima di iniziare le trasmissioni la stazione deve acquisire il sincronismo, inviando trame di tentativo e rivelando come si posizionano rispetto agli slot

Due trame o si sovrappongono completamente o non si sovrappongono per nulla



Slotted Aloha

- L'intervallo di vulnerabilità si riduce a T
 - $P_0 = e^{-G}$
 - $S = Ge^{-G}$
 - il massimo di S vale $S_{\max} = 1/e \cong 0.36$ per $G_{\max} = 1$





Algoritmi di back-off

- Aloha classico
 - Sceglie a caso (con probabilità uniforme) il nuovo istante di trasmissione nell'intervallo 0 e T_b
 - Deve essere $T_b \gg T$ per rendere piccola la probabilità di una nuova collisione
- Aloha slotted
 - Si ritrasmette negli istanti di sincronismo, ci sono due alternative:
 - Si prende $T_b = n_b T$ e si sceglie un numero a caso fra 0 ed $n_b - 1$
 - Si ritrasmette nel primo slot utile con probabilità p_b e si passa allo slot successivo con probabilità $(1-p_b)$; ripetendo l'algoritmo ad ogni slot fino a che non si trasmette
- A parità di valore medio del tempo di ritrasmissione, queste due alternative danno prestazioni simili



Traffico offerto e smaltito

- In condizioni di equilibrio il traffico offerto al sistema deve essere eguale al traffico smaltito

$$A_0 = S$$

- Per effetto delle fluttuazioni statistiche del traffico, su brevi intervalli di tempo risulterà

$$A_0 \neq S$$

- Se $A_0 < S_{\max}$ la dinamica naturale del sistema tende a portarsi in equilibrio
- Se $A_0 > S_{\max}$ è impossibile raggiungere una situazione di equilibrio
 - I dati si accumulano nello strato superiore al MAC, in quanto una buona parte di essi non riesce mai ad essere trasmessa

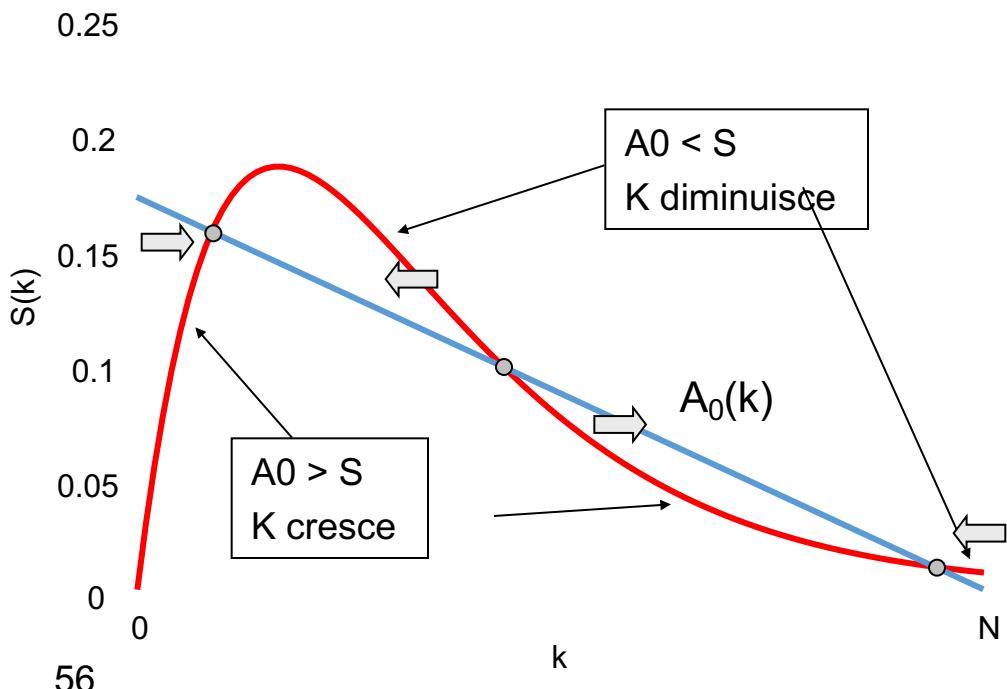


Numero di stazioni finito e stabilità

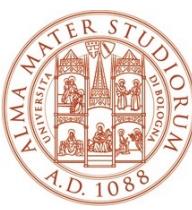
- Il traffico offerto dipende dal numero di stazioni backlogged k
 - Sia λ_i la frequenza media di arrivo delle trame da ciascuna stazione
 - Se la stazione non è backlogged: $\lambda_i = \lambda$
 - Se la stazione è backlogged non invia nuove trame: $\lambda_i = 0$
- Con k stazioni backlogged il traffico offerto vale
$$A_0 = \lambda T(N-k)$$
- Inoltre
 - $G(k)$ è una funzione monotona di k
 - $S(G(k))$ ha una forma simile a quella già evidenziata
- I valori di k che garantiscono condizioni di stabilità si trovano determinando k tale che $A_0(k) = S(k)$

Stabilità

- Se $A_0 > S$ si accumula traffico \Rightarrow si hanno collisioni \Rightarrow le stazioni backlogged aumentano $\Rightarrow A_0$ cala e k cresce
- Se $A_0 < S$ si smaltisce più traffico di quello nuovo in arrivo \Rightarrow si trasmettono trame che hanno colliso in precedenza \Rightarrow le stazioni backlogged calano $\Rightarrow A_0$ cresce e k cala



- 3 punti di stabilità
 - 2 sono stabili
 - 1 instabile
- Un aumento eccessivo di A_0 può portare ad una situazione di troppe collisioni



Controlled Aloha

- Per ovviare al problema dell'instabilità si possono usare varie tecniche
- Una delle più semplici è fare crescere il tempo di back-off
 - Alla prima collisione si pone $T_b = T_0$
 - Se la trama ritrasmessa collide di nuovo si pone $T_b = 2T_0$ e si continua a raddoppiare T_b ad ogni nuova collisione
 - Quando la trasmissione ha successo si ritorna a $T_b = T_0$
 - Nel caso slotted si puo dimezzare p_b ad ogni collisione
- Questo algoritmo si dice **back-off esponenziale** e si può dimostrare che elimina l'instabilità
 - Può fare sorgere problemi di fairness: una stazione che ha subito molte collisioni viene tagliata fuori dalle trasmissioni



Derivati del protocollo Aloha

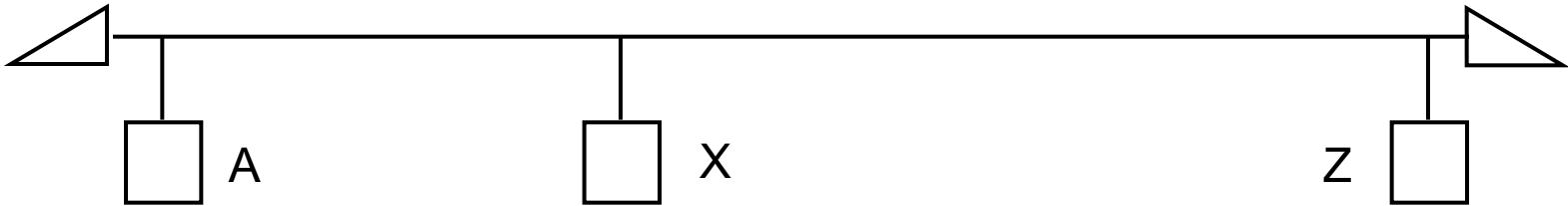
- Il protocollo Aloha può essere implementato su qualunque mezzo trasmissivo e qualunque topologia
- Ha una efficienza piuttosto bassa ma è circa quanto di meglio si può fare quando i ritardi di propagazione sono grandi come nel caso del satellite
- Se lo si vuole applicare ad una rete locale conviene sfruttare la conoscenza che ogni stazione può acquisire sull'attività delle altre
- Nasce così il protocollo **CSMA Carrier Sensing Multiple Access**
 - Viene proposto su una topologia a Bus bidirezionale
 - È ancora un protocollo ad accesso casuale a contesa



CSMA: Carrier Sensing Multiple Access

- **Carrier sensing**
 - Ogni stazione che debba trasmettere **rivela** presenza di segnale sul bus e trasmette solo se è libero
 - Se il bus è occupato si aspetta la fine della trama e poi
 - Si trasmette (caso 1 persistent)
 - Si fa partire l'algoritmo di back off (caso non persistent o 0 persistent)
 - Si trasmette con probabilità p e si fa partire l'algoritmo di back off con probabilità $(1-p)$ (caso p persistent)
- Una volta iniziata la trasmissione, i dati inviati da una stazione **possono collidere** con quelli di un'altra
 - Questo avviene a causa del ritardo di propagazione non nullo
 - Sul bus non c'è un meccanismo immediato di rivelazione delle collisioni: occorre affidarsi a un sistema di Acknowledgement
- L'algoritmo di back-off può essere come quello dell'Aloha con $T_b \gg 2\tau$

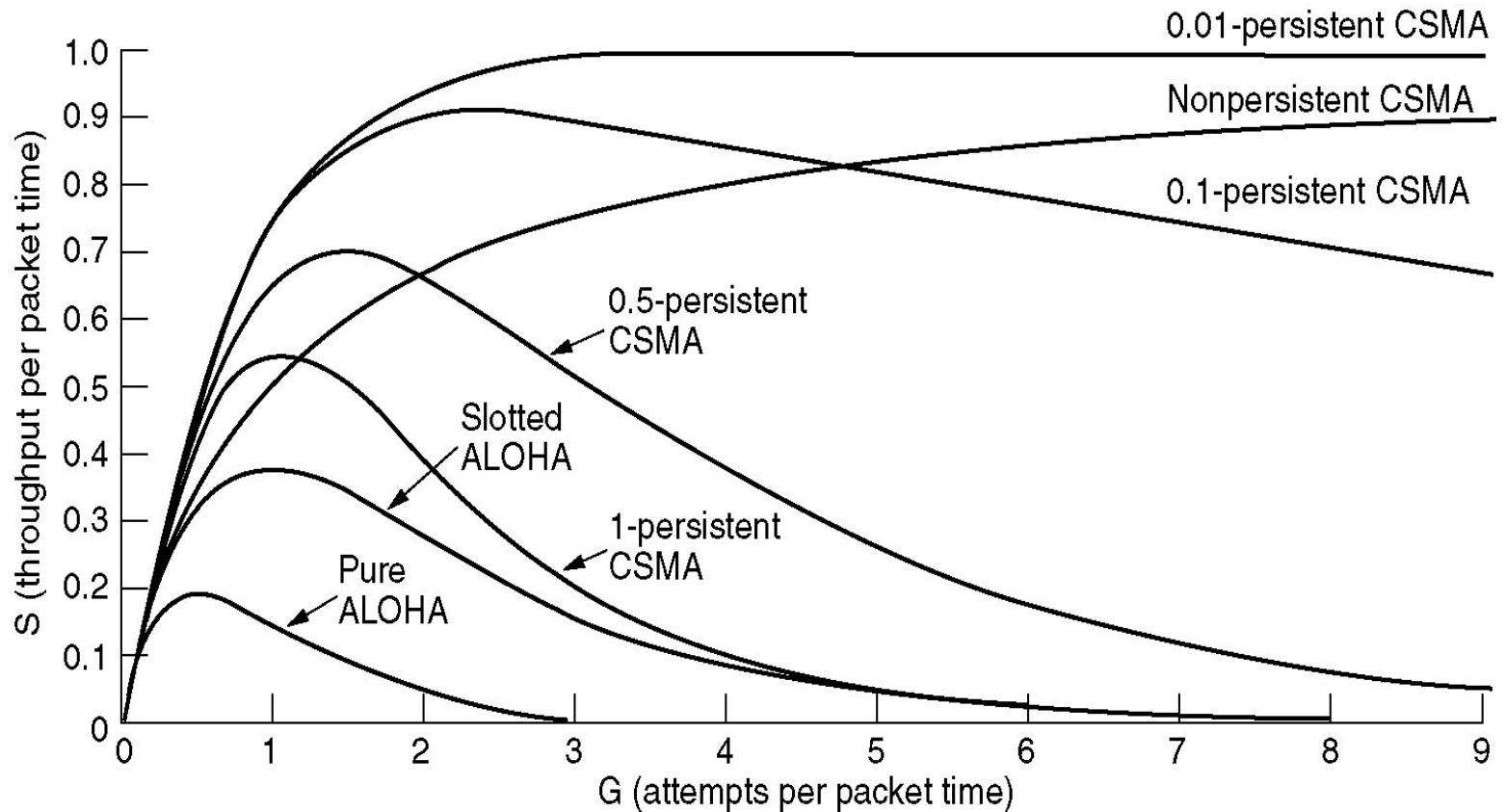
CSMA: intervallo di vulnerabilità



- Chiamiamo A e Z le due stazioni più distanti sul Bus e τ il **tempo di propagazione** fra di loro + il tempo necessario per rivelare il segnale
- A esegue il carrier sensing nell'istante t_A
 - Se Z fa carrier sensing fra t_A e $t_A + \tau$ non rileva attività e può quindi anch'essa iniziare a trasmettere: *si ha collisione*
 - Analogamente se Z ha trasmesso fra t_A e $t_A - \tau$ A non rileva il segnale di Z e trasmette in t_A : *si ha collisione*
- L'intervallo di vulnerabilità vale 2τ
- Le prestazioni sono tanto migliori dell'Aloha quanto più $\tau/T < 1$
 - In generale le prestazioni dipendono anche dal valore di p

Persistent e Nonpersistent CSMA

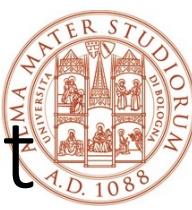
Utilizzazione del canale per Aloha e CSMA





Versione slotted e problemi di stabilità

- Anche per il CSMA esiste la **versione slotted**
- In questo caso la misura più opportuna del **tempo di slot** è τ
- L'intervallo di vulnerabilità vale τ invece che 2τ
- Anche per il CSMA come per tutti i protocolli a contesa ci sono problemi di stabilità
- Si può usare un algoritmo di back-off esponenziale



CSMA/CD: CSMA con Collision Detect

- Un miglioramento del CSMA è stato proposto da **Metcalfe** nel 1976
- **Collision Detection:**
 - Una stazione è in grado di rilevare l'avvenuta collisione *rimanendo in ascolto* sul mezzo mentre trasmette
 - E' un processo analogico basato sulla **rilevazione di potenza** sul canale (facilitato anche dalla codifica di Manchester adottata)
- In caso di collisione:
 - si ferma subito la trasmissione
 - si invia una particolare sequenza di bits (**jamming**) per informare tutte le altre stazioni dell'avvenuta collisione

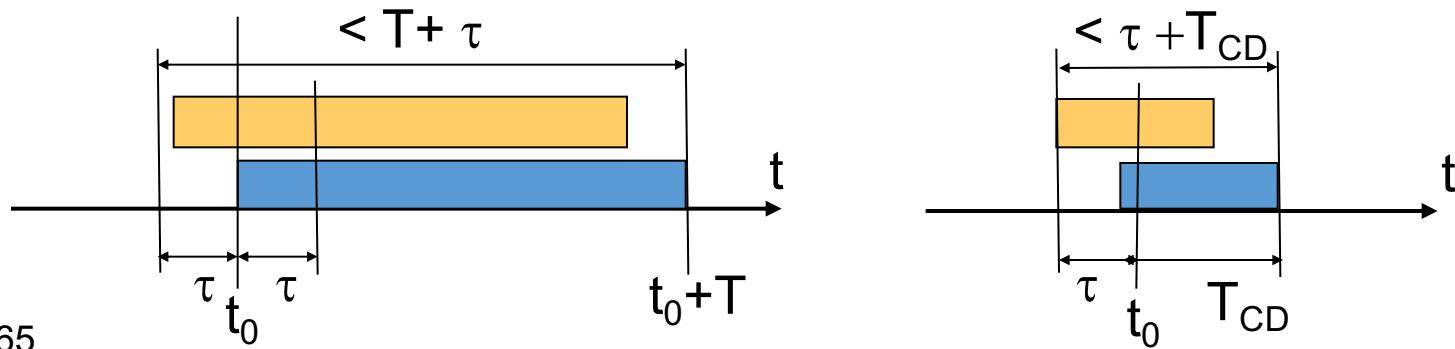


Codifica di Manchester

- Rappresentazione dei bit
 - “0” logico: segnale basso (-0.85 Volt) per mezzo tempo di simbolo e segnale alto (+0.85 Volt) per l’altro mezzo
 - “1” logico: segnale alto per mezzo tempo di simbolo e poi segnale basso
- Vantaggi
 - Una transizione al centro di ogni bit, che può essere rivelata mediante un derivatore, facilita
 - L’acquisizione del sincronismo
 - Il carrier sensing
 - Il collision detection
 - Sono disponibili simboli (alto alto e basso basso) per rappresentare *non dati*
- Svantaggi
 - Per trasmettere a 10 Mbit/s occorre un clock a 20 MHz
- Il protocollo CSMA/CD con codifica di Manchester è stato adottato nella rete Ethernet, standard di mercato per le LAN

Cosa migliora nel CSMA-CD

- Nei casi in cui avviene collisione
 - Nel CSMA le stazioni continuano la trasmissione dell'intera trama
 - Il canale rimane impegnato inutilmente per un intervallo di tempo all'incirca pari a T
 - Nel CSMA/CD
 - Al più il canale rimane impegnato inutilmente al più per la somma di
 - Un intervallo di vulnerabilità ()
 - Il tempo necessario a rilevare la collisione più il tempo della sequenza di Jamming (T_{CD})





ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Protocolli di Trasporto

Franco CALLEGATI

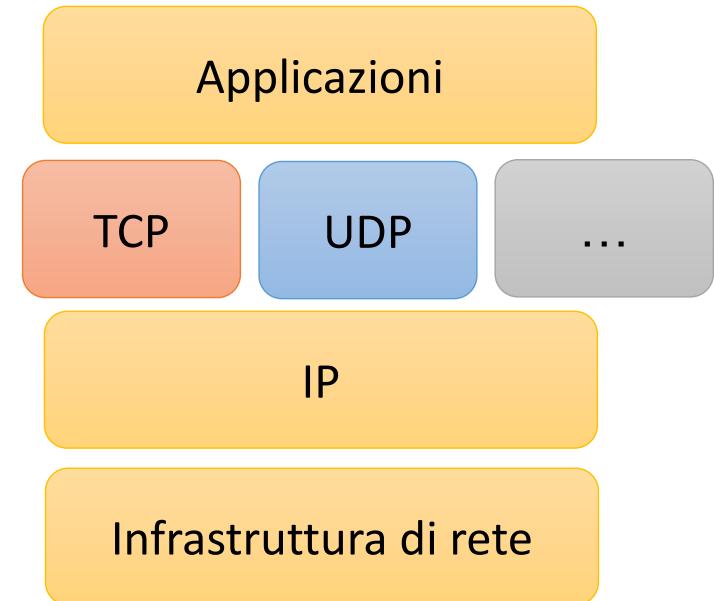
Dipartimento di Informatica: Scienza e Ingegneria



Trasporto in Internet

- Architettura protocollare tradizionale di Internet:

- Tipologia di servizio: dati
- Due protocolli di trasporto:
 - TCP: connection oriented
 - UDP: connectionless



- Nuovi servizi multimediali:

- Emergono problematiche di trasporto real time
- Vengono definiti nuovi protocolli di trasporto
 - RTP, RTCP



Funzioni dello strato di trasporto

- Consente la **Multiplazione**:
 - Permette a più processi applicativi di utilizzare le sue funzioni di comunicazione in contemporanea
- Utilizza il numero di porta per distinguere flussi dati di applicazioni diverse
- Controlla il comportamento del canale di comunicazione end-to-end
 - L'obiettivo è quello di garantire la qualità del trasferimento dati a livello di trasporto richiesto dall'applicazione



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

User Datagram Protocol



User Datagram Protocol (UDP)

- Protocollo “connectionless”
 - Non esiste il concetto di “connessione”
 - Ogni messaggio è indipendente da tutti gli altri
- Pensato per
 - Invio di blocchi dati di limitate dimensioni
 - Comunicazione fra applicazioni che non richiede un controllo della qualità del trasporto
 - Esempi: e-mail, DNS, ...



Il messaggio UDP

Screenshot of Wireshark showing a DNS query from 192.168.10.199 to 137.204.59.1. The packet details for the DNS query (No. 10) are highlighted and expanded.

```

No. Time           Source          Destination        Protocol Length Info
1 0.000000 192.168.10.199 137.204.59.1   DNS      Standard query A deisnet.deis.unibo.it
2 0.001714 137.204.59.1 192.168.10.199  DNS      Standard query response CNAME deis85.deis.unibo.it A 1
3 0.002763 192.168.10.199 137.204.57.85  TCP      nim > http [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_P
4 0.003487 137.204.57.85 192.168.10.199  TCP      http > nim [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1
5 0.003512 192.168.10.199 137.204.57.85  TCP      nim > http [ACK] Seq=1 Ack=1 Win=65535 Len=0
6 0.003732
7 0.005748
8 0.009821
9 0.011087
10 0.011148
11 0.014612
12 0.015858

```

The expanded details for the DNS query (No. 10) are:

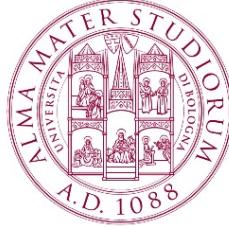
- Frame 1: 81 bytes on wire (648 bits), 81 bytes captured (648 bits)
- Ethernet II, Src: DellComp_89:b3:e9 (00:06:5b:89:b3:e9), Dst: DellComp_ec:46:62 (00:b0:d0:ec:46:62)
- Internet Protocol, Src: 192.168.10.199 (192.168.10.199), Dst: 137.204.59.1 (137.204.59.1)
- User Datagram Protocol, Src Port: startron (1057), Dst Port: domain (53)
 - Source port: startron (1057)
 - Destination port: domain (53)
 - Length: 47
 - Checksum: 0x17eb [validation disabled]
- Domain Name System (query)

The bytes of the DNS query are shown below:

0000 00 b0 d0 ec 46 62 00 06 5b 89 b3 e9 08 00 45
0010 00 43 01 83 00 00 80 11 00 00 c0 a8 0a c7 89 c0
0020 3b 01 04 21 00 35 00 2f 17 eb 76 46 01 00 00 01
0030 00 00 00 00 00 00 07 64 65 69 73 6e 65 74 04 64
0040 65 69 73 05 75 6e 69 62 6f 02 69 74 00 00 01 00
0050 ..

A callout diagram illustrates the structure of a UDP datagram:

- 32 bit / 4 byte
- IP header
- Source Port Destination Port
- Length Checksum
- User data



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Transmission Control Protocol



Il segmento TCP

- TCP incapsula i dati delle applicazioni in pacchetti detti “**segmenti**”
- Il segmento TCP prevede
 - Un header standard di 20 byte
 - Un header variabile per negoziare delle opzioni
 - Un payload di dimensione variabile contenente i dati di applicazione
- Il segmento TCP ha una dimensione massima detta Maximum Segment Size (MSS)
 - MSS corrisponde alla massima dimensione del blocco dati di applicazione che può essere contenuto nel segmento



Formato del segmento TCP (1)

32 bit

Source Port		Destination Port	
Sequence number			
Acknowledge number			
TCP header length	Reserved	U R G A C K P C S H T R S Y N F I	Window
Checksum		Urgent Pointer	
Opzioni		Padding	
Dati			



Formato del segmento TCP (2)

- **Source (Destination) port:** numero della porta sorgente (destinazione)
- **Sequence number:** numero di sequenza del primo byte del pacchetto; se è presente il bit SYN questo è il numero di sequenza iniziale su cui sincronizzarsi
- **Acknowledge number:** se il bit ACK è a 1 allora questo numero contiene il numero di sequenza del blocco di dati che il ricevitore si aspetta di ricevere
- **TCP Header Length (4 bit):** numero di parole di 32 bit dell'intestazione TCP; indica dove iniziano i dati
- **Reserved:** sei bit riservati per uso futuro



Formato del segmento TCP (3)

- **Control bit:** sono 6 bit di controllo
 - **URG** posto a 1 se si deve considerare il campo Urgent Pointer
 - **ACK** posto a 1 se si deve considerare il campo Acknowledge
 - **PSH** posto a 1 serve per la funzione di push
 - **RST** posto a 1 per resettare la connessione
 - **SYN** posto a 1 per sincronizzare i numeri di sequenza
 - **FIN** posto a 1 per indicare la fine dei dati





Formato del segmento TCP (4)

- **Window**: finestra del ricevitore, cioè il numero di byte che il ricevitore è disposto a ricevere, partendo dal numero di sequenza di quello contenuto nel campo acknowledge
- **Checksum**: controllo dell' errore sul segmento
- **Urgent Pointer**: contiene puntatore a dati urgenti eventualmente presenti nel pacchetto (es. per abortire programma remoto in esecuzione), ha senso se il bit URG è posto ad 1
- **Options**: contiene opzioni per la connessione
- **Padding**: bit aggiuntivi per fare in modo che l' intestazione sia multipla di 32 bit



Dimensioni del segmento TCP

- MSS deve
 - Essere inferiore alla massima dimensione del payload IP meno un header TCP
 - $2^{16} = 65535 - 20 - 20 = 65495$ byte
 - Rispettare i limiti imposti ai pacchetti dalle reti che deve attraversare e che hanno una Maximum Transfer Unit (MTU)
 - Un tipico valore per MTU sono i 1500 byte imposti da Ethernet
- MSS dipende dall' implementazione
 - Normalmente è configurabile
- In generale non è possibile sapere la MTU di ogni rete intermedia che va attraversata
 - La rete ha il meccanismo per la frammentazione dei pacchetti
 - Questo può condurre a inefficienza, soprattutto in caso di una grande quantità di dati da trasmettere su una connessione
 - È stato definito un algoritmo detto “Path MTU discovery” (RFC 1191) basato su ICMP

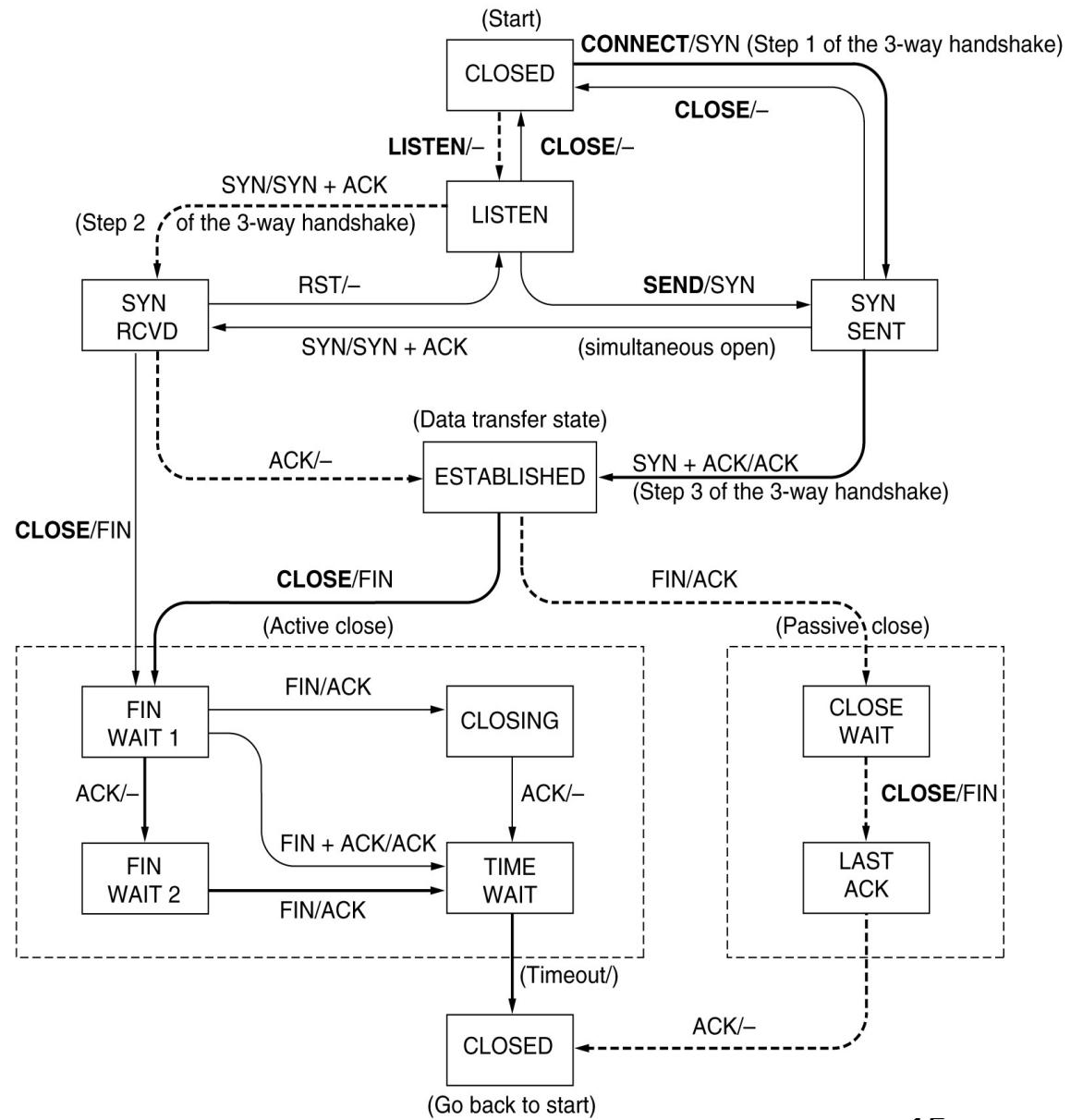


Checksum

- Viene calcolato applicando Internet Checksum a
 - Pseudo-header (pseudo-intestazione)
 - Indirizzi IP sorgente e destinatario
 - Protocol
 - Lunghezza in byte del segmento TCP (payload IP)
 - Non viene trasmessa ma viene calcolata in trasmissione e in ricezione
 - Intestazione TCP
 - Con campo checksum posto a 0
 - Dati del segmento TCP
 - Se il numero di byte è dispari viene aggiunto un byte di padding con tutti i bit a 0
 - Il padding non viene trasmesso

La macchina a stati finiti del TCP

- Linee tratteggiate
 - Azioni tipiche di un server
- Linee nere
 - Azioni tipiche di un client
- Linee chiare
 - Eventi inusuali
- Transizioni
 - Causa/effetto



Da A.S. Tanenbaum, "Reti di Calcolatori"



Definizioni

- **Stato:** Le condizioni che descrivono il software del protocollo in un particolare calcolatore in un determinato istante.
- **Transizione:** Il passaggio da uno stato ad un altro.
- **Evento:** Qualunque cosa che provochi una transizione di stato per il protocollo.
- **Azione:** Qualcosa che il software del protocollo fa in un dispositivo come risposta ad un evento prima di effettuare un transizione di stato

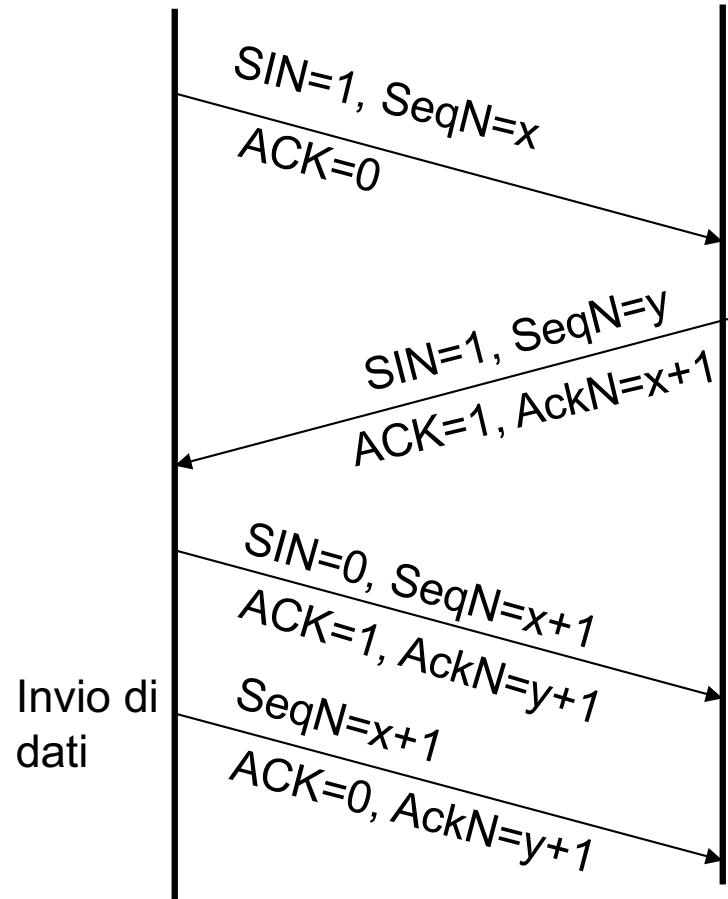


Dialogo su rete inaffidabile

- Se il mezzo di comunicazione è inaffidabile risulta sostanzialmente impossibile avere uno scambio di informazioni con conferma certa
 - Problema logico delle 3 armate
 - A invia un messaggio e B lo conferma
 - Se A non riceve la conferma non può sapere se B abbia ricevuto il messaggio o meno
 - Perdita del messaggio o della conferma?
 - Il ragionamento si può proseguire sulla conferma della conferma ecc.
- È necessario decidere dove fermarsi per raggiungere un determinato grado di affidabilità



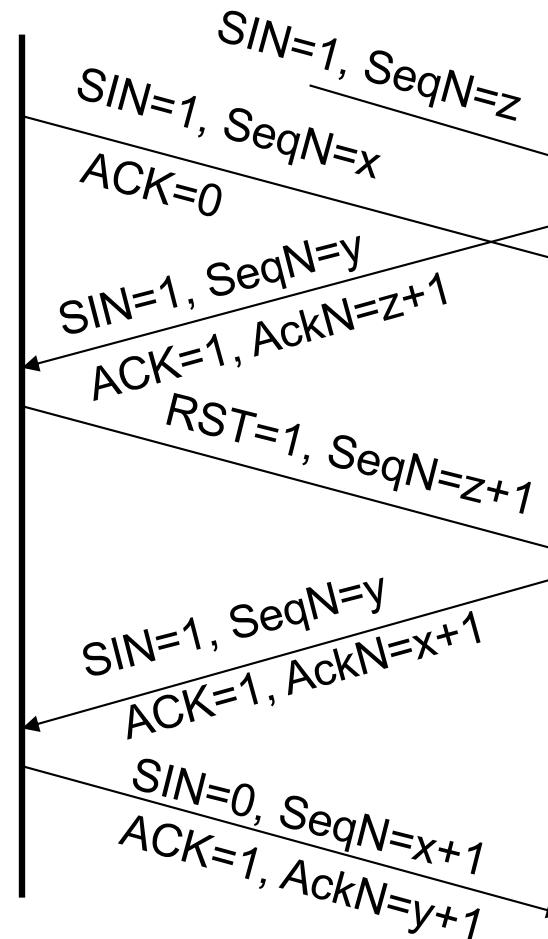
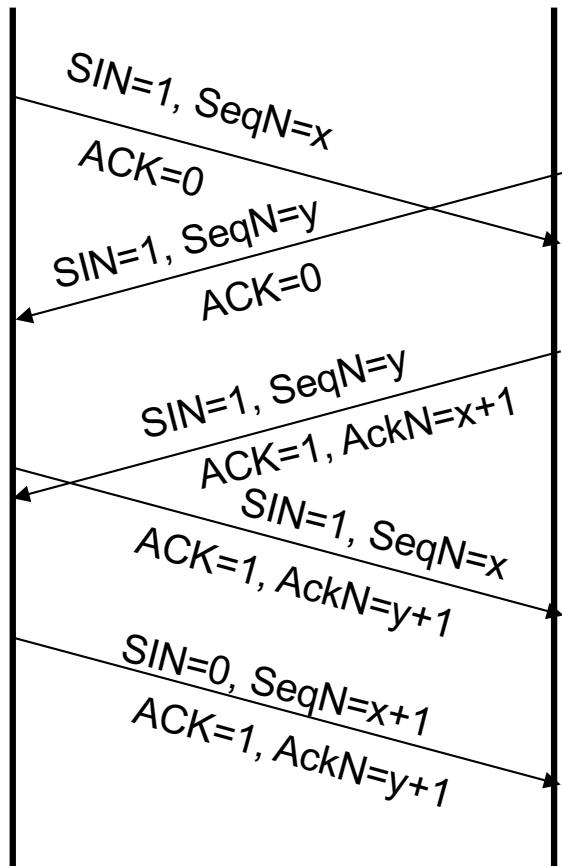
Apertura della connessione TCP



- L' apertura della connessione è critica
 - I segmenti di segnalazione possono essere persi, duplicati e ritardati
- **Three ways handshake**
 - Si è dimostrato molto robusto alla prova dei fatti
- Utilizza sinergicamente i bit di flag e quelli di numerazione
 - Si noti che il primo pacchetto dati ha numero di sequenza uguale all' ACK precedente (ACK non occupa spazio di numerazione)

Caratteristiche del TWH

- Il three-ways handshake
 - resiste alla instaurazione contemporanea di due connessioni
 - ignora pacchetti di apertura ritardatari





Il problema delle «incarnazioni»

- Connessione fra Host A e Host B
 - Host A viene riavviato
 - La connessione viene di fatto chiusa in modo errato
 - La connessione rimane attiva in Host B
- Dopo il riavvio l' applicazione vuole riprendere il dialogo e riprova ad aprire la connessione con Host B
 - Cosa accade se decide di utilizzare i medesimi numeri di porta (ossia riavviare la medesima connessione)
- Esistono a breve distanza di tempo diverse “**incarnazioni**” della medesima connessione logica



Ripresa dopo un problema

- Il three ways handshake per una nuova incarnazione
 - L'host B ritiene che la connessione precedente sia ancora aperta
 - L'ambiguità viene risolta

TCP A	TCP B
1. (CRASH)	(send 300, receive 100)
2. CLOSED	ESTABLISHED
3. SYN-SENT --> <SEQ=400><CTL=SYN>	--> (??)
4. (!!) <-- <SEQ=300><ACK=100><CTL=ACK>	<-- ESTABLISHED
5. SYN-SENT --> <SEQ=100><CTL=RST>	--> (Abort!!)
6. SYN-SENT	CLOSED
7. SYN-SENT --> <SEQ=400><CTL=SYN>	-->



Chiusura della connessione TCP

- **Soft release**
 - Il TCP cerca di realizzare la chiusura ordinata della connessione, garantendo che non vadano persi dati
- Anche questo problema non può essere risolto in modo rigoroso su una rete inaffidabile
- TCP sceglie di realizzare la chiusura con modalità “simplex”
 - Le due direzioni vengono rilasciate in modo **indipendente**
 - Il TCP che intende terminare la trasmissione emette un segmento con **FIN=1**
 - Quando questa entità riceve l’ Ack la direzione si considera chiusa
 - Se dopo un certo tempo non arriva l’ Ack il mittente del FIN rilascia comunque la connessione
 - L’ altra direzione può continuare a trasmettere dati finché non decide di chiudere



Esempio di chiusura normale

- TCP A decide di chiudere
 - Inizia la procedura inviando un segmento con FIN=1
- TCP B rileva la richiesta di chiusura
 - Procede anche lui all' invio di un segmento con FIN=1

TCP A		TCP B
1. ESTABLISHED		ESTABLISHED
2. (Close)		
FIN-WAIT-1	--> <SEQ=100><ACK=300><CTL=FIN, ACK>	--> CLOSE-WAIT
3. FIN-WAIT-2	<-- <SEQ=300><ACK=101><CTL=ACK>	<-- CLOSE-WAIT
4.		(Close)
TIME-WAIT	<-- <SEQ=300><ACK=101><CTL=FIN, ACK>	<-- LAST-ACK
5. TIME-WAIT	--> <SEQ=101><ACK=301><CTL=ACK>	--> CLOSED
6. (2 MSL)		
CLOSED		



Chiusura contemporanea

- TCP A e B decidono in contemporanea di chiudere la connessione
 - La procedura funziona correttamente anche in questo caso

TCP A		TCP B
1. ESTABLISHED		ESTABLISHED
2. (Close)		(Close)
FIN-WAIT-1	--> <SEQ=100><ACK=300><CTL=FIN, ACK>	... FIN-WAIT-1
	<-- <SEQ=300><ACK=100><CTL=FIN, ACK>	<--
	... <SEQ=100><ACK=300><CTL=FIN, ACK>	-->
3. CLOSING	--> <SEQ=101><ACK=301><CTL=ACK>	... CLOSING
	<-- <SEQ=301><ACK=101><CTL=ACK>	<--
	... <SEQ=101><ACK=301><CTL=ACK>	-->
4. TIME-WAIT (2 MSL)		TIME-WAIT (2 MSL)
CLOSED		CLOSED



Svolgimento del dialogo

- Protocollo ARQ per rendere affidabile il dialogo
- Rispetto ai protocolli di linea TCP affronta un problema molto più complesso
 - Le caratteristiche dei ricevitori possono essere estremamente variabili
 - I segmenti sono trasmessi utilizzando una rete connectionless e quindi non è garantito l'arrivo in sequenza
 - I percorsi in rete e le condizioni di congestione possono variare e quindi il ritardo di propagazione non è costante
 - La banda del canale dipende dal percorso



Lo stato ESTABLISHED

- Nello stato si trasferiscono i dati utilizzando il protocollo ARQ
 - Si numerano sequenzialmente i segmenti
 - Si conferma la corretta ricezione dei segmenti
 - Il recupero degli errori avviene tramite ritrasmissione
 - Il meccanismo di ritrasmissione è una sorta di Go-back-N modificato



Numerazione in TCP

- Per avere la massima **flessibilità** si sceglie di assegnare un numero non ai segmenti ma **ai singoli byte** trasportati nei segmenti
 - I dati trasportati sono pensati come un **unico flusso (stream)** di byte (**byte stream**)
 - Si comincia a numerare da un numero x scelto all'atto dell'apertura della connessione
 - Il campo Seq. number numera il primo byte del segmento
- La conferma di avvenuta ricezione viene data mettendo nel campo **Ack. Number** il numero del byte successivo all'ultimo ricevuto
 - primo byte che ci si aspetta di ricevere



Numerazione duplicata

- Possono facilmente avversi segmenti ritardati o duplicati
 - All' istante t_0 viene inviato il segmento S_0 con numero di sequenza X
 - S_0 viene duplicato nella rete
 - Una copia viene correttamente ricevuta e confermata
 - Una copia viene ritardata rispetto alle altre
 - La trasmissione continua
 - Si esaurisce lo spazio di numerazione e quindi si riutilizzano numeri già usati
 - All' istante $t_k > t_0$ viene inviato il segmento S_k che ha anch' esso numero di sequenza X
 - La copia ritardata di S_0 arriva poco prima di S_k
 - viene interpretata come segmento valido



Maximum Segment Lifetime

- I numeri di sequenza possono essere riutilizzati?
 - Solo se si è sicuri che non esistano più in rete vecchi segmenti numerati con tali numeri
- Lo spazio di numerazione finito rende necessario limitare il tempo di vita dei segmenti
- Il massimo tempo di vita dei segmenti (Maximum Segment Lifetime o MSL) deve essere noto
 - $MSL = 2 \text{ min}$ (RFC 793)
 - Su un collegamento a 2 Megabits/sec servono 4.5 ore per esaurire lo spazio di numerazione di 2^{32} ottetti
 - Su un collegamento a 100 Megabits/sec servono 5.4 minuti per esaurire lo spazio di numerazione di 2^{32} ottetti
 - Cosa accade per reti con velocità del Gbit/s e oltre?



Inizializzazione della sequenza

- All'apertura della connessione si deve scegliere il numero di sequenza iniziale (Initial Sequence Number o ISN)
 - Numero prefissato uguale per tutti
 - Numero puramente casuale
 - Numero legato al valore di un contatore
- ISN
 - Deve garantire che non ci sia duplicazione nell' uso dei numeri di sequenza
 - Qualora non sia prefissato e costante deve essere concordato fra i due host che aprono la connessione

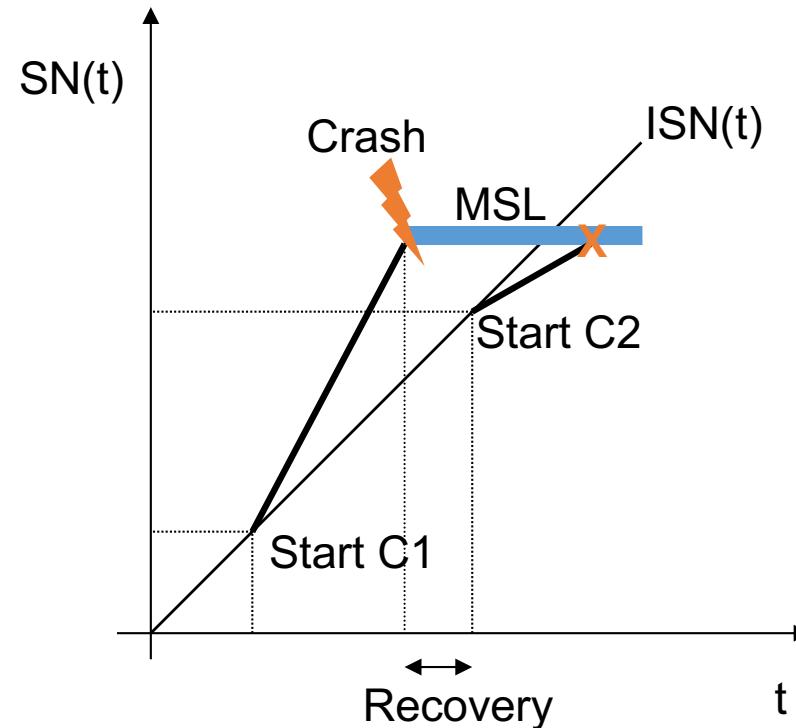
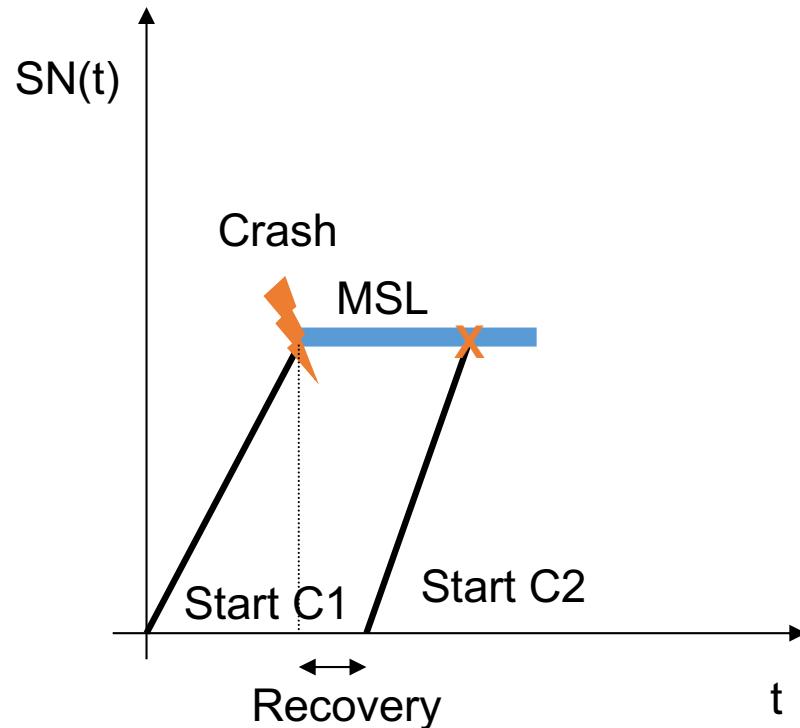


Se qualcosa va male

- Un host viene riavviato a causa di un problema?
 - Vengono ricreate nuove “*incarnazioni*” di vecchie connessioni
 - Vengono scelti nuovi ISN
 - Una nuova incarnazione può finire per usare numeri già utilizzati dall’ incarnazione precedente
 - Segmenti ritardati della prima incarnazione possono essere ricevuti dalla nuova incarnazione con numero di sequenza apparentemente corretto

Esempio

- ISN sempre uguale
 - $\text{ISN} = 0$
- $\text{ISN}(t) = t$
 - ISN uguale al valore di un contatore





ISN nella RFC 793

- ISN è funzione del tempo utilizzando un sistema di conteggio
 - Contatore a 32 bit
 - Incrementato ogni 4 μ sec
 - Il contatore ripete la sequenza ogni 2^{32} 4 μ sec = 4.77 h
- TCP quiet time
 - Dopo un qualunque riavvio un host attende almeno un MSL prima di riaprire le connessioni TCP

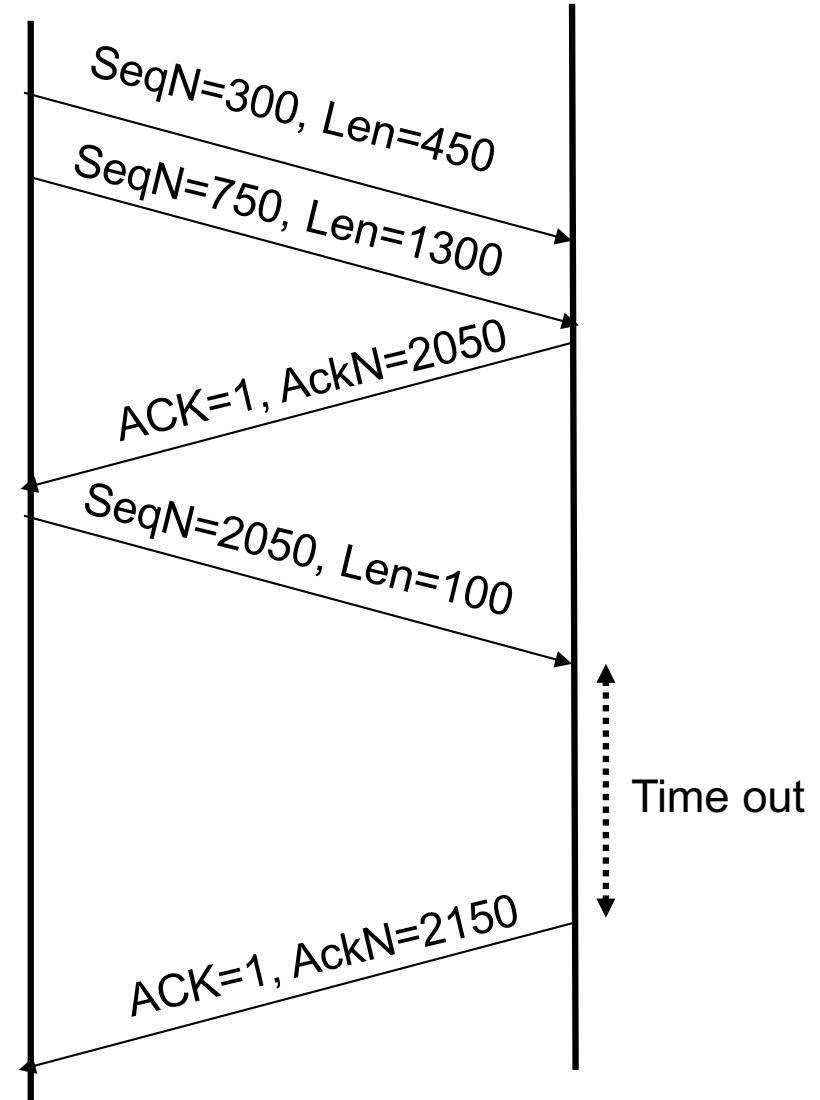


Messaggi di conferma (ACK)

- Gli Ack sono **cumulativi**
- In piggybacking
 - La conferma ha la forma di un normale messaggio TCP con il flag **ACK=1**
 - Può contenere dati oppure no
 - Se non contiene dati ne risulta un datagramma IP di 40 byte
 - 20 byte = min. header IP + 20 byte = min. header TCP
- Di default la conferma è esplicita
 - Il ricevitore trasmette un ACK per ogni segmento ricevuto
- ACK **ritardati**
 - Al momento della ricezione corretta di un segmento il ricevitore può inviare subito un ACK oppure ritardarlo
 - L' obiettivo è quello di minimizzare il numero di ACK
 - Se si ritarda troppo possono scattare i time-out

ACK ritardati

- Un ACK e' ritardato fino a che
 - è stato ricevuto un ulteriore pacchetto
 - Scatta un time out
 - Uguale a 200 ms nelle principali implementazioni
- Si riduce il traffico di ack
- Normalmente si produce un ACK ogni due segmenti





In ricezione

- Il ricevitore ha ricevuto fino a $\text{SeqN} = N$
 - Attende un segmento con $\text{SeqN} = N+1 \bmod M$
 - Riceve un segmento con $\text{SeqN} = X \neq N+1 \bmod M$
 - Se X è precedente ad N il segmento viene considerato un duplicato ritardato e viene scartato
 - Se X è successivo ad N il segmento è fuori sequenza (manca qualcosa)
 - Cosa può essere accaduto?
 - Uno o più segmenti sono andati persi
 - Un segmento trasmesso dopo un altro lo ha superato a causa dei diversi percorsi possibili e dei ritardi variabili in rete
 - Cosa fare?
 - Il ricevitore memorizza il segmento se X è entro W_R
 - Ritrasmette l'ultima conferma inviata (ACK duplicato)
 - Quando riceve $\text{seqN} = N$ a completamento della sequenza conferma $\text{ackN} = X+1$



In trasmissione

- Il trasmettitore invia il segmento $\text{seqN} = N$
 - Se riceve un ACK con $\text{ackN} = N+1$ toglie il segmento dalla memoria e fa scorrere la finestra di trasmissione
 - Se non riceve $\text{ackN} = N+1$ allo scadere di RTO ritrasmette il segmento
 - Ignora eventuali ACK duplicati

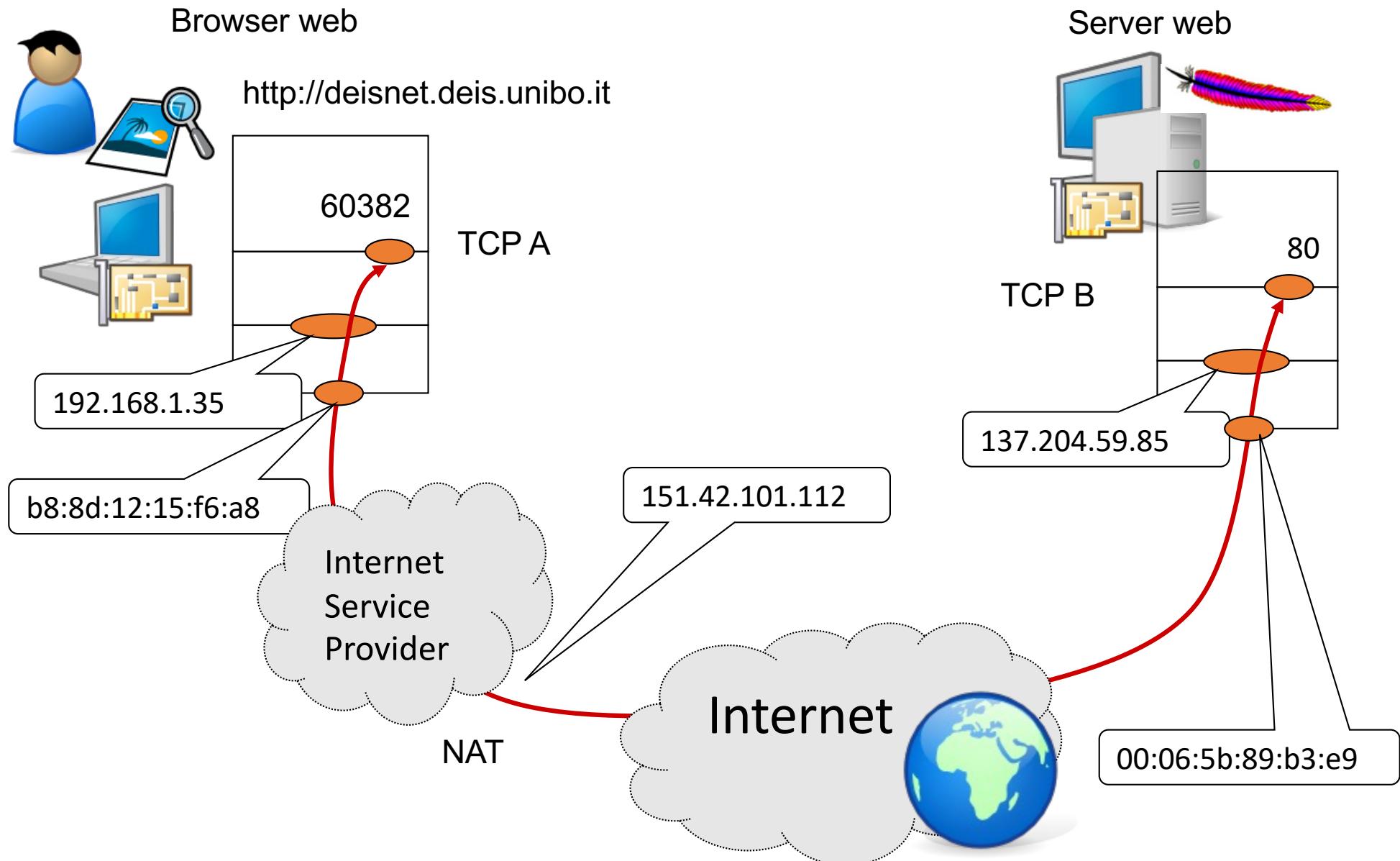


ACK duplicati

- Il TCP ricevente ritrasmette l'ACK per l'ultimo segmento ricevuto nella corretta sequenza generando un **ACK duplicato** (duplicate ACK)
 - Le implementazioni classiche di TCP ignorano gli ACK duplicati
 - Le implementazioni recenti prendono specifiche azioni se ricevono dei “duplicate ACK”



Topologia e indirizzi





Apertura: lato client – porte e flag

No	Time	Source	Destination	Src Port	Dst Port	Protocol	Length	Info
5036	69.032250	192.168.1.35	137.204.57.85	60382	80	TCP	78	60382-http [SYN] Seq=3874871719 Win=65535 Len=0 MSS=460
5038	69.185181	137.204.57.85	192.168.1.35		80	60382 TCP	74	http-60382 [SYN, ACK] Seq=847290582 Ack=3874871720 Win=57480
5039	69.185319	192.168.1.35	137.204.57.85	60382	80	TCP	66	60382-http [ACK] Seq=3874871720 Ack=847290583 Win=13480
5090	69.187162	192.168.1.35	137.204.57.85	60382	80	HTTP	763	GET / HTTP/1.1
5128	69.328866	137.204.57.85	192.168.1.35		80	60382 TCP	66	http-60382 [ACK] Seq=847290583 Ack=3874872417 Win=7188 Len=632
5136	69.344413	137.204.57.85	192.168.1.35		80	60382 TCP	1506	[TCP segment of a reassembled PDU]
5137	69.344939	137.204.57.85	192.168.1.35		80	60382 TCP	1506	[TCP segment of a reassembled PDU]
5138	69.345002	192.168.1.35	137.204.57.85	60382	80	TCP	66	60382-ht
5207	69.524222	137.204.57.85	192.168.1.35		80	60382 TCP	1506	[TCP seg
5208	69.524362	137.204.57.85	192.168.1.35		80	60382 HTTP	124	HTTP/1.1
5209	69.524458	192.168.1.35	137.204.57.85	60382	80	TCP	66	60382-ht
5476	70.329117	192.168.1.35	137.204.57.85	60382	80	HTTP	766	GET /images/english_flag.gif HTTP/1.1
5504	70.471930	137.204.57.85	192.168.1.35		80	60382 HTTP	1001	HTTP/1.1 200 OK (GIF89a)
5505	70.472011	192.168.1.35	137.204.57.85	60382	80	TCP	66	60382-http [ACK] Seq=3874873117 Ack=847295896 Win=130112
5525	70.886308	137.204.57.85	192.168.1.35		80	60382 HTTP	1001	[TCP Retransmission] HTTP/1.1 200 OK (GIF89a)
5526	70.886409	192.168.1.35	137.204.57.85	60382	80	TCP	66	[TCP Window Update] 60382-http [ACK] Seq=3874873117 Ack=847295896 Win=130112
5527	70.886411	137.204.57.85	192.168.1.35		80	60382 TCP	66	60382-ht

Three ways handshake

```
▶ Frame 5026: 78 bytes on wire (624 bits), 78 bytes captured (624 bits)
▶ Ethernet II, Src: Apple_15:f6:a8 (b8:8d:12:15:f6:a8), Dst: Netgear_c0:7f:be (00:18:4d:c0:7f:be)
▶ Internet Protocol Version 4, Src: 192.168.1.35 (192.168.1.35), Dst: 137.204.57.85 (137.204.57.85)
▽ Transmission Control Protocol, Src Port: 60382 (60382), Dst Port: http (80), Seq: 3874871719, Len: 6
```

Source Port: 60382 (60382)
Destination Port: http (80)
[Stream index: 150]
[TCP Segment Len: 0]
Sequence number: 3874871719
Acknowledgment number: 0
Header Length: 44 bytes

Porta sorgente = 60382

Porta destinazione = 80

Flag SYN impostato a 1



Apertura: lato client - ISN

No.	Time	Source	Destination	Src Port	Dst Port	Protocol	Length	Info
5026	69.032250	192.168.1.35	137.204.57.85	60382	80	TCP	78	60382>http [SYN] Seq=3874871719 Win=65535 Len=0 MSS= 460
5038	69.185181	137.204.57.85	192.168.1.35		80	60382 TCP	74	http->60382 [SYN, ACK] Seq=847290582 Ack=3874871720 Win=57
5039	69.185319	192.168.1.35	137.204.57.85	60382	80	TCP	66	60382>http [ACK] Seq=3874871720 Ack=847290583 Win=134480
5090	69.187102	192.168.1.35	137.204.57.85	60382	80	HTTP	705	GET / HTTP/1.1
5128	69.328866	137.204.57.85	192.168.1.35	80	60382	TCP	66	http->60382 [ACK] Seq=847290583 Ack=3874872417 Win=7188 Le
5136	69.344413	137.204.57.85	192.168.1.35	80	60382	TCP	1506	[TCP segment reassembled PDU]
5137	69.344939	137.204.57.85	192.168.1.35	80	60382	TCP	1506	[TCP segment reassembled PDU]
5138	69.345002	192.168.1.35	137.204.57.85	60382	80	TCP	66	60382>http [ACK] Seq=3874873117 Ack=847295896 Win=130112
5207	69.524222	137.204.57.85	192.168.1.35	80	60382	TCP	1506	[TCP segment reassembled PDU]
5208	69.524362	137.204.57.85	192.168.1.35	80	60382	HTTP	124	HTTP/1.1
5209	69.524458	192.168.1.35	137.204.57.85	60382	80	TCP	66	60382>http [ACK] Seq=3874873117 Ack=847295896 Win=130112
5476	70.329117	192.168.1.35	137.204.57.85	60382	80	HTTP	766	GET /images/english_flag.gif HTTP/1.1
5504	70.471930	137.204.57.85	192.168.1.35	80	60382	HTTP	1001	HTTP/1.1 200 OK (GIF89a)
5505	70.472011	192.168.1.35	137.204.57.85	60382	80	TCP	66	60382>http [ACK] Seq=3874873117 Ack=847295896 Win=130112
5525	70.886308	137.204.57.85	192.168.1.35	80	60382	HTTP	1001	[TCP Retransmission] HTTP/1.1 200 OK (GIF89a)
5526	70.886409	192.168.1.35	137.204.57.85	60382	80	TCP	66	[TCP Window Update] 60382>http [ACK] Seq=3874873117 Ack=847295896 Win=130112

```

Frame 5026: 78 bytes on wire (624 bits), 78 bytes captured (624 bits)
Ethernet II, Src: Apple_15:f6:a8 (b8:8d:12:15:f6:a8), Dst: Netgear_c0:7f:be (00:18:4d:c0:7f:be)
Internet Protocol Version 4, Src: 192.168.1.35 (192.168.1.35), Dst: 137.204.57.85 (137.204.57.85)
Transmission Control Protocol, Src Port: 60382 (60382), Dst Port: http (80), Seq: 3874871719, Len: 0
    Source Port: 60382 (60382)
    Destination Port: http (80)
    [Stream index: 150]
    [TCP Segment Len: 0]
    Sequence number: 3874871719
    Acknowledgment number: 0
    Header Length: 44 bytes
    ...
    Window size value: 65535
    Calculated window size: 65535
0000  00 18 4d c0 7f be b8 8d  12 15 f6 a8 08 00
0010  00 40 41 be 40 00 40 06  74 0d c0 a8 01 23 80
0020  39 55 eb de 00 50 e6 f5  d9 a7 00 00 00 00 b0 00
0030  ff ff b6 c2 00 00 02 04  05 b4 01 03 03 05 01 00
0040  08 0a 11 ce 3c b3 00 00  00 00 04 02 00 00

```

Three ways handshake

TCP A invia il SYN e propone il suo ISN

SYN è il primo messaggio della connessione, non si sa nulla dell'altro e quindi ACK-N = 0



Invio richiesta HTTP

No.	Time	Source	Destination	Src Port	Dst Port	Protocol	Length	Info
5026	69.032250	192.168.1.35	137.204.57.85	60382	80	TCP	78	60382->http [SYN] Seq=3874871719 Win=65535 Len=0 MSS=1460
5088	69.185181	137.204.57.85	192.168.1.35		80	60382 TCP	74	http->60382 [SYN, ACK] Seq=847290582 Ack=3874871720 Win=57
5089	69.185319	192.168.1.35	137.204.57.85	60382	80	TCP	66	60382->http [ACK] Seq=3874871720 Ack=847290583 Win=132480
5090	69.187162	192.168.1.35	137.204.57.85	60382	80	HTTP	763	GET / HTTP/1.1
5128	69.328866	137.204.57.85	192.168.1.35		80	60382 TCP	66	http->60382 [ACK] Seq=847290583 Ack=3874872417 Win=7188 Len=129632
5136	69.344413	137.204.57.85	192.168.1.35		80	60382 TCP	1506	[TCP segment of a reassembled PDU]
5137	69.344939	137.204.57.85	192.168.1.35		80	60382 TCP	1506	[TCP segment of a reassembled PDU]
5138	69.345002	192.168.1.35	137.204.57.85	60382		TCP		
5207	69.524222	137.204.57.85	192.168.1.35		80	60382 TCP		
5208	69.524362	137.204.57.85	192.168.1.35		80	60382 HTTP		
5209	69.524458	192.168.1.35	137.204.57.85	60382	80	TCP		
5476	70.329117	192.168.1.35	137.204.57.85	60382	80	HTTP		
5504	70.471930	137.204.57.85	192.168.1.35		80	60382 HTTP		
5505	70.472011	192.168.1.35	137.204.57.85	60382	80	TCP		
5525	70.886308	137.204.57.85	192.168.1.35		80	60382 HTTP	1001	[TCP Retransmission] HTTP/1.1 200 OK (GIF89a)
5526	70.886409	192.168.1.35	137.204.57.85	60382	80	TCP	66	[TCP Window Update] 60382->http [ACK] Seq=3874873117 Ack=8
5527	70.886414	137.204.57.85	192.168.1.35		80	60382 TCP	66	[TCP Window Update] 60382->http [ACK] Seq=3874873117 Ack=8

```

> Frame 5090: 763 bytes on wire (6104 bits), 763 bytes captured (6104 bits)
> Ethernet II, Src: Apple_15:f6:a8 (b8:8d:12:15:f6:a8), Dst: Netgear_c0:7f:be (00:18:4d:c0:7f:be)
> Internet Protocol Version 4, Src: 192.168.1.35 (192.168.1.35), Dst: 137.204.57.85 (137.204.57.85)
> Transmission Control Protocol, Src Port: 60382 (60382), Dst Port: http (80), Seq: 3874871720, Ack: 847290583, Len: 697
  Hypertext Transfer Protocol
    GET / HTTP/1.1\r\n
      Host: deisnet.deis.unibo.it\r\n
      Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
      Connection: keep-alive\r\n
    [truncated]Cookie: __utma=83985245.1349407932.1432415213.1432484513.3
      User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_3) AppleWebKit/600.5.17
      Accept-Language: it-it\r\n
      DNT: 1\r\n

```

0020	39 55 eb de 00 50 e6 f5	d9 a8 32 80 a0 d7 80 18	9U...P... 2...
0030	10 2c 44 ff 00 00 01 01	08 0a 11 ce 3d 46 43 62	,D..... =FO
0040	0b 1c 47 45 54 20 2f 20	48 54 54 50 2f 31 2e 31	.GET / HTTP/1
0050	0d 0a 48 6f 73 74 3a 20	64 65 69 73 6e 65 74 2e	.Host: deisnet
0060	64 65 69 73 2e 75 6e 69	62 6f 2e 69 74 0d 0a 41	deis.uni bo.it.
0070	63 63 65 70 74 3a 20 74	65 78 74 2f 68 74 6d 6c	ccept: t ext/ht
0080	2c 61 70 70 6c 69 63 61	74 69 6f 6e 2f 78 68 74	, applica tion/xl
0090	6d 6c 2b 78 6d 6c 2c 61	70 70 6c 69 63 61 74 69	ml+xml, a pplicati

Terminata l'apertura della connessione TCP A chiede l'invio della pagina web

Il segmento TCP contiene 697 byte di dati (payload) che sono le informazioni di livello applicativo

Il segmento TCP contiene un messaggio HTTP di tipo GET che richiede l'invio della pagina di default del sito web



Invio richiesta HTTP

Numero di sequenza iniziale (ISN) concordato con il three ways handshake è
3874071720

Il segmento contiene 697 byte numerati sequenzialmente da 3874071720 a 3874072416 (in totale 697 numeri)
Il primo byte del prossimo segmento deve essere quello successivo: 3874072417

L'analizzatore di protocollo calcola il prossimo numero di sequenza e lo indica (non è incluso nell'intestazione TCP)

No.	Time	Source	Destination
5026	69.032250	192.168.1.35	137.204.57.85
5088	69.185181	137.204.57.85	192.168.1.35
5089	69.185319	192.168.1.35	137.204.57.85
5090	69.187162	192.168.1.35	137.204.57.85
5128	69.328866	137.204.57.85	192.168.1.35
5136	69.344413	137.204.57.85	192.168.1.35
5137	69.344939	137.204.57.85	192.168.1.35
5138	69.345002	192.168.1.35	137.204.57.85
5207	69.524222	137.204.57.85	192.168.1.35
5208	69.524362	137.204.57.85	192.168.1.35
5209	69.524458	192.168.1.35	137.204.57.85
5476	70.329117	192.168.1.35	137.204.57.85
5504	70.471930	137.204.57.85	192.168.1.35
5505	70.472011	192.168.1.35	137.204.57.85
5525	70.886308	137.204.57.85	192.168.1.35
5526	70.886409	192.168.1.35	137.204.57.85

```
Frame 5090: 763 bytes on wire (6104 bits), 763 bytes captured (6104 bits) on interface eth0
Ethernet II, Src: Apple_15:f6:a8 (b8:8d:12:15:f6:a8), Dst: deisnet (00:0c:29:82:00:00)
Internet Protocol Version 4, Src: 192.168.1.35, Dst: 137.204.57.85
Transmission Control Protocol, Src Port: 60382 (60382), Dst Port: http (80)
Source Port: 60382 (60382)
Destination Port: http (80)
[Stream index: 150]
[TCP Segment Len: 697]
Sequence number: 3874871720
[Next sequence number: 3874872417]
Acknowledgment number: 847290583
Header Length: 32 bytes
.... 0000 0001 1000 = Flags: 0x018 (PSH, ACK)
Window size value: 4140
```

0020	39 55 eb de 00 50 e6 f5 d9 a8 32 80 a0 d7 80 18 9U...P...
0030	10 2c 44 ff 00 00 01 01 08 0a 11 ce 3d 46 43 62 ,D...
0040	0b 1c 47 45 54 20 2f 20 48 54 54 50 2f 31 2e 31 ..GET / HTTP/1.1
0050	0d 0a 48 6f 73 74 3a 20 64 65 69 73 6e 65 74 2e ..Host: deisnet.
0060	64 65 69 73 2e 75 6e 69 62 6f 2e 69 74 0d 0a 41 deis.uni bo.it..A
0070	63 63 65 70 74 3a 20 74 65 78 74 2f 68 74 6d 6c ccept: text/html
0080	2c 61 70 70 6c 69 63 61 74 69 6f 6e 2f 78 68 74 ,application/xhtml+xml, application/xml, application/atom+xml
0090	6d 6c 20 78 6d 6c 2c 61 70 70 6c 69 63 61 74 69 ml+xml, application/atom+xml



Il primo ACK



La pagina web

No.	Time	Source	Destination	Src Port	Dst Port	Protocol	Length	Info
5026	69.032250	192.168.1.35	137.204.57.85	60382	80	TCP	78	60382->http [SYN] Seq=3874871719 Win=65535 Len=0 MSS=1460
5088	69.185181	137.204.57.85	192.168.1.35		80	60382	74	TCP->60382 [SYN, ACK] Seq=847290582 Ack=3874871720 Win=57
5089	69.185319	192.168.1.35	137.204.57.85	60382	80	TCP	66	60382->http [ACK] Seq=3874871720 Ack=847290583 Win=132480
5090	69.187162	192.168.1.35	137.204.57.85	60382	80	HTTP	763	GET / HTTP/1.1
5128	69.328866	137.204.57.85	192.168.1.35		80	60382	66	HTTP->60382 [ACK] Seq=847290583 Ack=3874872417 Win=7188 Le
5136	69.344413	137.204.57.85	192.168.1.35		80	60382	1506	[TCP segment of a reassembled PDU]
5137	69.344939	137.204.57.85	192.168.1.35		80	60382	1506	[TCP segment of a reassembled PDU]
5138	69.345002	192.168.1.35	137.204.57.85	60382	80	TCP	66	60382->http [ACK] Seq=3874872417 Ack=847293463 Win=129632
5207	69.524222	137.204.57.85	192.168.1.35		80	60382	1506	[TCP segment of a reassembled PDU]
5208	69.524362	137.204.57.85	192.168.1.35		80	60382	124	HTTP/1.1 200 OK (text/html)
5209	69.524459	192.168.1.35	137.204.57.85				66	60382->http [ACK] Seq=3874872417 Ack=847294961 Win=129568
5476	70.3						766	GET /images/english_flag.gif HTTP/1.1
5504	70.4						1001	HTTP/1.1 200 OK (GIF89a)
5505	70.4						66	60382->http [ACK] Seq=3874873117 Ack=847295896 Win=130112
5525	70.8						1001	[TCP Retransmission] HTTP/1.1 200 OK (GIF89a)

L'ultima trama MAC contiene quanto rimasto per completare la pagina web, complessivamente composta da $4378 = 1440 * 3 + 58$ byte

```
▷ Frame 5136: 1506 bytes on wire (12048 bits), 1506 bytes captured (12048 bits)
▷ Ethernet II, Src: Netgear_c0:7f:be (00:18:4d:c0:7f:be), Dst: Apple_17:b2:4c (08:00:27:b2:4c:17)
▷ Internet Protocol Version 4, Src: 137.204.57.85 (137.204.57.85), Dst: 137.204.57.17 (137.204.57.17)
▽ Transmission Control Protocol, Src Port: http (80), Dst Port: 60382
    Source Port: http (80)
    Destination Port: 60382 (60382)
    [Stream index: 150]
    [TCP Segment Len: 1440]
    Sequence number: 847290583
    [Next sequence number: 847292023]
    Acknowledgment number: 3874872417
    Header Length: 32 bytes
```

0000	b8	8d	12	15	f6	a8	00	18	4d	c0	7f	be	08	00	45	00
0010	05	d4	ab	98	40	00	32	06	12	9f	89	cc	39	55	c0	a8
0020	01	23	00	50	eb	de	32	80	a0	d7	e6	f5	dc	61	80	10
0030	07	05	b6	a3	00	00	01	01	08	0a	43	62	0b	44	11	ce
0040	3d	46	48	54	54	50	2f	31	2e	31	20	32	30	30	20	4f
0050	4b	0d	0a	44	61	74	65	3a	20	53	75	6e	2c	20	32	34
0060	20	4d	61	79	20	32	30	31	35	20	31	36	3a	32	33	3a
0070	32	33	20	47	4d	54	0d	0a	53	65	72	76	65	72	3a	20

Il client inizia a ricevere segmenti con dati dal server.
Sono 4 segmenti di lunghezza 1506, 1506, 1506 e 124 byte

La pagina web inviata dal server è stata suddivisa in 4 blocchi per riempire 3 datagrammi IP e 3 trame MAC al massimo possibile

Di questi 1506 byte 1440 sono relativi al protocollo HTTP ossia dati dell'applicazione server



Ritrasmissione: lato client

No.	Time	Source	Destination	Src Port	Dst Port	Protocol	Length	Info
5026	69.032250	192.168.1.35	137.204.57.85	60382	80	TCP	78	60382->http [SYN] Seq=3874871719 Win=65535 Len=0 MSS=1460
5088	69				74	http->60382 [SYN, ACK] Seq=847290582 Ack=3874871720 Win=57		
5089	69				66	60382->http [ACK] Seq=3874871720 Ack=847290583 Win=132480		
5090	69				763	GET / HTTP/1.1		
5128	69				66	http->60382 [ACK] Seq=847290583 Ack=3874872417 Win=7188 Le		
5136	69				1506	[TCP segment of a reassembled PDU]		
5137	69				1506	[TCP segment of a reassembled PDU]		
5138	69.345002	192.168.1.35	137.204.57.85		80	TCP	66	60382->http [ACK] Seq=3874872417 Ack=847293463 Win=129632
5207	69.524222	137.204.57.85	192.168.1.35		80	TCP	1506	[TCP segment of a reassembled PDU]
5208	69.524362	137.204.57.85	192.168.1.35		80	HTTP	124	HTTP/1.1 200 OK (text/html)
5209	69.524458	192.168.1.35	137.204.57.85	60382			66	60382->http [ACK] Seq=3874872417 Ack=847294961 Win=129568
5476	70.329117	192.168.1.35	137.204.57.85	60382	80	HTTP	766	GET /images/english_flag.gif HTTP/1.1
5504	70.471930	137.204.57.85	192.168.1.35	80	60382	HTTP	1001	HTTP/1.1 200 OK (GIF89a)
5505	70.472011	192.168.1.35	137.204.57.85	60382	80	TCP	66	60382->http [ACK] Seq=3874873117 Ack=847295896 Win=130112
5525	70.886308	137.204.57.85	192.168.1.35	80	60382	HTTP	1001	[TCP Retransmission] HTTP/1.1 200 OK (GIF89a)
5526	70.886409	192.168.1.35	137.204.57.85	60382	80	TCP	66	[TCP Window Update] 60382->http [ACK] Seq=3874873117 Ack=847295896 Win=130112
5605	70.886409	192.168.1.35	137.204.57.85	60382	80	TCP	66	[TCP Window Update] 60382->http [ACK] Seq=3874873117 Ack=847295896 Win=130112

Questo messaggio viene ritrasmesso dal server,
nonostante che il client abbia correttamente inviato la
conferma della corretta ricezione



Lato server

No.	Time	Source	Destination	Src Port	Dst Port	Protocol	Length	Info
1883	54.742790	151.42.79.161	137.204.57.85	60382	80	TCP	78	60382>http [SYN] Seq=3874871719 Win=65535 Len=0 MSS=1452
1884	54.742816	137.204.57.85	151.42.79.161		80	60382	TCP	74 http->60382 [SYN, ACK] Seq=847290582 Ack=3874871720 Win=57
1892	54.877566	151.42.79.161	137.204.57.85	60382	80	TCP	66	60382>http [ACK] Seq=3874871720 Ack=847290583 Win=132480
1893	54.886652	151.42.79.161	137.204.57.85	60382	80	HTTP	763	GET / HTTP/1.1
1894	54.886673	137.204.57.85	151.42.79.161		80	60382	TCP	66 http->60382 [ACK] Seq=847290583 Ack=3874872417 Win=7188 Le
1897	54.899808	137.204.57.85	151.42.79.161		80	60382	TCP	1506 [TCP segment of a reassembled PDU]
1898	54.899843	137.204.57.85	151.42.79.161		80	60382	TCP	1506 [TCP segment of a reassembled PDU]
1903	55.080073	151.42.79.161	137.204.57.85	60382	80	TCP	66	60382>http [ACK] Seq=3874872417 Ack=847293463 Win=129632
1904	55.080095	137.204.57.85	151.42.79.161		80	60382	TCP	1506 [TCP segment of a reassembled PDU]
1905	55.080103	137.204.57.85	151.42.79.161		80	60382	HTTP	124 HTTP/1.1 200 OK (text/html)
1918	55.217302	151.42.79.161	137.204.57.85	60382	80	TCP	66	60382>http [ACK] Seq=3874872417 Ack=847294961 Win=129568
1980	56.028035	151.42.79.161	137.204.57.85	60382	80	HTTP	766	GET /images/english_flag.gif HTTP/1.1
1981	56.028436	137.204.57.85	151.42.79.161		80	60382	HTTP	1001 HTTP/1.1 200 OK (GIF89a)
2012	56.442888	137.204.57.85	151.42.79.161		80	60382	HTTP	1001 [TCP Retransmission] HTTP/1.1 200 OK (GIF89a)
2017	56.580028	151.42.79.161	137.204.57.85	60382	80	TCP	66	60382>http [ACK] Seq=3874873117 Ack=847295896 Win=131072
2423	71.032219	137.204.57.85	151.42.79.161		80	60382	TCP	66 http->60382 [FIN, ACK] Seq=847295896 Ack=3874873117 Win=85
2423	71.167242	151.42.79.161	137.204.57.85	60382	80	TCP	66	60382>http [ACK] Seq=3874873117 Ack=847295896 Win=131072

Frame 1883: 78 bytes on wire (624 bits), 78 bytes captured (624 bits)
Ethernet II, Src: Cisco_8f:21:47 (00:16:47:8f:21:47), Dst: Intel_5e:3f:1c (00:0e:0c:5e:3f:1c)
Internet Protocol Version 4, Src: 151.42.79.161 (151.42.79.161), Dst: 137.204.57.85 (137.204.57.85)
Transmission Control Protocol, Src Port: 60382 (60382), Dst Port: http (80), Seq: 3874871719, Len: 0

Source Port: 60382 (60382)
Destination Port: http (80)
[Stream index: 44]
[TCP Segment Len: 0]
Sequence number: 3874871719
Acknowledgment number: 0
Header Length: 44 bytes

.... 0000 0000 0010 = Flags: 0x002 (SYN)

0000	00	0e	0c	5e	3f	1c	00	16	47	8f	21	47	08	00	45	00^?... G.!G..E.
0010	00	40	41	be	40	00	32	06	5d	0d	97	2a	4f	a1	89	cc	.@A.@.2.].*0...
0020	39	55	eb	de	00	50	e6	f5	d9	a7	00	00	00	00	b0	02	9U...P..
0030	ff	ff	91	ca	00	00	02	04	05	ac	01	03	03	05	01	01<....
0040	08	0a	11	ce	3c	b3	00	00	00	00	04	02	00	00	00	00<....



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Controllo di flusso in TCP



Controllo di flusso

- Le velocità di trasmettitore e ricevitore possono essere molto diverse
 - Il trasmettitore non deve saturare il ricevitore
- Analogamente al caso dello strato 2
 - Si utilizza un meccanismo a finestra scorrevole
- **Quale deve essere la dimensione della finestra?**
 - Deve essere dimensionata in modo congruente con le memoria di trasmissione e ricezione
 - Il trasmettitore conosce le dimensioni della propria memoria ma non conosce quelle delle memoria di ricezione
- Il ricevitore deve comunicare al trasmettitore le dimensioni della sua memoria di ricezione
 - Nell' intestazione del pacchetto TCP è contenuto il campo advertised window (AW)



Finestra di trasmissione e di ricezione

- Ricevitore e trasmettitore possono memorizzare segmenti
 - W_T = finestra di trasmissione
 - Insieme di segmenti inviabili da trasmettitore senza ricevere conferme di ricezione
 - W_R = finestra di ricezione
 - Insieme di segmenti memorizzabili fuori sequenza al ricevitore
- M = spazio di numerazione (in TCP 2^{32})
- Se $W_T = W_R$ allora deve essere $W_T + W_R \leq M$
 - $W_T \leq 2^{31}$
 - $W_R \leq 2^{31}$



Unità di misura di W

- TCP adotta la numerazione sequenziale dei dati trasmessi per byte
- W può essere misurata
 - In byte (w)
 - In numero di segmenti (W)
 - In questo caso si deve indicare quale lunghezza si assume per i segmenti
- Normalmente W viene misurata in segmenti di dimensione massima (full sized segments)

$$W \text{ MSS} = w$$



Dimensionamento di W

- W (o w) viene messa a punto dinamicamente sulla base di informazioni
 - Provenienti dal ricevente (advertised window o AW)
 - Comunicate in modo esplicito al trasmettitore
 - Correlate allo stato di congestione della rete (congestion window o CW)
 - Desunte dal trasmettitore in base al comportamento del canale
- AW e CW sono funzione del tempo
- In un generico istante di tempo la connessione imposta

$$W = \min[AW, CW]$$

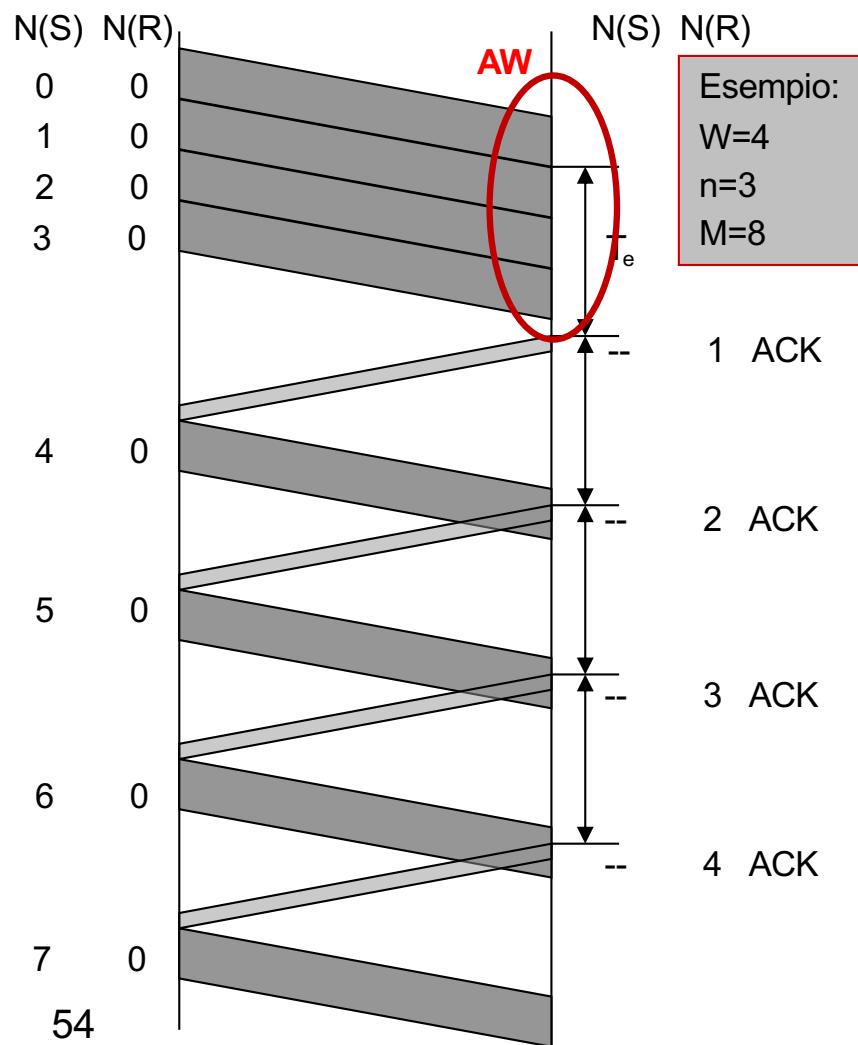


Motivazioni

- Il protocollo a finestra ha l'obiettivo di garantire il controllo di flusso
 - Impedire che una sorgente congestioni il sistema di comunicazione impedendo il corretto trasferimento dei dati
- Nello strato 4 un'errato dimensionamento di W può congestionare:
 - Il ricevitore
 - Per prevenire questo viene utilizzata AW
 - I nodi intermedi della rete
 - Per prevenire questo viene utilizzata CW
- Impostare la finestra al minimo fra AW e CW rappresenta la soluzione più conservativa

Controllo di flusso: ruolo di AW

- Riprendiamo l'esempio di trasmettitore veloce e ricevitore lento



- Con AW il ricevitore indica al trasmettitore la dimensione del suo buffer
- In questo modo si è certi che possa ricevere l'intera prima finestra di dati



Attuazione del controllo con AW

- Come un ricevitore lento blocca un trasmettitore più veloce
 - Il buffer di ricezione si riempie
 - $AW = 0$
 - $W = 0$
 - Il trasmettitore blocca la trasmissione
- Ripresa della trasmissione
 - Il processo ricevente legge dal buffer
 - $AW > 0$
 - Arrivano gli ACK
 - Si libera il buffer di trasmissione
 - Viene ricevuto $AW > 0$
 - Il processo trasmittente ricomincia a trasmettere



Deadlock

Trasmittente

- Invia messaggi
- Riceve un messaggio con AW=0
- Sospende l' invio dei dati

Ricevente

- Il buffer di ricezione si riempie
- Invia un messaggio con AW=0
- Non ha altri messaggi da trasmettere

A questo punto il protocollo è in deadlock

- Il trasmittente non può inviare dati poiché AW=0
- Il ricevente non ha dati da inviare quindi non ha modo di comunicare AW>0

TCP prevede che sia sempre possibile inviare un segmento di 1 byte anche se AW=0



Persist timer e window probe

- Il trasmettitore riceve ACK fino al byte X ma contenente AW=0
- Fa partire il “persist timer”
 - $PT = 1,5$ sec per un normale collegamento LAN
 - Quando PT scade si invia un segmento di 1 byte
 - seqN=X+1
 - Il ricevitore deve rispondere
 - Invia ACK con ackN=X+2 e AW>0
 - La trasmissione riprende
 - Invia ACK con ackN=X+1 e AW=0
 - Non ha spazio nel buffer di ricezione perciò non può ricevere il byte X+1
 - $PT = 2PT$ e si ricomincia ad attendere
 - Il massimo valore di PT viene fissato a 60 sec



Problemi

- Questo meccanismo può dare luogo a inefficienze in casi particolari
 - Trasmissione “lenta”
 - Ad esempio l’ applicazione trasmette un carattere per volta
 - Ricezione “lenta”
 - Ad esempio l’ applicazione è lenta ad accettare i dati, legge un byte alla volta e comunica una dimensione di finestra molto piccola



Ricevitore lento: Silly window syndrom

- In caso di applicazione ricevente lenta
 - Il buffer di ricezione si riempie $\Rightarrow AW=0$
 - L' applicazione legge un byte e trasmette $AW=1$
 - Il trasmettitore manda un segmento di un byte
 - Il buffer di ricezione si riempie $\Rightarrow AW=0$
- Viene trasmesso un byte alla volta
 - Qualunque sia la velocità della rete il throughput risulta dell' ordine di un byte per RTT
- Soluzione:
 - Il ricevitore non può aumentare AW a meno che
 - Il nuovo valore di AW sia almeno pari a MSS
 - Il nuovo valore di AW sia almeno pari a metà del buffer di ricezione

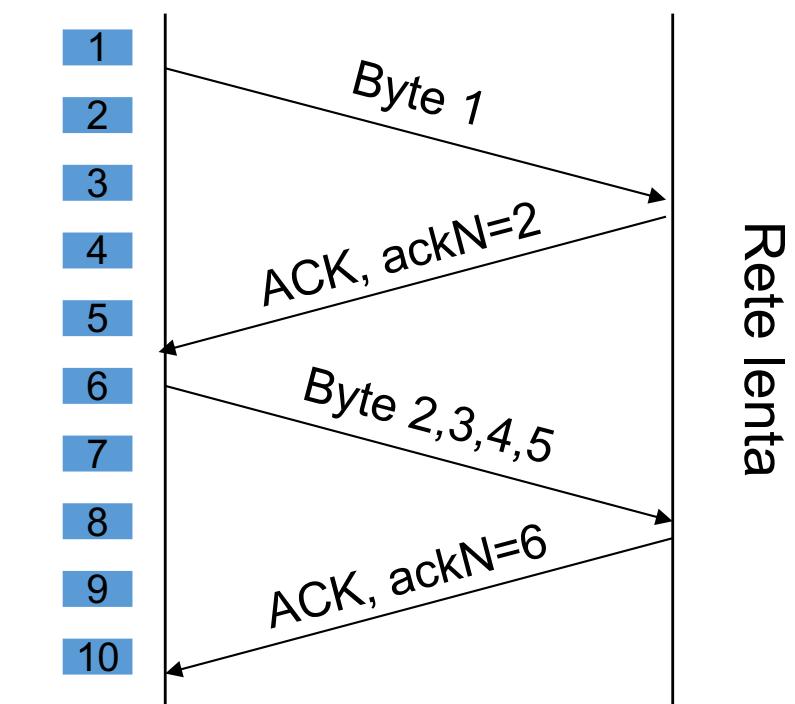
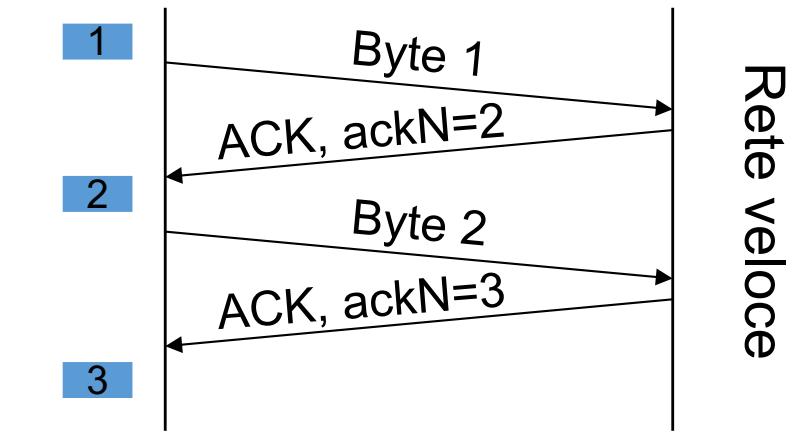


Tramettitore lento: Algoritmo di Nagle

- Applicazione trasmittente lenta
 - Passa a TCP un carattere per volta (Telnet, Rlogin)
 - Vengono trasmessi dei “tinygram”, ossia segmenti di un solo byte
 - Ciascun byte richiede almeno 40 byte di header più 40 byte di ACK
 - L’ overhead per byte risulta essere molto elevato (efficienza 1/81)
- Si deve aumentare la dimensione del messaggio
- Soluzione (algoritmo di Nagle)
 - Il trasmettitore trasmette un nuovo segmento solo se è vera una delle seguenti condizioni
 - Il segmento è di dimensioni pari a MSS
 - Il segmento è di dimensioni almeno pari a metà del valore di AW
 - Non vi sono ACK pendenti ed è possibile trasmettere tutto ciò che è in attesa nel buffer di trasmissione

Effetto dell'algoritmo di Nagle

- Si può avere un solo segmento pendente per il quale non si è ricevuto ACK
 - Più veloci arrivano gli ACK più velocemente si trasmette
- Ethernet
 - RTT tipico dell' ordine di 10 ms
 - Oltre 60 caratteri al sec.
 - Un carattere per tinygram
- Rete geografica
 - RTT dell' ordine dei sec.
 - Numerosi caratteri per tinygram





Disabilitare l'algoritmo di Nagle

- L' algoritmo di Nagle tende a ritardare i dati nel buffer di trasmissione
 - Per alcune applicazioni questo potrebbe non essere accettabile
 - X-windows: i movimenti del mouse devono essere trasmessi in tempo reale
- È possibile disabilitare l'algoritmo



Controllo di congestione

- W è limitato superiormente da AW o da CW
- Come viene determinata CW ?
- **TCP cerca di adattare la dimensione della finestra alle condizioni di congestione della rete**
- Idea base:
 - se si verifica congestione in rete si rallenta la trasmissione
 - Quando si verifica una perdita si riduce W
 - Quando gli ACK arrivano correttamente W viene aumentata

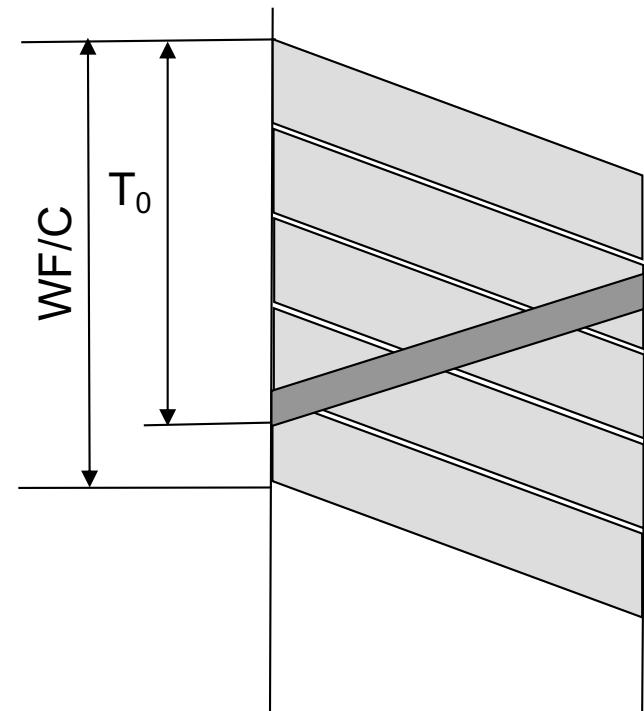


Efficienza

- Abbiamo già visto che l'efficienza ottima in assenza di errori richiede

$$WF \geq C T_0$$

- Nella letteratura sul TCP si fa solitamente riferimento al cosiddetto **prodotto banda-ritardo**
- Di fatto il prodotto banda-ritardo è una stima veloce di $C T_0$
- Questo è possibile solo se AW non pone un vincolo più stringente



Esempio

- Caso di studio:

- TCP A invia dati a TCP B
- Capacità del canale B = 10 Mbit/s
- Ritardo di propagazione T = 10 ms
- Round trip time RTT = 20 ms
- Tempo di elaborazione trascurabile
- MSS = 1000 byte = 8000 bit

- Con riferimento alla slide precedente

$$T_0 = \text{MSS}/B + \text{RTT} = 20.8 \text{ ms}$$

$$\text{MSS}/B = 0.8 \text{ ms}$$

$W = 26$ segmenti di dimensione MSS

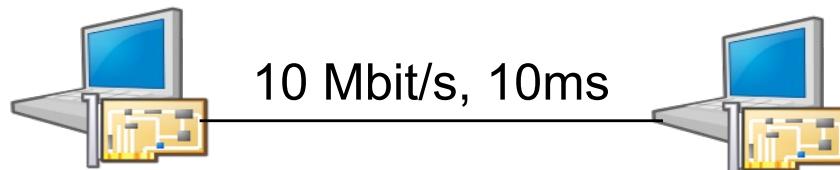
- Prodotto banda ritardo

$$B \cdot RTT = 20 \cdot 10^{-3} \cdot 10 \cdot 10^6 = 200 \cdot 10^3 \text{ bit}$$

- Pertanto

$$B \cdot RTT / MSS = 25 \text{ segmenti}$$

$$Wid \geq B \cdot RTT / MSS$$





CW ideale

- Avendo a disposizione una banda B (byte/sec)
 - Il massimo throughput si ottiene quando il protocollo a finestra non limita la velocità di scambio dei dati
- In questo caso si utilizza al 100% la capacità disponibile nella tratta trasmettitore/ricevitore
 - Se $w < w_{id}$: si spreca banda
 - Se $w > w_{id}$: è necessario accodare nei router intermedi
 - cresce il ritardo e potenzialmente anche la perdita
- Il massimo throughput (byte/sec) vale circa:

$$S = w/RTT$$

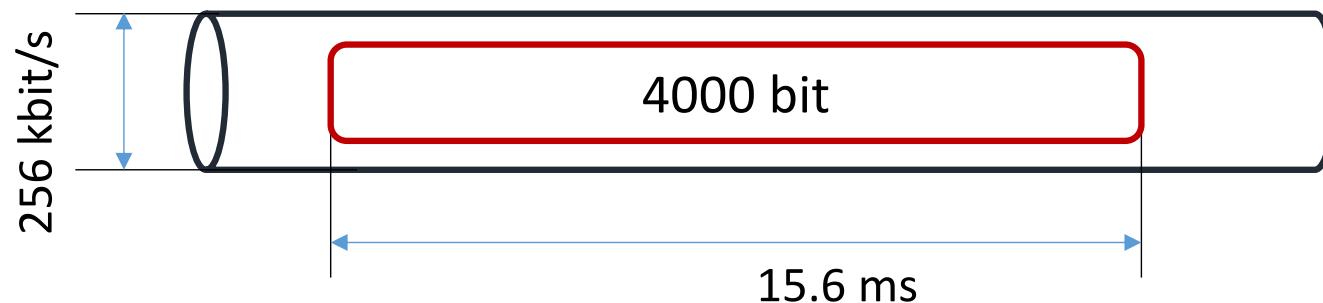


Nel mondo reale : la rete!

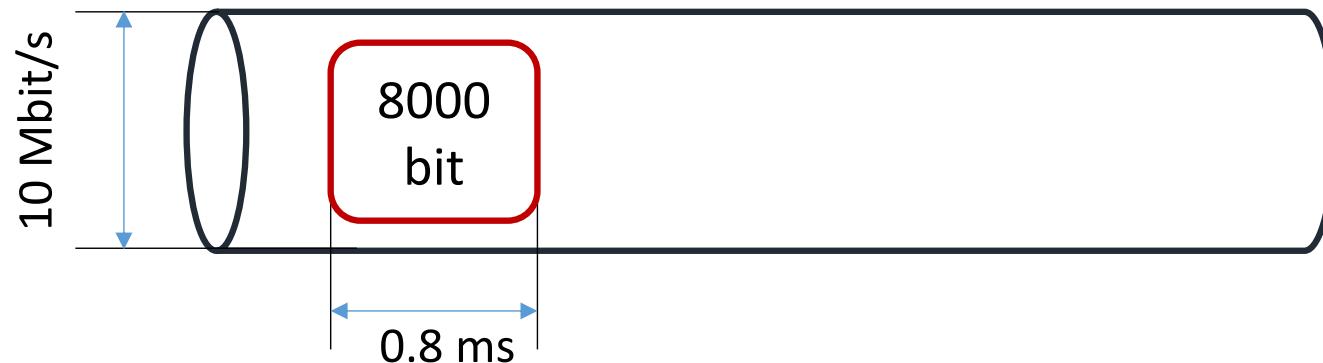
- Al momento dell'instaurazione della connessione:
 - La banda disponibile B è incognita
 - Il percorso che verrà seguito da dati e quindi il RTT è incognito
- Durante il trasferimento dei dati
 - Il canale di comunicazione è condiviso con altri utenti (rete a pacchetto) quindi la banda realmente disponibile varia continuamente
 - La rete può modificare il percorso dei dati pertanto il RTT può modificarsi
 - Lungo il percorso i pacchetti vengono accodati nei nodi di rete quindi il ritardo di propagazione può modificarsi anche segmento per segmento
- In conclusione ... il TCP dovrebbe poter «indovinare» i valori di banda e ritardo per poter dimensionare correttamente CW

La rappresentazione di Jackobson

- Diagramma bidimensionale
 - Orizzontalmente si indica il tempo
 - Verticalmente si indica la banda disponibile
 - Esempio: MSS=4000bit, B= 256Kbit/s

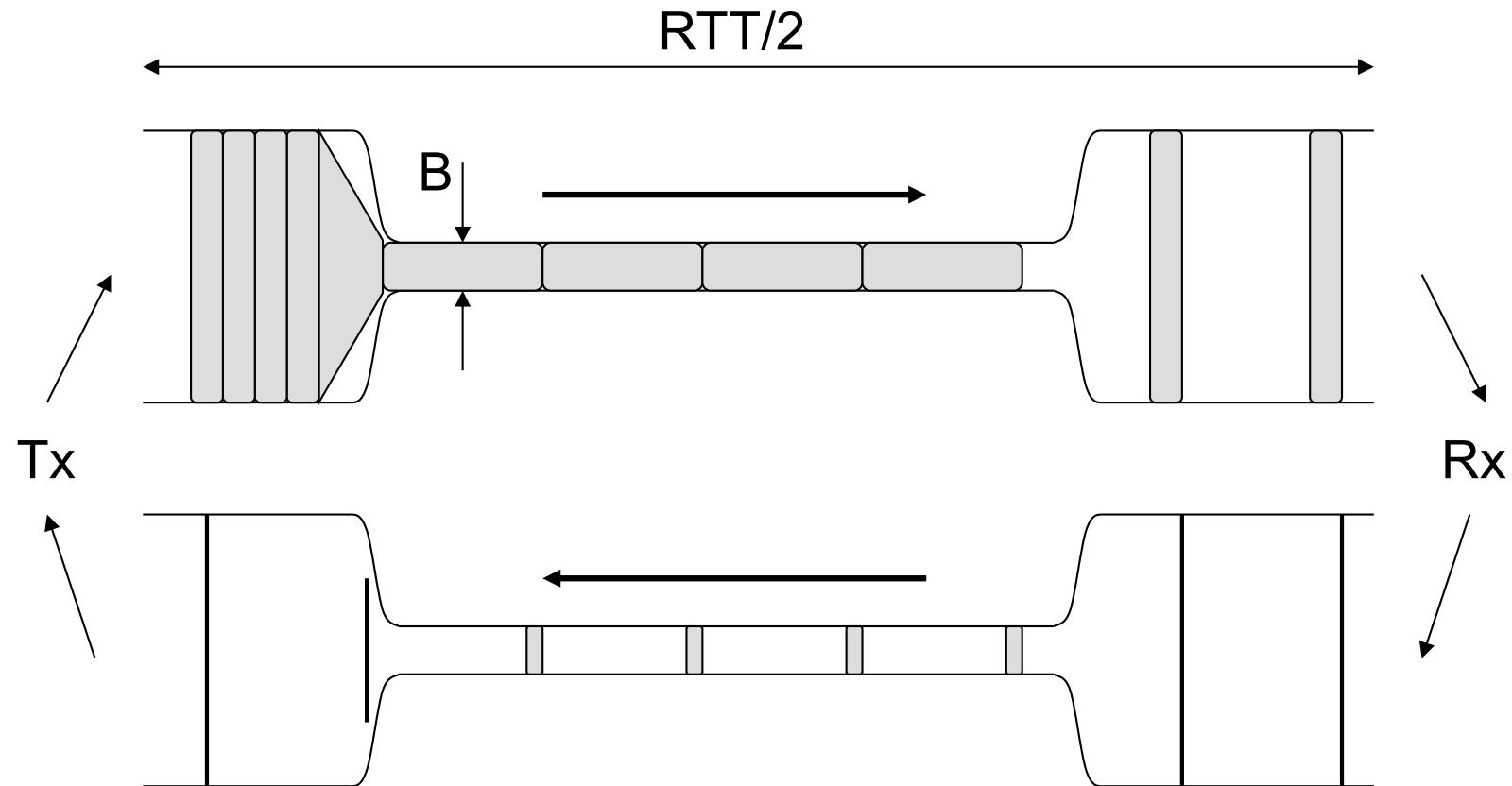


- Esempio: MSS = 8000 bit, B = 10 Mbit/s



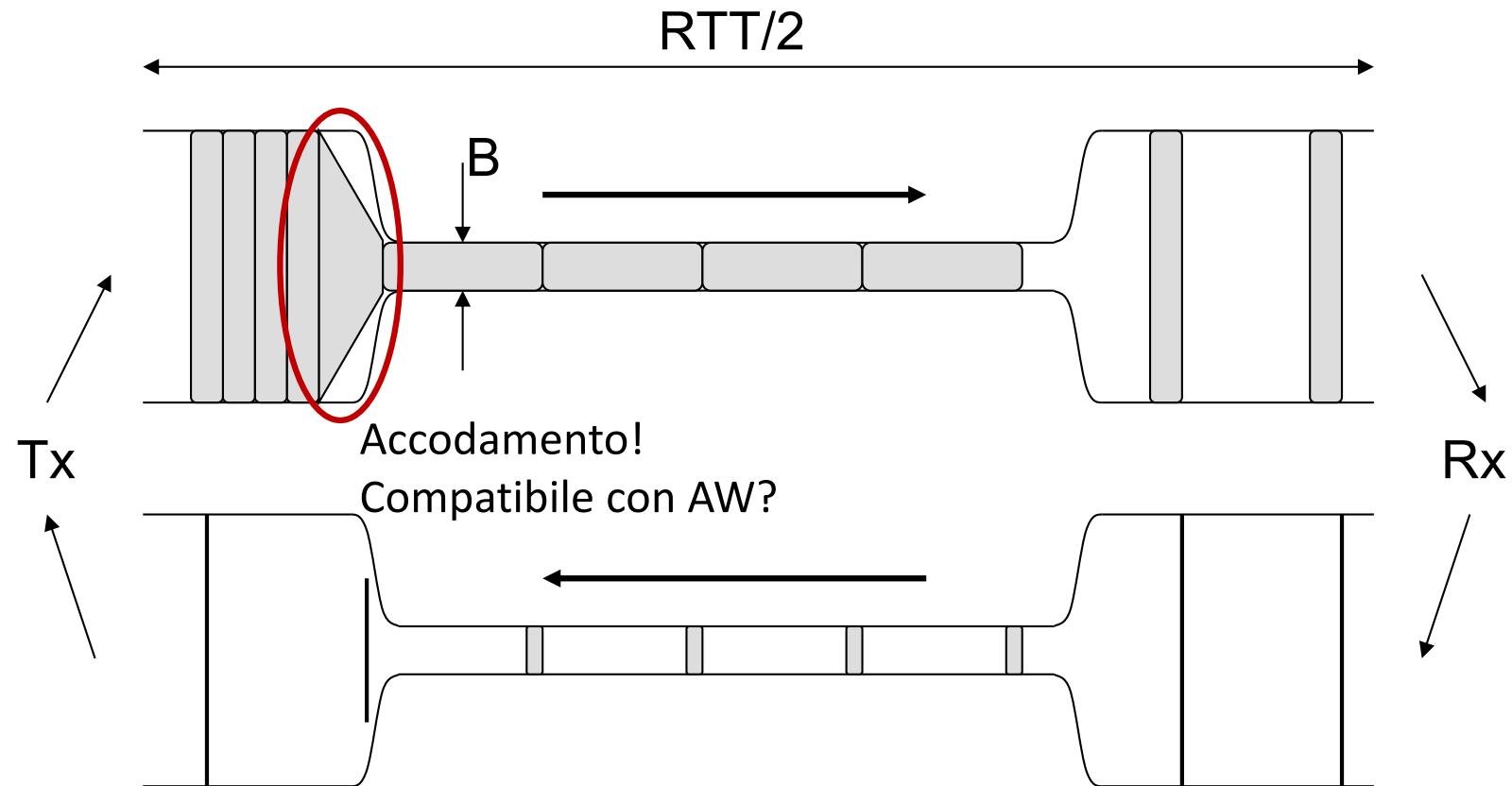


Nel mondo reale: la rete!





Nel mondo reale: la rete!





Problemi

- Al momento dell' instaurazione della connessione TCP la banda disponibile B è incognita
 - A quale valore si deve impostare CW?
- La banda disponibile B può cambiare durante la connessione
 - CW va adattata dinamicamente alla banda disponibile
- Sono definite due fasi che corrispondono a diverse dinamiche di CW
 - Slow start
 - Per raggiungere velocemente un W prossimo a W_{id}
 - Congestion avoidance
 - Per far sì che W sia il più prossimo possibile a W_{id} durante la connessione



Ipotesi di partenza

- Trasmettitore e ricevitore sono correttamente configurati
 - Non ci sono problemi di silly window syndrome
 - I buffer di trasmissione e ricezione sono abbastanza grandi per le necessità della connessione
 - Le applicazioni non determinano stagnazione dei dati nei buffer di ricezione
- W viene determinata dai meccanismi di controllo della congestione del TCP

$$AW \gg CW \text{ quindi } W = CW$$



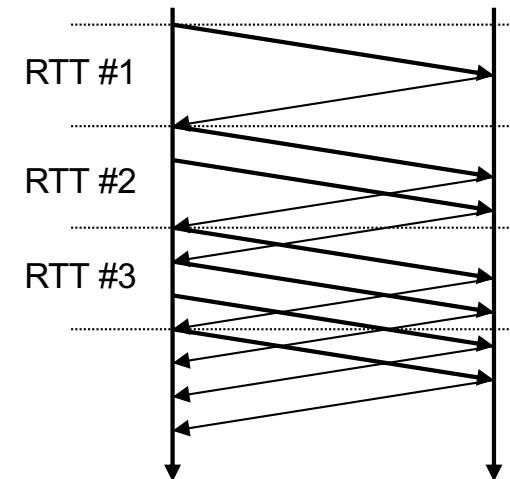
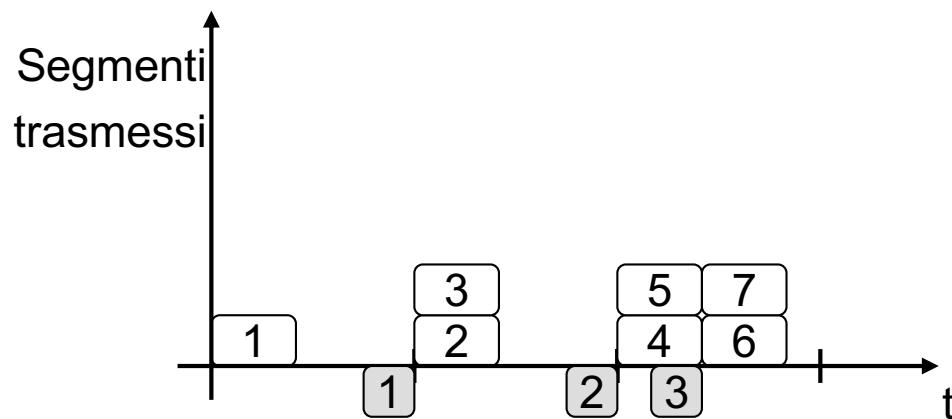
Slow start

- All' inizio della connessione
 $w \leq 2 * \text{MSS}$ e $W \leq 2$
- Per ogni ACK ricevuto senza scadenza di RTO
 $W = W + 1$ ($w \leq w + \text{MSS}$)
- Slow start termina quando
 - Si verifica congestione (no ACK in RTO)
 - $w > \text{ssthrt}$ (slow start threshold)
 - ssthrt all' apertura della connessione può essere configurato ad un valore arbitrariamente alto (uguale a AW oppure a 64 Kbyte)
 - $\text{SSTHR} = \text{ssthrt}/\text{MSS}$ limite per W
- Se $w = \text{ssthrt}$ si può usare Slow Start o Congestion Avoidance

Dinamiche di Slow Start

- RTT approssimativamente costante
 - Si ipotizza una situazione di rete abbastanza stazionaria
 - L'evoluzione di W avviene per tempi multipli di RTT
- W ha una **crescita esponenziale** in Slow Start
 - Al termine di ogni RTT la finestra è raddoppiata
- La fase di Slow start dura approssimativamente

$$T_{ss} = \text{RTT} \log_2 \text{SSTHR}$$





Congestion avoidance

- Si passa da una crescita esponenziale ad una crescita lineare
- **w viene incrementata di un MSS per ogni RTT**
 - Fino a quando si verifica congestione oppure si raggiunge AW
- Implementazione dell' incremento:
 - Ricevuto ACK i-esimo
$$W = W + 1/W$$
 - Ricevuti gli ACK di una intera finestra risulta $W \approx W+1$
 - Normalmente si implementa il calcolo in byte
$$w = w + MSS^2/w$$



Il calcolo reale

- Si ipotizzi $W = CW = 4$
 - Vengono trasmessi 4 segmenti
 - Ricevuto il primo ACK risulta
 - $W = 4 + 1/4 = 4,25$
 - Ricevuto il secondo ACK risulta
 - $W = 4,25 + 1/4,25 = 4,49$
 - ...
 - Ricevuto il quarto ACK, ossia terminato il RTT dell'intera finestra
 - $W = 4,92$
 - Non è esattamente $W = W+1$ dopo la ricezione di tutti gli ACK della finestra
 - La crescita di W non è esattamente lineare
- Per semplicità ipotizzeremo la crescita di W strettamente lineare nel tempo

LABORATORY OF NETWORK PROGRAMMABILITY AND AUTOMATION

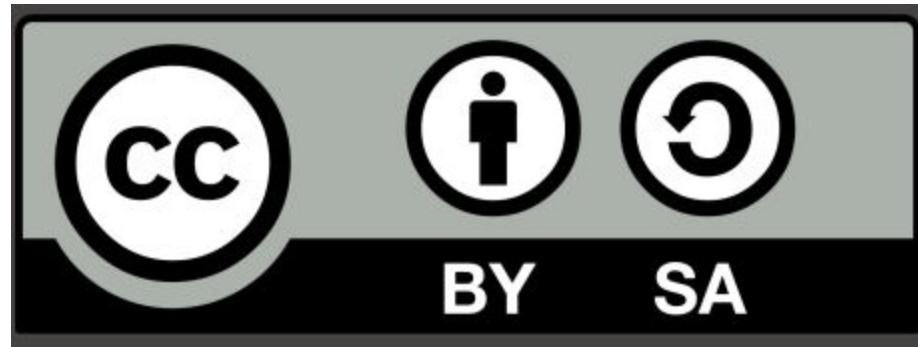
PROGRAMMABLE DATA PLANE WITH P4

Andrea Melis

Computer Science and Engineering Department, University of Bologna

License

Released under CC-BY-SA License



<https://creativecommons.org/licenses/by-sa/4.0/>

Agenda

- Data Plane Introduction
 - What it is?
 - What does it do?
 - Data Plane in a SDN
- Why we need a Programmable Data Plane
 - Challenges
 - Benefits
- P4 Language
 - History
 - WorkFlow
 - Architecture
 - Goal

What is the data plane?

- Processing packet streams
 - Large volume, packets come in streams, algorithms process them
 - super fast → small time to process single packet
 - matching bitfields, simple actions
 - at end hosts → NIC
 - inside the network → router, switch, firewall
- Bunch of different functionality
 - packet forwarding (switch)
 - access control (firewall)
 - tunneling
 - traffic monitoring
 - buffering and marking
 - shaping and scheduling
 - Deep packet inspection (DPI box)



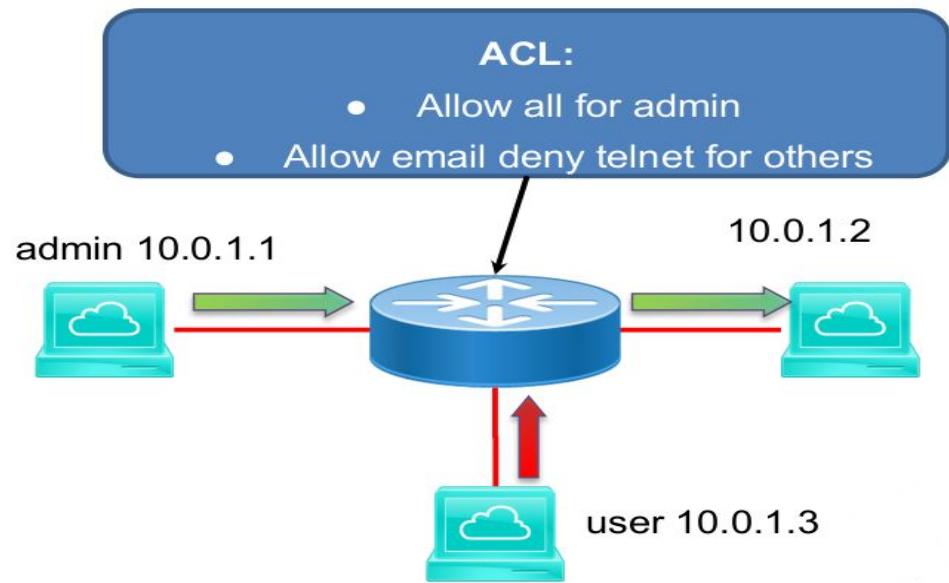
Access Control

- **Packet Filtering → Access Control Lists (ACL)**

- Src, Dst IP address
 - Src, Dst ports
 - Protocol ID

- **Stateful operations**

- also for security,
e.g. attacks
 - e.g. block all TCP syn
packets from outside
 - requires to parse TCP
headers and maintain
flow state



Access Control List

- Accept/Drop actions
 - ordered and list
 - Wildcard rules possible
 - list entries can overlap → priority

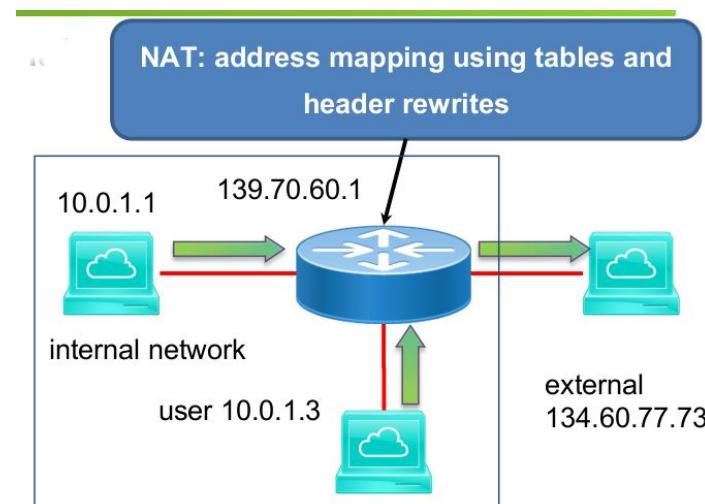
- Packet classification
 - match header fields
 - identify match with highest priority

- Different approaches
 - multi-dimensional classification algorithms
 - Use TCAMs: ternary content addressable memory

Src=1.2.3.4, Dest=5.6.7.8	accept
Dest=1.2.3.*	drop
Dest=1.2.3.8, Dport!=53	accept
Src=1.2.3.7, Dport=100	accept
Dport=100	drop

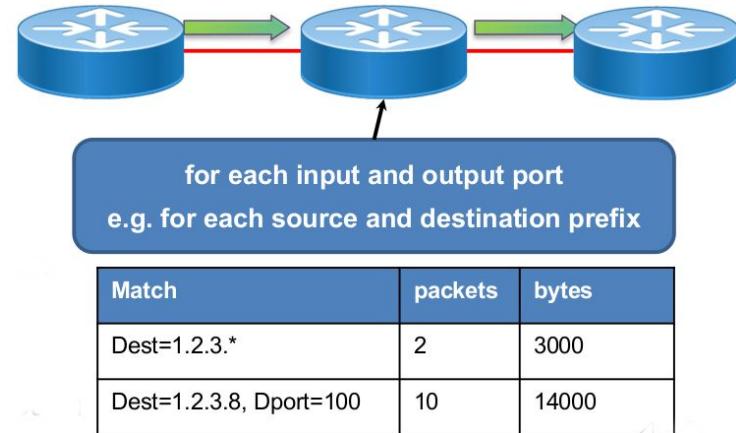
Network Address Translation - NAT

- Mapping between internal and external addresses
 - IP-addresses: between end-hosts and NAT
 - ports: each connection needs to be unique
- NAT Table
 - entries are dynamically created
 - when to remove entries?
 - what if both ends are behind NAT?



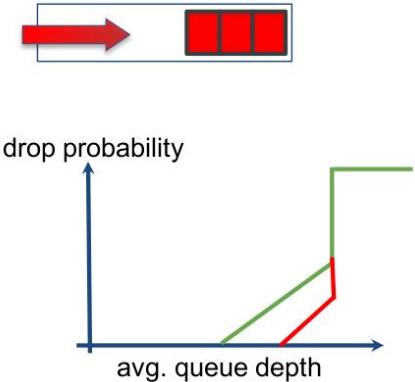
Traffic Monitoring

- Why Traffic Monitoring?
 - volume based charging, traffic engineering,
 - anomaly detection, ...
- How?
 - matching header fields
 - updating counter of packets/bytes
- Challenges
 - identify correct aggregates: proactive vs. reactive
 - more information, e.g. time in queue, congestion states,..
 - some packets of a flow might pass through other nodes, e.g. MPTCP



Buffering and Queue Management

- First In First Out (FIFO) → Drop Tail
 - packets served in arriving order
 - if queue is full, arriving packet is dropped
- Random Early Detection (RED)
 - drop earlier (function of buffer size)
 - or mark to signal congestion to end hosts
 - different traffic classes can be handled differently
- Multiple Traffic Classes
 - separate FIFO queue
 - for each flow or traffic class (e.g. voice, video)
 - need scheduler to decide serving order
- Active Queue Management (AQM)
 - queue autotunes itself to e.g. latency target
 - CoDel, PIE, FqCoDel,...
 - Packet Value based dropping

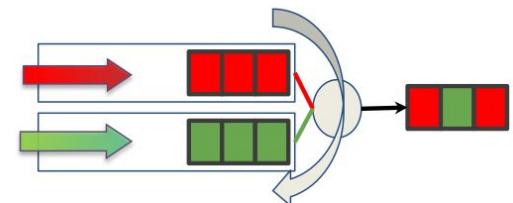
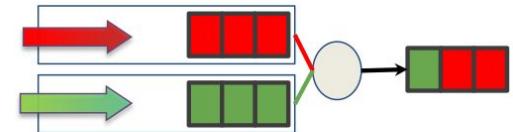


Packet Marking and Traffic Classification

- **Mark a packet to signal downstream or end-hosts**
 - ECN - Early Congestion notification
 - Reuse some of the IP header fields, e.g. Type of Service (ToS) bits can use bufferstate → RED, CoDel,...
- **How to mark?**
 - End hosts based on applications
 - how can the network trust the endhosts?
 - Network nodes based on traffic classes
 - how can the network infer application requirements?
- **How to identify traffic classes?**
 - using flow specification based on five tuple
 - rate limitations, what conforms to profile, what is out of profile

Packet Scheduling

- Determines the serving order of packets
 - when there are multiple queues to serve
 - multiple algorithms, different complexity
- Strict priority
 - assign to each queue a priority number
 - serve always the queue with highest priority first if it has packets
- Round Robin
 - go through queues in round robin way
 - if packet in the queue, serve, otherwise check next one
- Weighted Fair Scheduling
 - assign weights to queues
 - serve proportionally many packets



- WHAT ABOUT SDN?

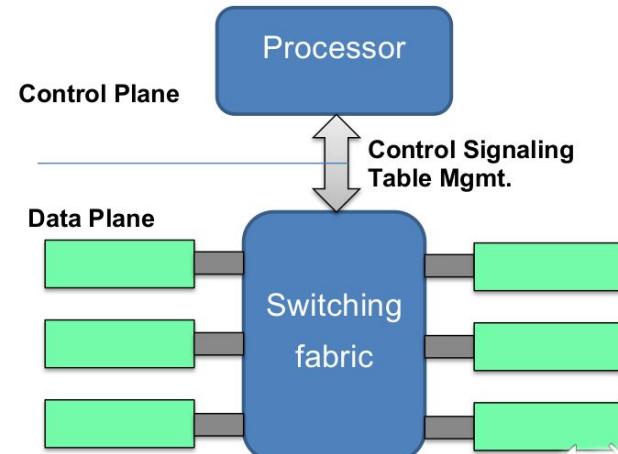
SDN

- Control Plane

- calculates the forwarding table
- determines output port based on
- destination address

- Data Plane

- manages individual incoming packets
- matches destination address
 - switch: Dst MAC addr
 - routers: longest IP Prefix
- lookup the output port
- action: the packet is sent to output port
- switching fabric: directs packets from input to output



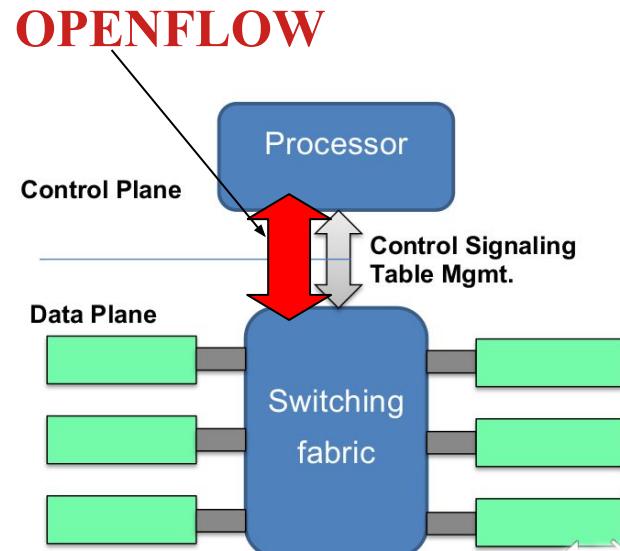
SDN

- Control Plane

- calculates the forwarding table
 - determines output port based on
 - destination address

- Data Plane

- manages individual incoming packets
 - matches destination address
 - switch: Dst MAC addr
 - routers: longest IP Prefix
 - lookup the output port
 - action: the packet is sent to output port
 - switching fabric: directs packets from input to output



OpenFlow

- Match

- Header fields
 - Subset
 - IP, MAC, MPLS
- Ports
- supported by most TCAM implementations

- Actions

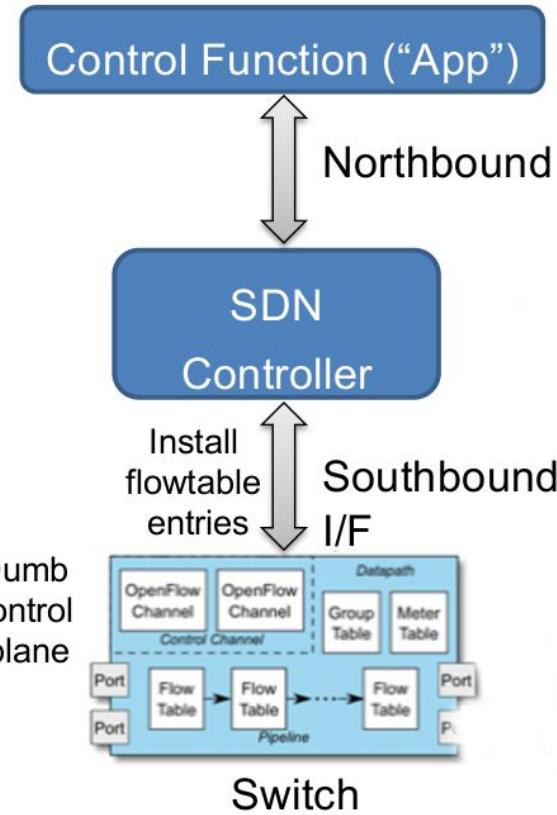
- If match found, perform action on
 - packet
 - drop
 - rewrite header fields
 - forward on port X
 - count packets
 - flood

Prio	Match	Action
1	Src=1.2.3.4, Dest=5.6.7.8	forward (1)
2	Dest=1.2.3.*	drop
3	Dest=1.2.3.8, Dport!=53	forward (2)
4	Src=1.2.3.7, Dport=100	forward (3)
5	Dport=100	forward controller

OpenFlow and SDN

- Pros

- OpenFlow = standardized protocol to interact with switch
 - download flow table entries, query statistics, etc.
 - IP, MAC, MPLS
- OpenFlow = standardized model
 - match/action abstraction
- Concept of logically centralized control via a single entity (“SDN controller”)
 - Simplifies control plane



OpenFlow ... in the Beginning

- OpenFlow was simple
- A single rule table
 - Priority, pattern, actions, counters, timeouts
- Matching on any of 12 fields, e.g.,
 - MAC addresses
 - IP addresses
 - Transport protocol
 - Transport port numbers

OpenFlow ... then

- Proliferation of header fields

Version	Date	# Headers
OF 1.0	Dec 2009	12
OF 1.1	Feb 2011	15
OF 1.2	Dec 2011	36
OF 1.3	Jun 2012	40
OF 1.4	Oct 2013	41

- Things are changing...

Future SDN Switches

- Configurable packet parser
 - Not tied to a specific header format
- Flexible match+action tables
 - Multiple tables (in series and/or parallel)
 - Able to match on all defined fields
- General packet-processing primitives
 - Copy, add, remove, and modify
 - For both header fields and meta-data

Need Hardware

- Configurable switch needs a programmable switch
- Programmable switches “always” existed
- Issues
 - Hard to program
 - Lack of standard
 - Lack of common interface
 - Definitely not performant
 - Definitely not supported

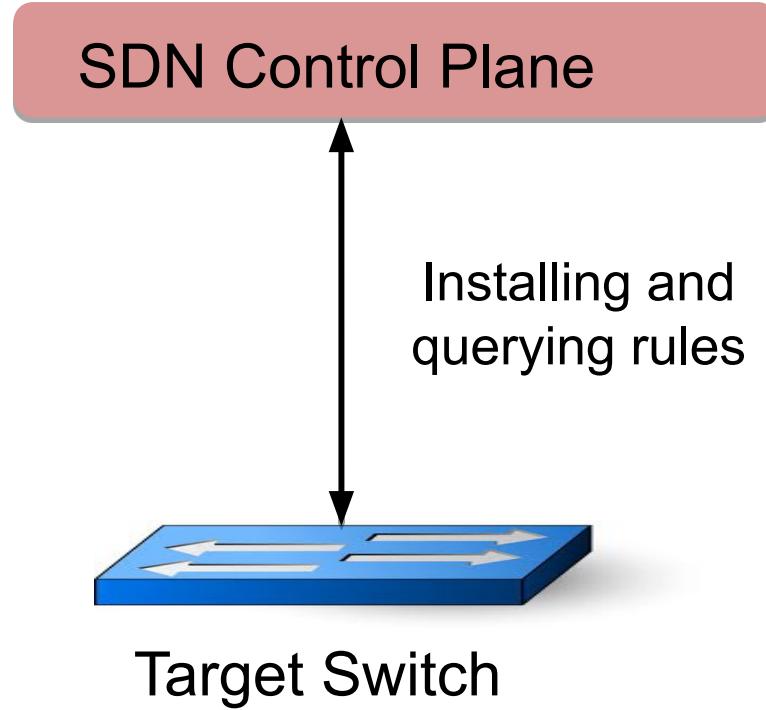
Need Hardware

- Things changed
- New generation of switch ASICs
 - Intel FlexPipe
 - Cisco Doppler
 - Tofino
- Pros
 - Performances:
 - <http://conferences.sigcomm.org/sigcomm/2013/papers/sigcomm/p99.pdf>
- Cons
 - Still need a standard interface.
 - Custom, vendor-specific interfaces
 - Low-level, akin to microcode programming

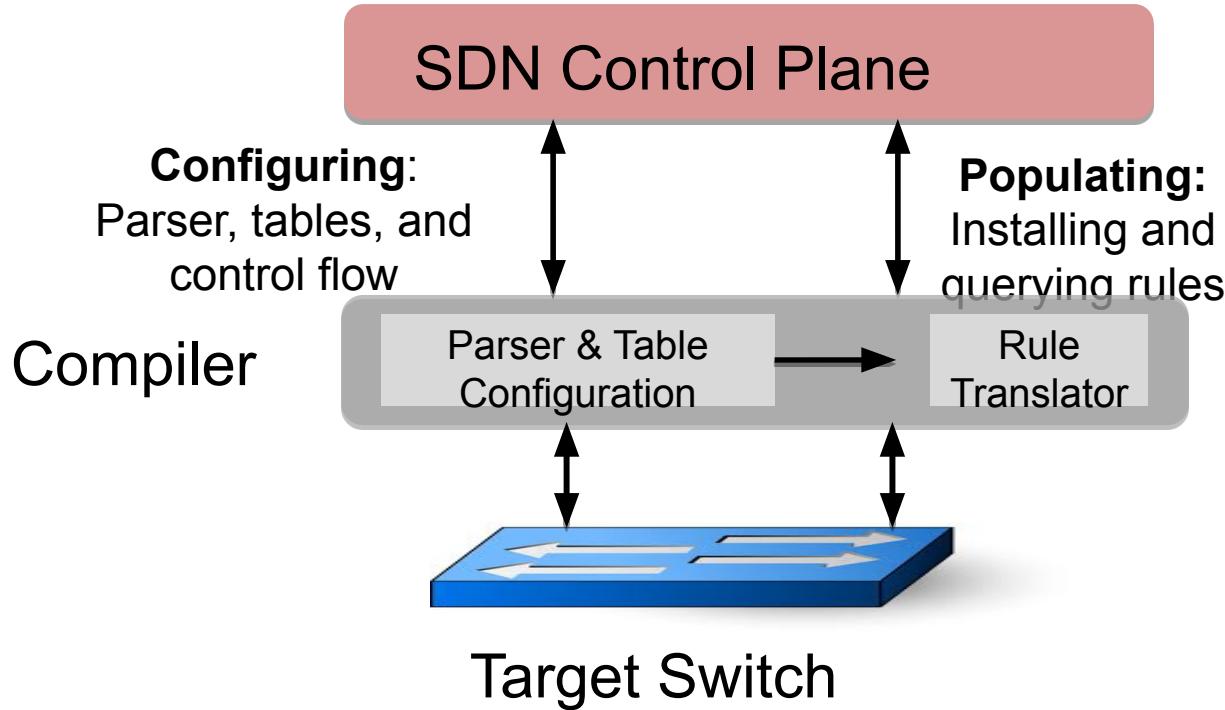
Need a Programmable Interface

- Three goals
- 1) Protocol independence
 - Configure a packet parser
 - Define a set of typed match+action tables
- 2) Target independence
 - Program without knowledge of switch details
 - Rely on compiler to configure the target switch
- 3) Reconfigurability
 - Change parsing and processing in the field

OpenFlow



New OpenFlow



What is Data Plane Programming?

- **Why program the Data Plane?**

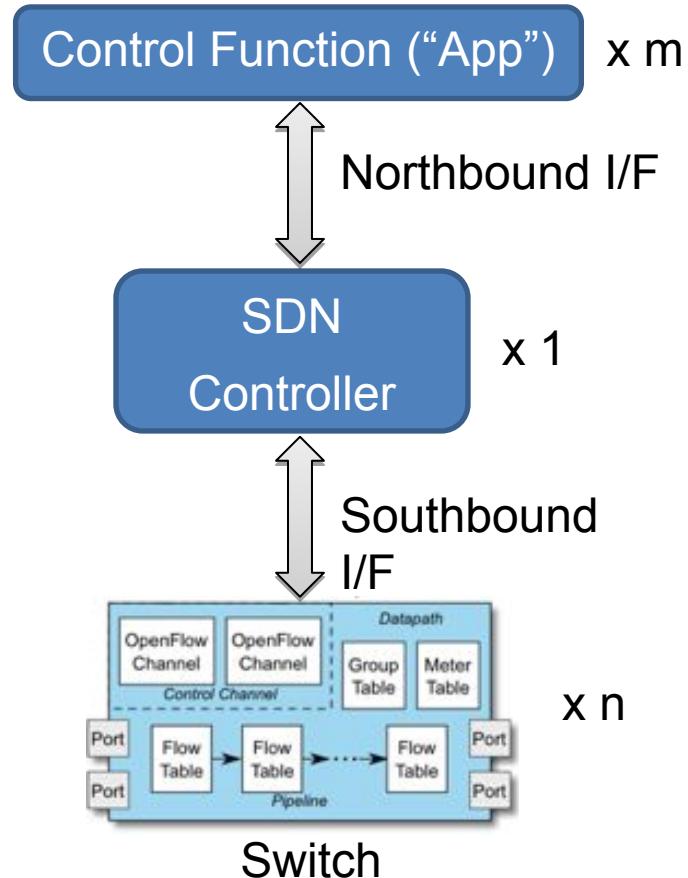
Software Defined Networking (SDN)

- **Main contributions**

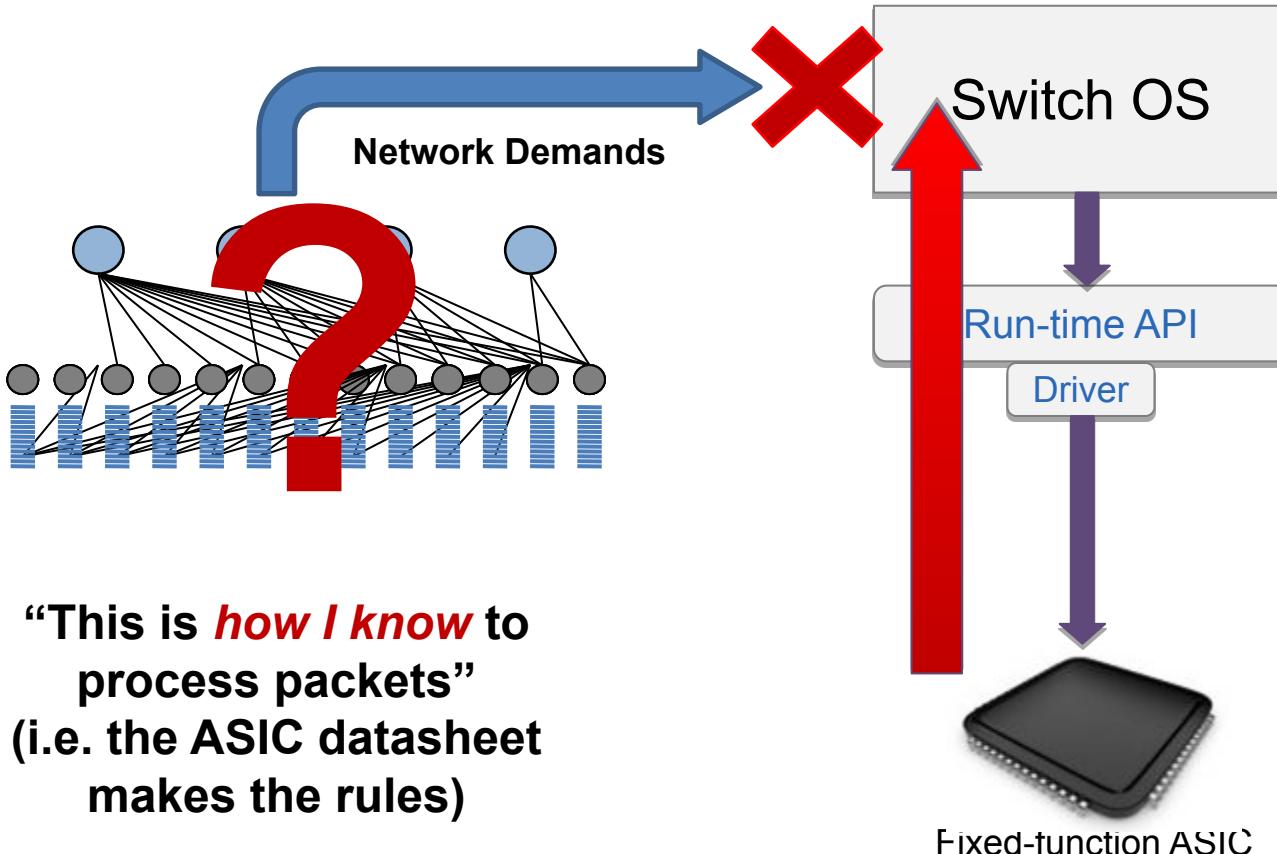
- OpenFlow = standardized *protocol* to interact with switch
 - download flow table entries, query statistics, etc.
- OpenFlow = standardized *model*
 - match/action abstraction
- *Concept of logically centralized control via a single entity (“SDN controller”)*
 - Simplifies control plane

- **Issues**

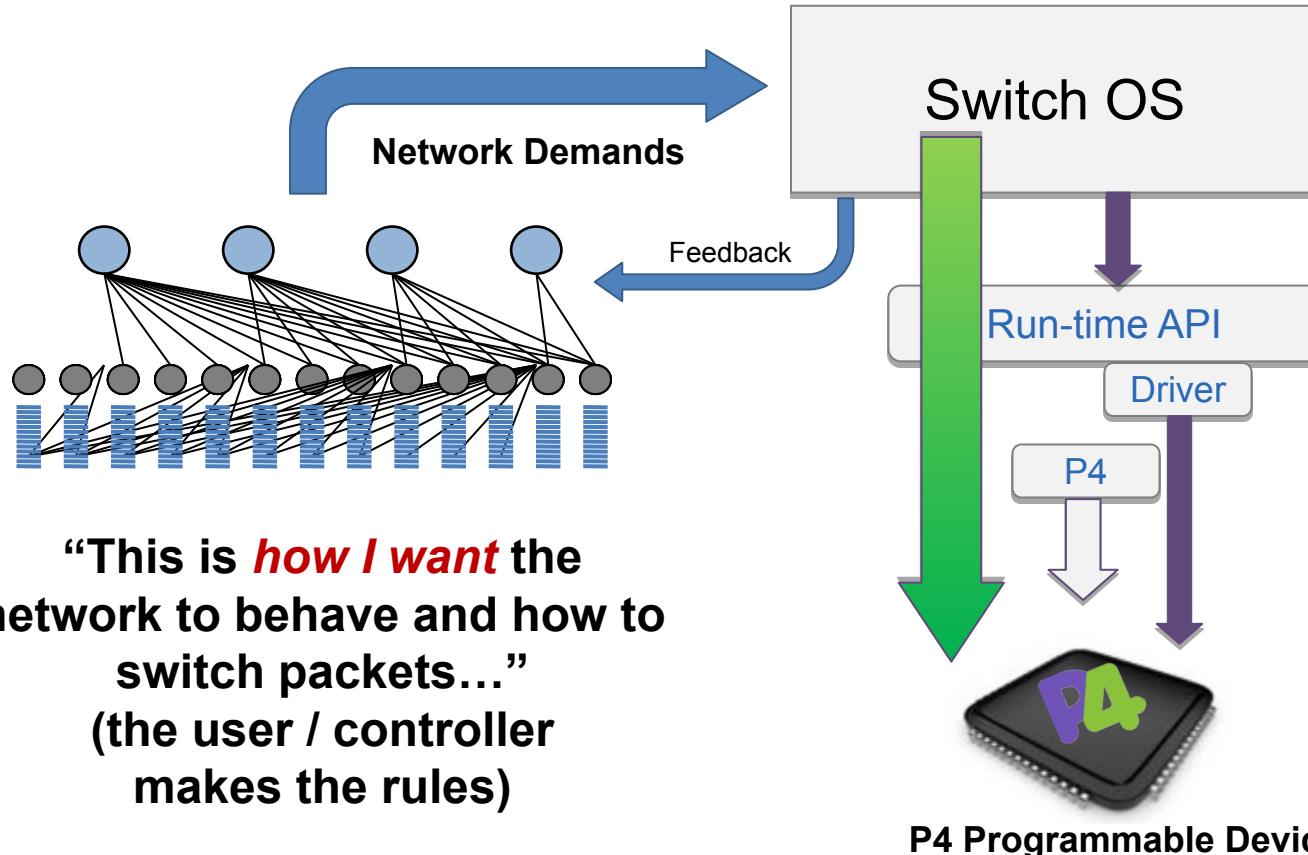
- Data-plane protocol evolution requires changes to standards (12 → 40 OpenFlow match fields)
- Limited interoperability between vendors (OpenFlow / netconf / JSON / XML variants)
- Limited programmability



Status Quo: Bottom-up design



A Better Approach: Top-down design



Benefits of Data Plane Programmability

- **New Features** – Add new protocols
- **Reduce complexity** – Remove unused protocols
- **Efficient use of resources** – flexible use of tables
- **Greater visibility** – New diagnostic techniques, telemetry, etc.
- **SW style development** – rapid design cycle, fast innovation, fix data plane bugs in the field
- **You keep your own ideas**

Think programming rather than protocols...

Programmable Network Devices

- **PISA: Flexible Match+Action ASICs**
 - Intel Flexpipe, Cisco Doppler, Cavium (Xpliant), Barefoot Tofino, ...
- **NPU**
 - EZchip, Netronome, ...
- **CPU**
 - Open Vswitch, eBPF, DPDK, VPP...
- **FPGA**
 - Xilinx, Altera, ...

These devices let us tell them how to process packets.

What can you do with P4?

- Layer 4 Load Balancer – SilkRoad[1]
- Low Latency Congestion Control – NDP[2]
- In-band Network Telemetry – INT[3]
- In-Network caching and coordination – NetCache[4] / NetChain[5]
- Aggregation for MapReduce Applications [7]
- ... and much more

[1] Miao, Rui, et al. "SilkRoad: Making Stateful Layer-4 Load Balancing Fast and Cheap Using Switching ASICs." SIGCOMM, 2017.

[2] Handley, Mark, et al. "Re-architecting datacenter networks and stacks for low latency and high performance." SIGCOMM, 2017.

[3] Kim, Changhoon, et al. "In-band network telemetry via programmable dataplanes." SIGCOMM. 2015.

[4] Xin Jin et al. "NetCache: Balancing Key-Value Stores with Fast In-Network Caching." To appear at SOSP 2017

[5] Jin, Xin, et al. "NetChain: Scale-Free Sub-RTT Coordination." NSDI, 2018.

[6] Dang, Huynh Tu, et al. "NetPaxos: Consensus at network speed." SIGCOMM, 2015.

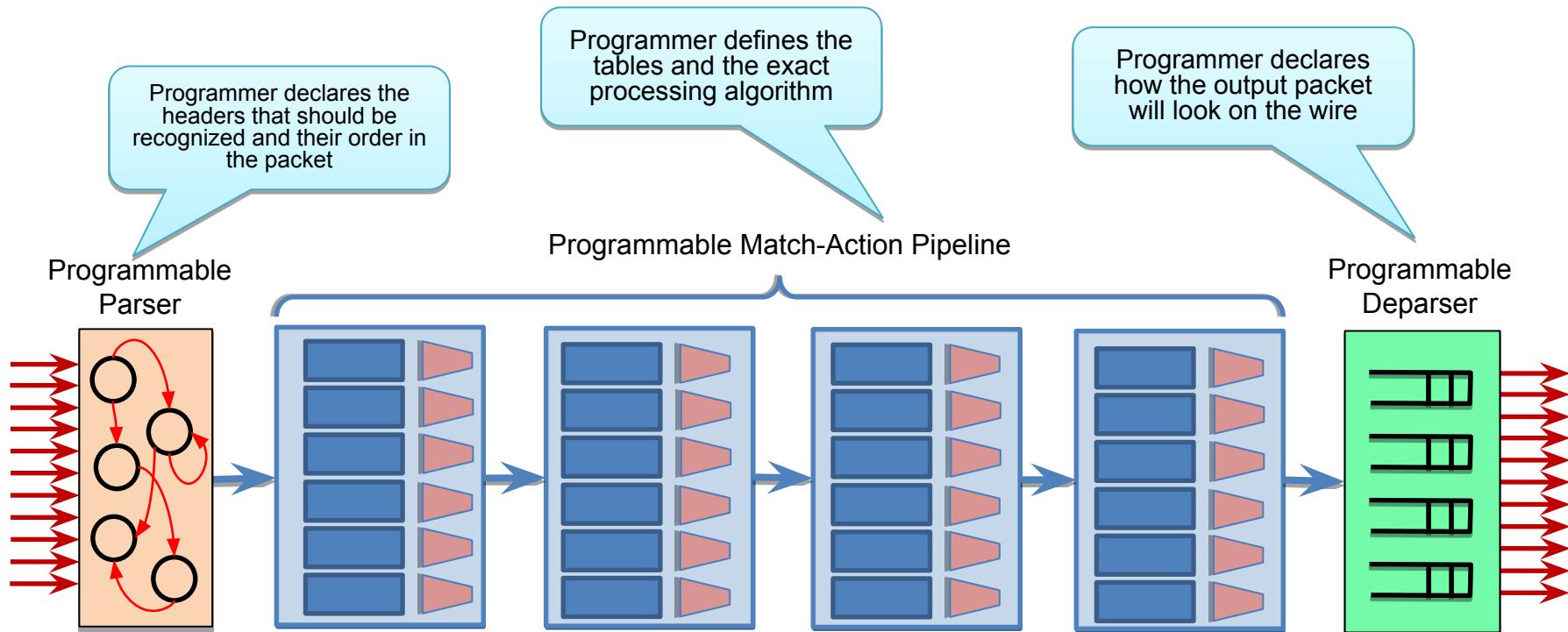
[7] Sapiro, Amedeo, et al. "In-Network Computation is a Dumb Idea Whose Time Has Come." *Hot Topics in Networks*. ACM, 2017.

Brief History and Trivia

- May 2013: Initial idea and the name “P4”
- July 2014: First paper (SIGCOMM CCR)
- Aug 2014: First P4₁₄ Draft Specification (v0.9.8)
- Sep 2014: P4₁₄ Specification released (v1.0.0)
- Jan 2015: P4₁₄ v1.0.1
- Mar 2015: P4₁₄ v1.0.2
- Nov 2016: P4₁₄ v1.0.3
- May 2017: P4₁₄ v1.0.4
- Apr 2016: P4₁₆ – first commits
- Dec 2016: First P4₁₆ Draft Specification
- May 2017: P4₁₆ Specification released

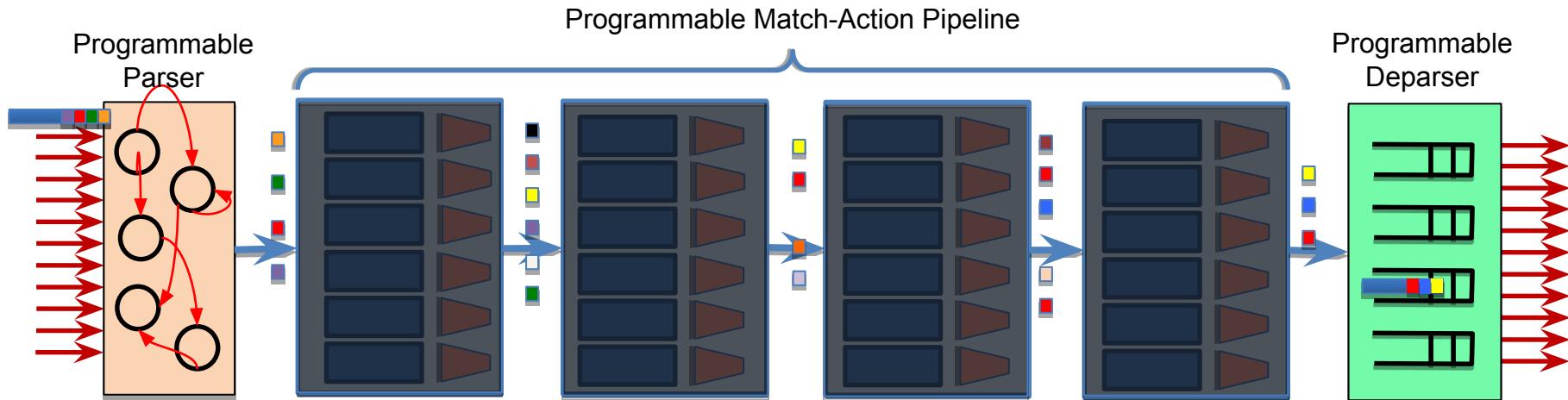
P4_16 Data Plane Model

PISA: Protocol-Independent Switch Architecture

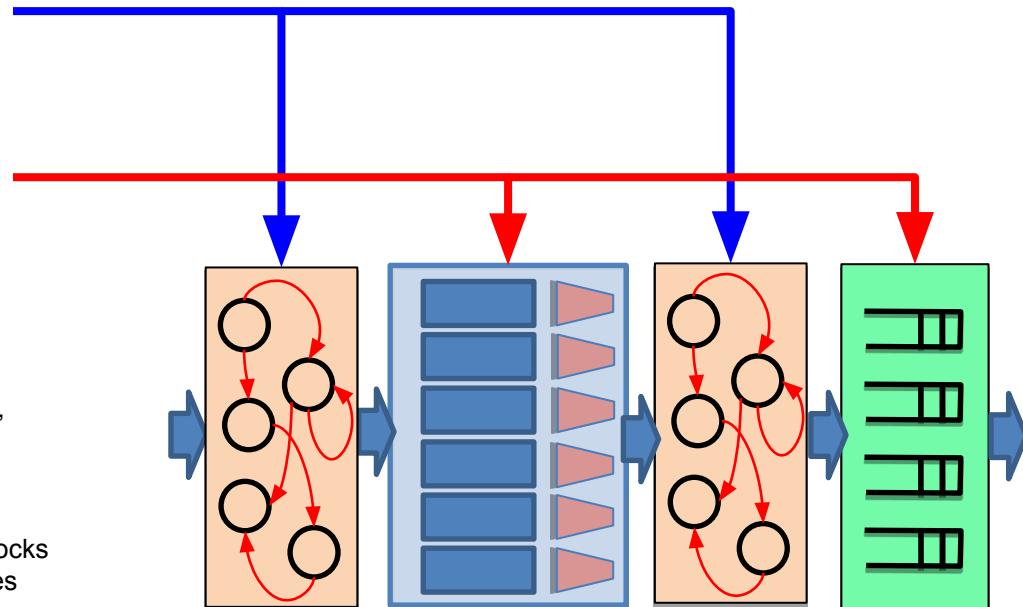


PISA in Action

- Packet is parsed into individual headers (parsed representation)
- Headers and intermediate results can be used for matching and actions
- Headers can be modified, added or removed
- Packet is deparsed (serialized)

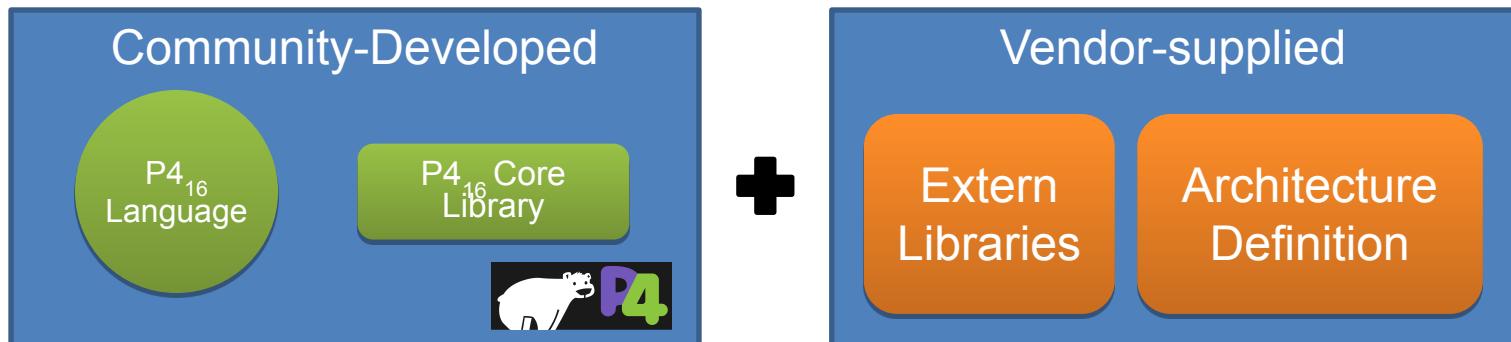


P4₁₆ Language Elements

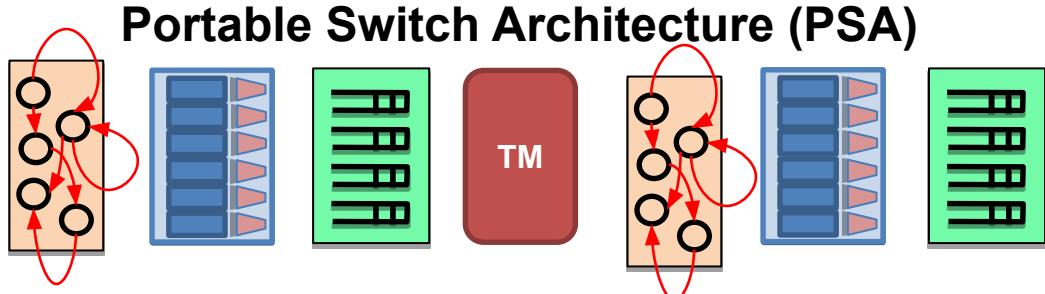
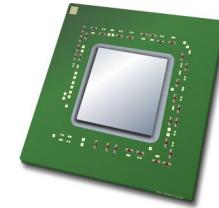
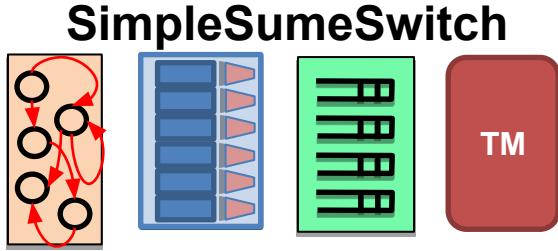
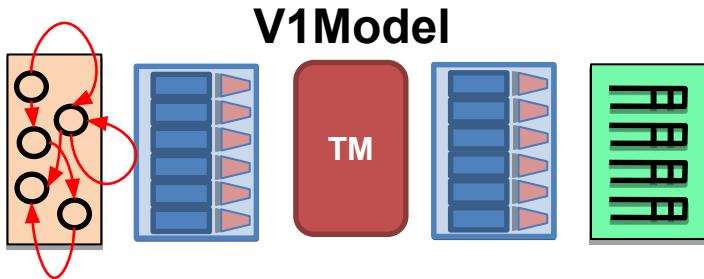


P4_16 Approach

Term	Explanation
P4 Target	An embodiment of a specific hardware implementation
P4 Architecture	Provides an interface to program a target via some set of P4-programmable components, externs, fixed components

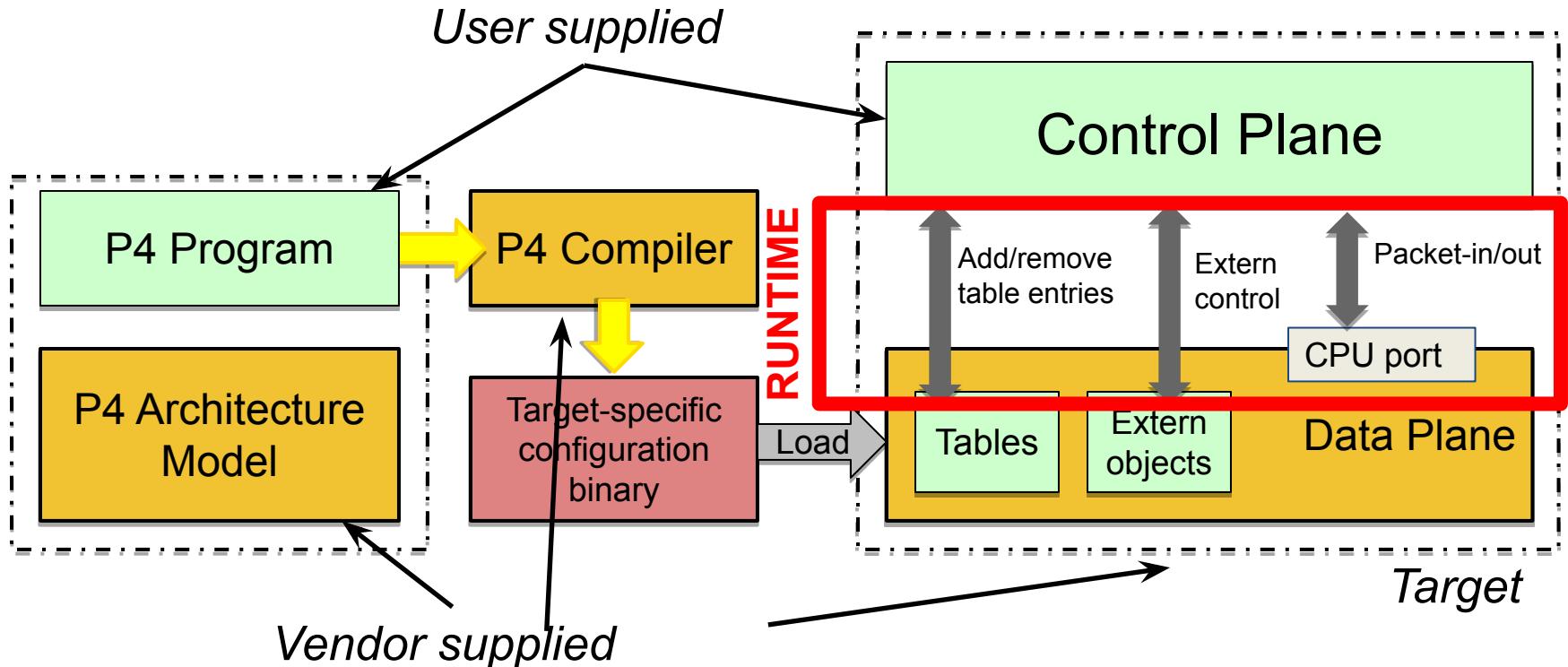


Example Architectures and Targets



Anything

Programming a P4 Target



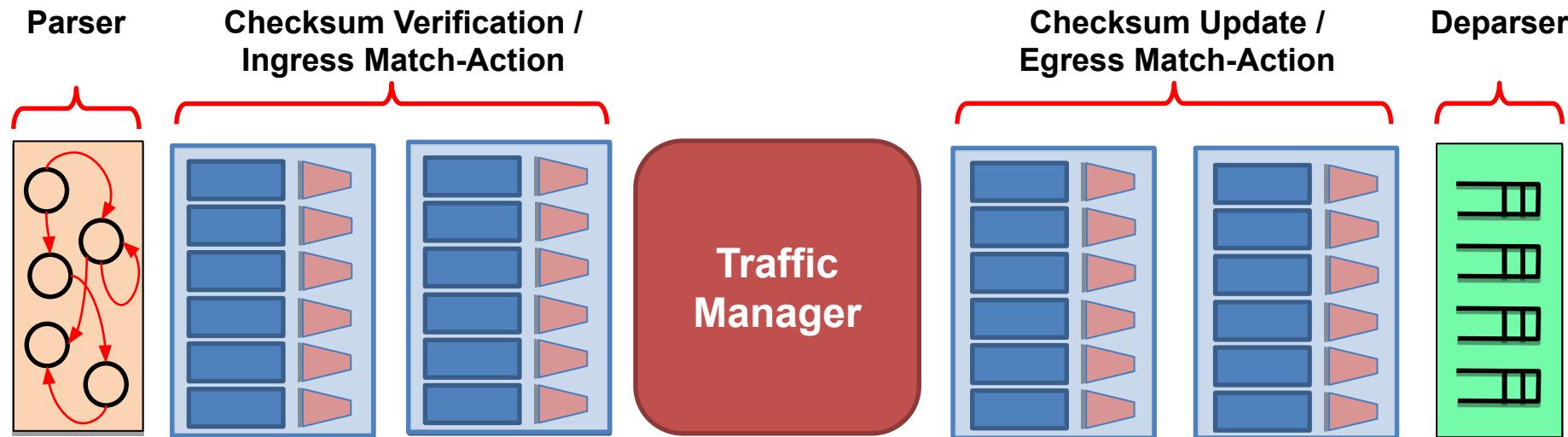
Lab 1: Basics

Before we start...

- **Install VM image**
- **Exercises under folder**
 - **/vt19/..**

V1Model Architecture

- Implemented on top of Bmv2's simple_switch target



V1Model Standard Metadata

```
struct standard_metadata_t {  
    bit<9> ingress_port;  
    bit<9> egress_spec;  
    bit<9> egress_port;  
    bit<32> clone_spec;  
    bit<32> instance_type;  
    bit<1> drop;  
    bit<16> recirculate_port;  
    bit<32> packet_length;  
    bit<32> enq_timestamp;  
    bit<19> enq_qdepth;  
    bit<32> deq_timedelta;  
    bit<19> deq_qdepth;  
    bit<48> ingress_global_timestamp;  
    bit<32> lf_field_list;  
    bit<16> mcast_grp;  
    bit<1> resubmit_flag;  
    bit<16> egress_rid;  
    bit<1> checksum_error;  
}
```

- **ingress_port** - the port on which the packet arrived
- **egress_spec** - the port to which the packet should be sent to
- **egress_port** - the port that the packet will be sent out of (read only in egress pipeline)

P4₁₆ Program Template (V1Model)

```
#include <core.p4>
#include <v1model.p4>
/* HEADERS */
struct metadata { ... }
struct headers {
    ethernet_t    ethernet;
    ipv4_t         ipv4;
}
/* PARSER */
parser MyParser(packet_in packet,
                 out headers hdr,
                 inout metadata meta,
                 inout standard_metadata_t smeta) {
    ...
}
/* CHECKSUM VERIFICATION */
control MyVerifyChecksum(in headers hdr,
                         inout metadata meta) {
    ...
}
/* INGRESS PROCESSING */
control MyIngress(inout headers hdr,
                  inout metadata meta,
                  inout standard_metadata_t std_meta) {
    ...
}
```

```
/* EGRESS PROCESSING */
control MyEgress(inout headers hdr,
                  inout metadata meta,
                  inout standard_metadata_t std_meta) {
    ...
}
/* CHECKSUM UPDATE */
control MyComputeChecksum(inout headers hdr,
                          inout metadata meta) {
    ...
}
/* DEPARSER */
control MyDeparser(inout headers hdr,
                   inout metadata meta) {
    ...
}
/* SWITCH */
V1Switch(
    MyParser(),
    MyVerifyChecksum(),
    MyIngress(),
    MyEgress(),
    MyComputeChecksum(),
    MyDeparser()
) main;
```

P4₁₆ Hello World (V1Model)

```
#include <core.p4>
#include <v1model.p4>
struct metadata {}
struct headers {}

parser MyParser(packet_in packet,
    out headers hdr,
    inout metadata meta,
    inout standard_metadata_t standard_metadata) {

    state start { transition accept; }

control MyVerifyChecksum(inout headers hdr, inout metadata meta) { apply { } }

control MyIngress(inout headers hdr,
    inout metadata meta,
    inout standard_metadata_t standard_metadata) {
apply {
    if (standard_metadata.ingress_port == 1) {
        standard_metadata.egress_spec = 2;
    } else if (standard_metadata.ingress_port == 2) {
        standard_metadata.egress_spec = 1;
    }
}
}
```

```
control MyEgress(inout headers hdr,
    inout metadata meta,
    inout standard_metadata_t standard_metadata) {
    apply { }
}

control MyComputeChecksum(inout headers hdr, inout metadata meta) {
    apply { }
}

control MyDeparser(packet_out packet, in headers hdr) {
    apply { }
}

V1Switch(
    MyParser(),
    MyVerifyChecksum(),
    MyIngress(),
    MyEgress(),
    MyComputeChecksum(),
    MyDeparser()
) main;
```

P4₁₆ Hello World (V1Model)

```
#include <core.p4>
#include <v1model.p4>
struct metadata {}
struct headers {}

parser MyParser(packet_in packet, out headers hdr,
    inout metadata meta,
    inout standard_metadata_t standard_metadata) {
    state start { transition accept; }
}

control MyIngress(inout headers hdr, inout metadata meta,
    inout standard_metadata_t standard_metadata) {
    action set_egress_spec(bit<9> port) {
        standard_metadata.egress_spec = port;
    }
}






```

```
control MyEgress(inout headers hdr,
    inout metadata meta,
    inout standard_metadata_t standard_metadata) {
    apply {   }
}

control MyVerifyChecksum(inout headers hdr, inout metadata meta) {   apply {   }   }

control MyComputeChecksum(inout headers hdr, inout metadata meta) {   apply {   }   }

control MyDeparser(packet_out packet, in headers hdr) {
    apply {   }
}

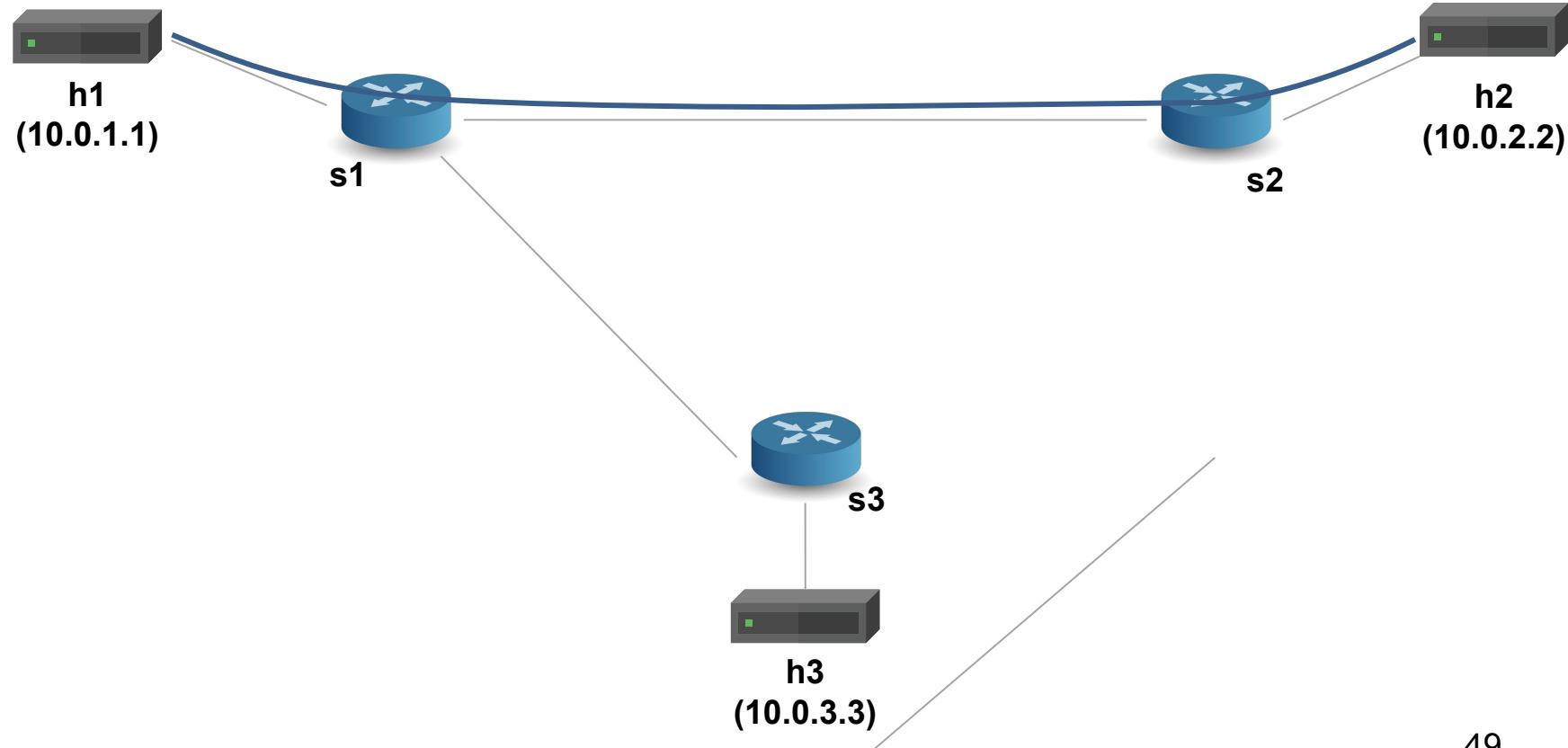
V1Switch( MyParser(), MyVerifyChecksum(), MyIngress(),
MyEgress(), MyComputeChecksum(), MyDeparser() ) main;
```

Key	Action ID	Action Data
1	set_egress_spec ID	2
2	set_egress_spec ID	1

Running Example: Basic Forwarding

- We'll use a simple application as a running example—a basic router—to illustrate the main features of P4₁₆
- Basic router functionality:
 - Parse Ethernet and IPv4 headers from packet
 - Find destination in IPv4 routing table
 - Update source / destination MAC addresses
 - Decrement time-to-live (TTL) field
 - Set the egress port
 - Deparse headers back into a packet
- We've written some starter code for you (`basic.p4`) and implemented a static control plane

Basic Forwarding: Topology



P4₁₆ Types (Basic and Header Types)

```
typedef bit<48> macAddr_t;
typedef bit<32> ip4Addr_t;
header ethernet_t {
    macAddr_t dstAddr;
    macAddr_t srcAddr;
    bit<16> etherType;
}

header ipv4_t {
    bit<4> version;
    bit<4> ihl;
    bit<8> diffserv;
    bit<16> totalLen;
    bit<16> identification;
    bit<3> flags;
    bit<13> fragOffset;
    bit<8> ttl;
    bit<8> protocol;
    bit<16> hdrChecksum;
    ip4Addr_t srcAddr;
    ip4Addr_t dstAddr;
}
```

Basic Types

- **bit<n>**: Unsigned integer (bitstring) of size n
- **bit** is the same as **bit<1>**
- **int<n>**: Signed integer of size n (≥ 2)
- **varbit<n>**: Variable-length bitstring

Header Types:

Ordered collection of members

- Can contain **bit<n>**, **int<n>**, and **varbit<n>**
- Byte-aligned
- Can be valid or invalid
- Provides several operations to test and set validity bit:
isValid(), **setValid()**, and **setInvalid()**

Typedef:

Alternative name for a type

P4₁₆ Types (Other Types)

```
/* Architecture */
struct standard_metadata_t {
    bit<9> ingress_port;
    bit<9> egress_spec;
    bit<9> egress_port;
    bit<32> clone_spec;
    bit<32> instance_type;
    bit<1> drop;
    bit<16> recirculate_port;
    bit<32> packet_length;
    ...
}

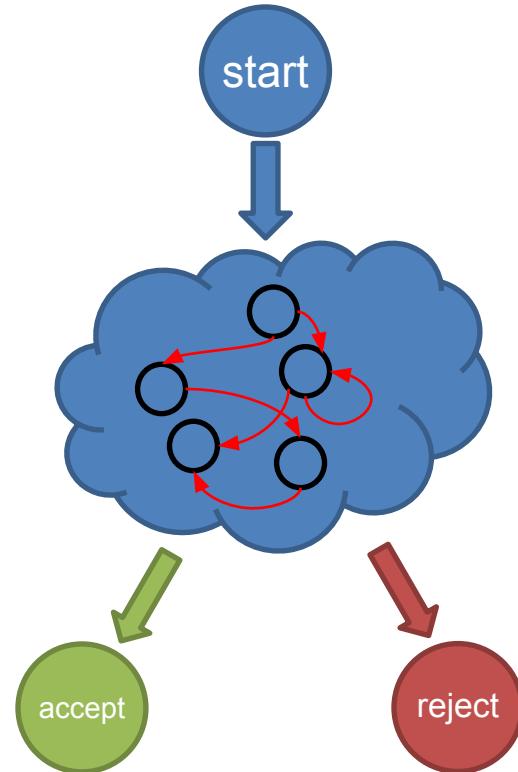
/* User program */
struct metadata {
    ...
}
struct headers {
    ethernet_t   ethernet;
    ipv4_t        ipv4;
}
```

Other useful types

- **Struct:** Unordered collection of members (with no alignment restrictions)
- **Header Stack:** array of headers
- **Header Union:** one of several headers

P4₁₆ Parsers

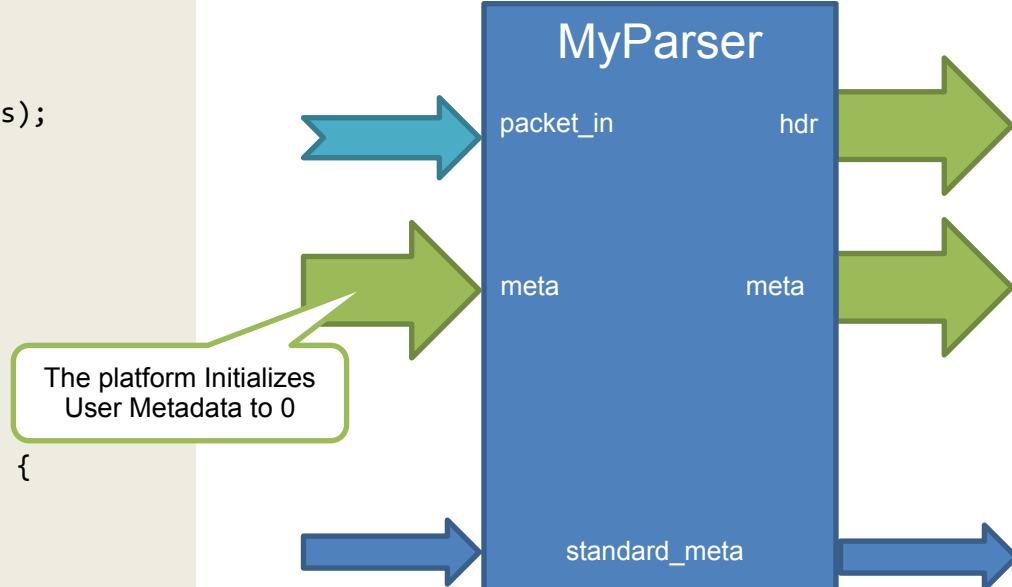
- **Parsers are functions that map packets into headers and metadata, written in a state machine style**
- **Every parser has three predefined states**
 - start
 - accept
 - reject
- **Other states may be defined by the programmer**
- **In each state, execute zero or more statements, and then transition to another state (loops are OK)**



Parsers (V1Model)

```
/* From core.p4 */
extern packet_in {
    void extract<T>(out T hdr);
    void extract<T>(out T variableSizeHeader,
                     in bit<32> variableFieldSizeInBits);
    T lookahead<T>();
    void advance(in bit<32> sizeInBits);
    bit<32> length();
}
/* User Program */
parser MyParser(packet_in packet,
                out headers hdr,
                inout metadata meta,
                inout standard_metadata_t std_meta) {

    state start {
        packet.extract(hdr.ethernet);
        transition accept;
    }
}
```



Select Statement

```
state start {  
    transition parse_ethernet;  
}  
  
state parse_ethernet {  
    packet.extract(hdr.ethernet);  
    transition select(hdr.ethernet.etherType) {  
        0x800: parse_ip4;  
        default: accept;  
    }  
}
```

P4₁₆ has a select statement that can be used to branch in a parser

Similar to case statements in C or Java, but without “fall-through behavior”—i.e., break statements are not needed

In parsers it is often necessary to branch based on some of the bits just parsed

For example, etherType determines the format of the rest of the packet

Match patterns can either be literals or simple computations such as masks

Coding Break

P4₁₆ Controls

- Similar to C functions (without loops)
- Can declare variables, create tables, instantiate externs, etc.
- Functionality specified by code in apply statement
- Represent all kinds of processing that are expressible as DAG:
 - Match-Action Pipelines
 - Deparsers
 - Additional forms of packet processing (updating checksums)
- Interfaces with other blocks are governed by user- and architecture-specified types (typically headers and metadata)

Example: Reflector (V1Model)

```
control MyIngress(inout headers hdr,
                  inout metadata meta,
                  inout standard_metadata_t std_meta) {
    /* Declarations region */
    bit<48> tmp;

    apply {
        /* Control Flow */
        tmp = hdr.ethernet.dstAddr;
        hdr.ethernet.dstAddr = hdr.ethernet.srcAddr;
        hdr.ethernet.srcAddr = tmp;
        std_meta.egress_spec = std_meta.ingress_port;
    }
}
```

Desired Behavior:

- Swap source and destination MAC addresses
- Bounce the packet back out on the physical port that it came into the switch on

Example: Simple Actions

```
control MyIngress(inout headers hdr,
                  inout metadata meta,
                  inout standard_metadata_t std_meta) {

    action swap_mac(inout bit<48> src,
                    inout bit<48> dst) {
        bit<48> tmp = src;
        src = dst;
        dst = tmp;
    }

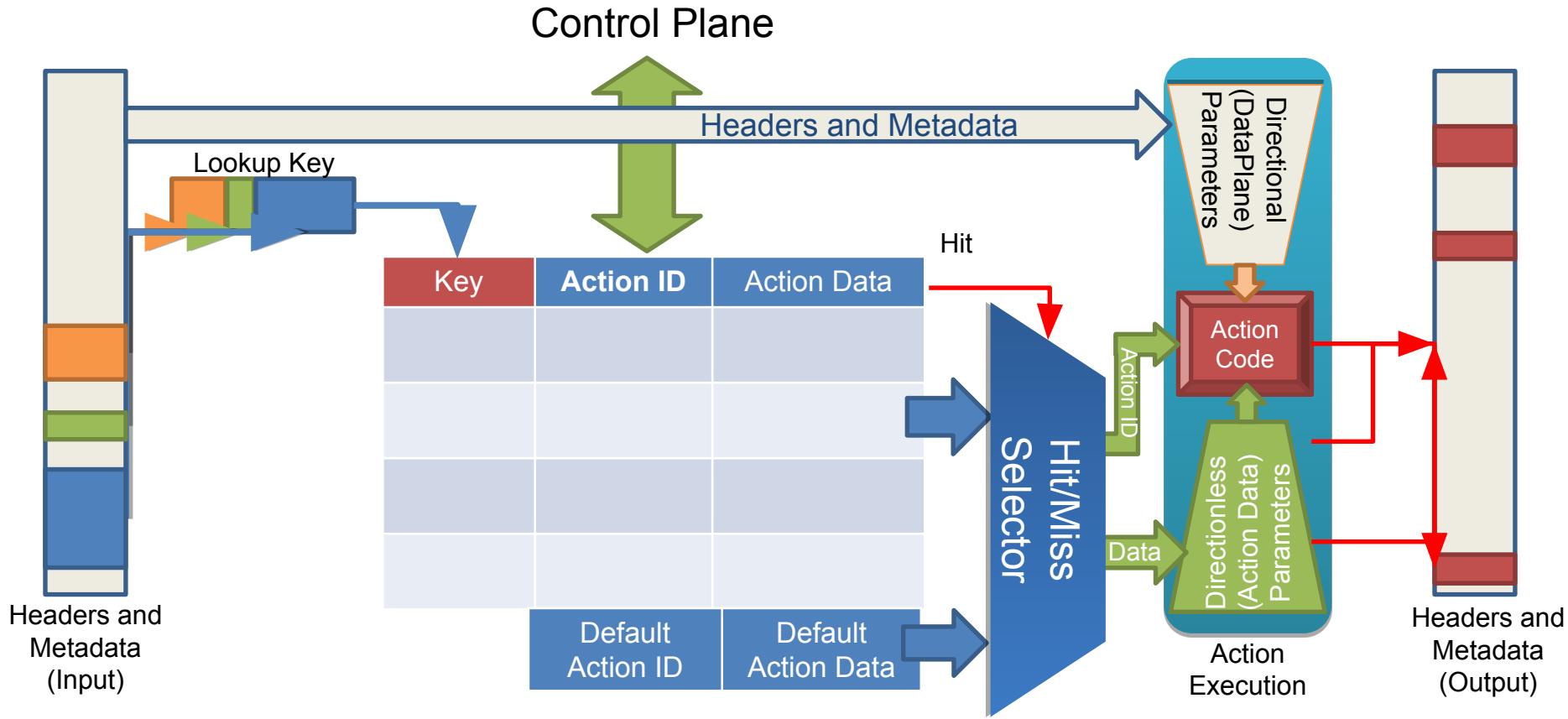
    apply {
        swap_mac(hdr.ethernet.srcAddr,
                  hdr.ethernet.dstAddr);
        std_meta.egress_spec = std_meta.ingress_port;
    }
}
```

- **Very similar to C functions**
- **Can be declared inside a control or globally**
- **Parameters have type and direction**
- **Variables can be instantiated inside**
- **Many standard arithmetic and logical operations are supported**
 - +, -, *
 - ~, &, |, ^, >>, <<
 - ==, !=, >, >=, <, <=
 - No division/modulo
- **Non-standard operations:**
 - Bit-slicing: [m:l] (works as l-value too)
 - Bit Concatenation: ++

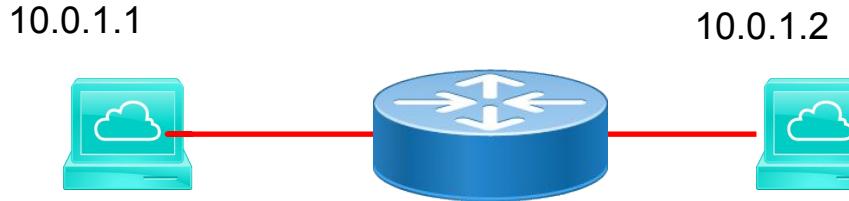
P4₁₆ Tables

- **The fundamental unit of a Match-Action Pipeline**
 - Specifies what data to match on and match kind
 - Specifies a list of *possible* actions
 - Optionally specifies a number of table **properties**
 - Size
 - Default action
 - Static entries
 - etc.
- **Each table contains one or more entries (rules)**
- **An entry contains:**
 - A specific key to match on
 - A **single** action that is executed when a packet matches the entry
 - Action data (possibly empty)

Tables: Match-Action Processing



Example: IPv4_LPM Table



Key	Action	Action Data
10.0.1.1/32	ipv4_forward	dstAddr=00:00:00:00:01:01 port=1
10.0.1.2/32	drop	
*	NoAction	

- **Data Plane (P4) Program**

- Defines the format of the table
 - Key Fields
 - Actions
 - Action Data
- Performs the lookup
- Executes the chosen action

- **Control Plane (IP stack, Routing protocols)**

- Populates table entries with specific information
 - Based on the configuration
 - Based on automatic discovery
 - Based on protocol calculations

IPv4_LPM Table

```
table ipv4_lpm {
    key = {
        hdr.ipv4.dstAddr: lpm;
    }
    actions = {
        ipv4_forward;
        drop;
        NoAction;
    }
    size = 1024;
    default_action = NoAction();
}
```

Match Kinds

```
/* core.p4 */
match_kind {
    exact,
    ternary,
    lpm
}

/* v1model.p4 */
match_kind {
    range,
    selector
}

/* Some other architecture */
match_kind {
    regexp,
    fuzzy
}
```

- **The type `match_kind` is special in P4**
- **The standard library (`core.p4`) defines three standard match kinds**
 - Exact match
 - Ternary match
 - LPM match
- **The architecture (`v1model.p4`) defines two additional match kinds:**
 - range
 - selector
- **Other architectures may define (and provide implementation for) additional match kinds**

Defining Actions for L3 forwarding

```
/* core.p4 */
action NoAction() {
}

/* basic.p4 */
action drop() {
    mark_to_drop();
}

/* basic.p4 */
action ipv4_forward(macAddr_t dstAddr,
                     bit<9> port) {
    ...
}
```

- Actions can have two different types of parameters
 - Directional (from the Data Plane)
 - Directionless (from the Control Plane)
- Actions that are called directly:
 - Only use directional parameters
- Actions used in tables:
 - Typically use directionless parameters
 - May sometimes use directional parameters too



Applying Tables in Controls

```
control MyIngress(inout headers hdr,
                  inout metadata meta,
                  inout standard_metadata_t standard_metadata) {
    table ipv4_lpm {
        ...
    }
    apply {
        ...
        ipv4_lpm.apply();
        ...
    }
}
```

P4₁₆ Deparsing

```
/* From core.p4 */
extern packet_out {
    void emit<T>(in T hdr);
}

/* User Program */
control DeparserImpl(packet_out packet,
                      in headers hdr) {
    apply {
        ...
        packet.emit(hdr.ethernet);
        ...
    }
}
```

- **Assembles the headers back into a well-formed packet**
- **Expressed as a control function**
 - No need for another construct!
- **packet_out extern is defined in core.p4:** emit(hdr): serializes header if it is valid
- **Advantages:**
 - Makes deparsing explicit...
...but decouples from parsing

Coding Break

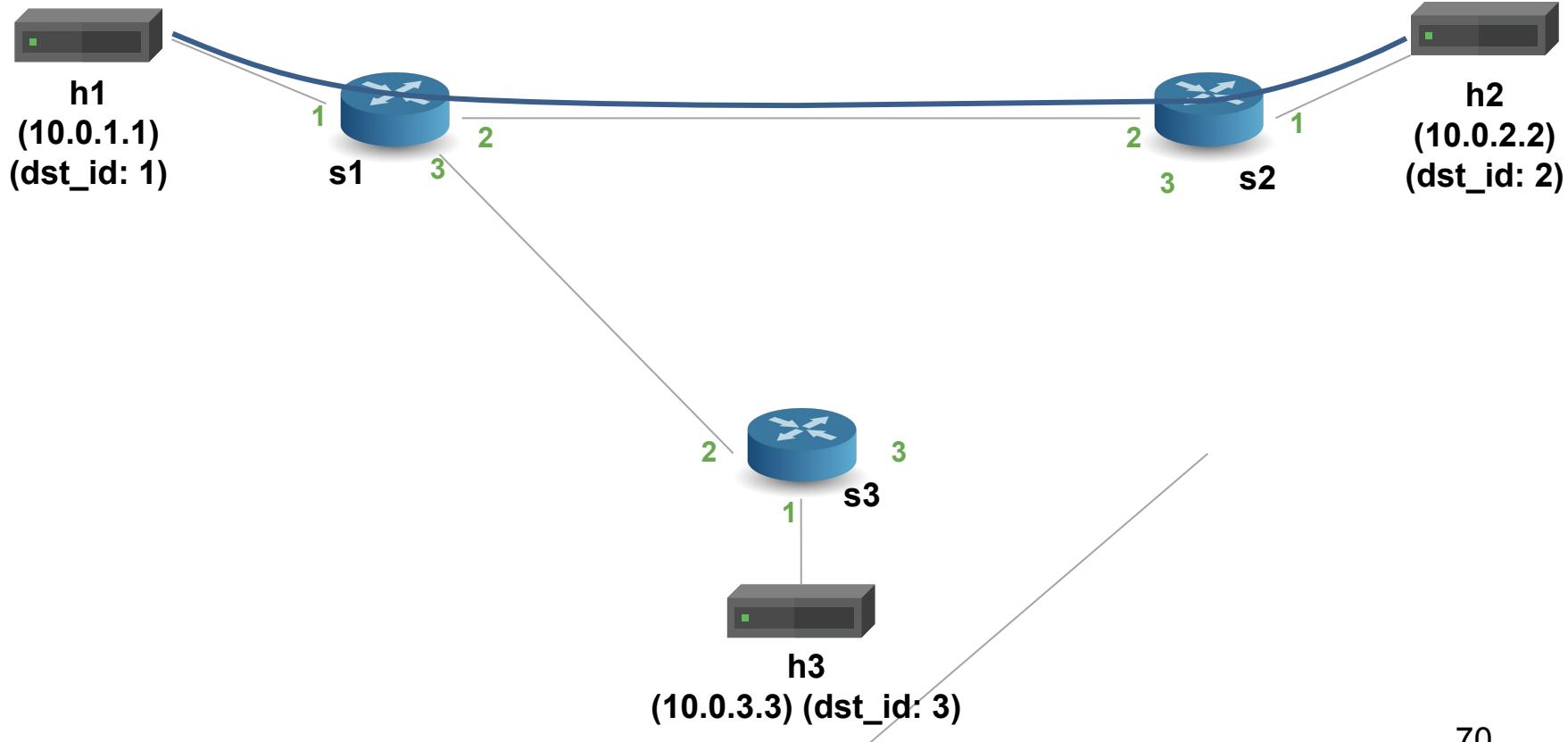
Basic Tunneling

- Add support for basic tunneling to the basic IP router
- Define a new header type (`myTunnel`) to encapsulate the IP packet
- `myTunnel` header includes:
 - `proto_id` : type of packet being encapsulated
 - `dst_id` : ID of destination host
- Modify the switch to perform routing using the `myTunnel` header

Basic Tunneling TODO List

- Define `myTunnel_t` header type and add to headers struct
- Update parser
- Define `myTunnel_forward` action
- Define `myTunnel_exact` table
- Update table application logic in `MyIngress apply` statement
- Update deparser
- Adding forwarding rules

Basic Forwarding: Topology



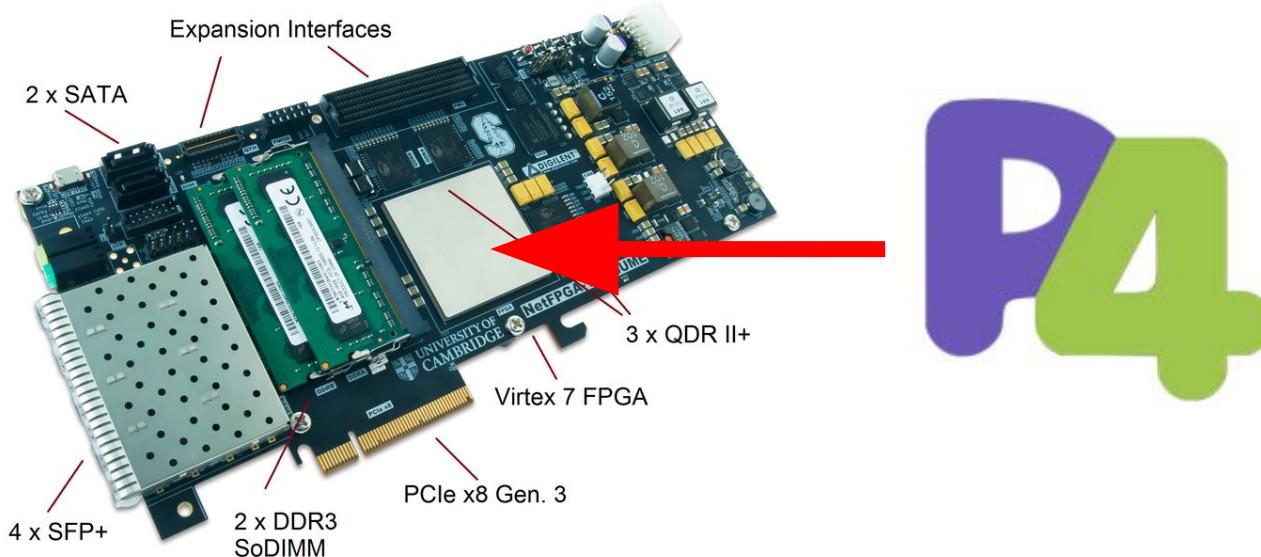
Coding Break

FAQs

- **Can I apply a table multiple times in my P4 Program?**
 - No (except via resubmit / recirculate)
- **Can I modify table entries from my P4 Program?**
 - No (except for direct counters)
- **What happens upon reaching the `reject` state of the parser?**
 - Architecture dependent
- **How much of the packet can I parse?**
 - Architecture dependent

P4→NetFPGA

- Prototype and evaluate P4 programs in real hardware!
- 4x10G network interfaces
- Special price for academic users :)
- <https://github.com/NetFPGA/P4-NetFPGA-public/wiki>



P4

Debugging

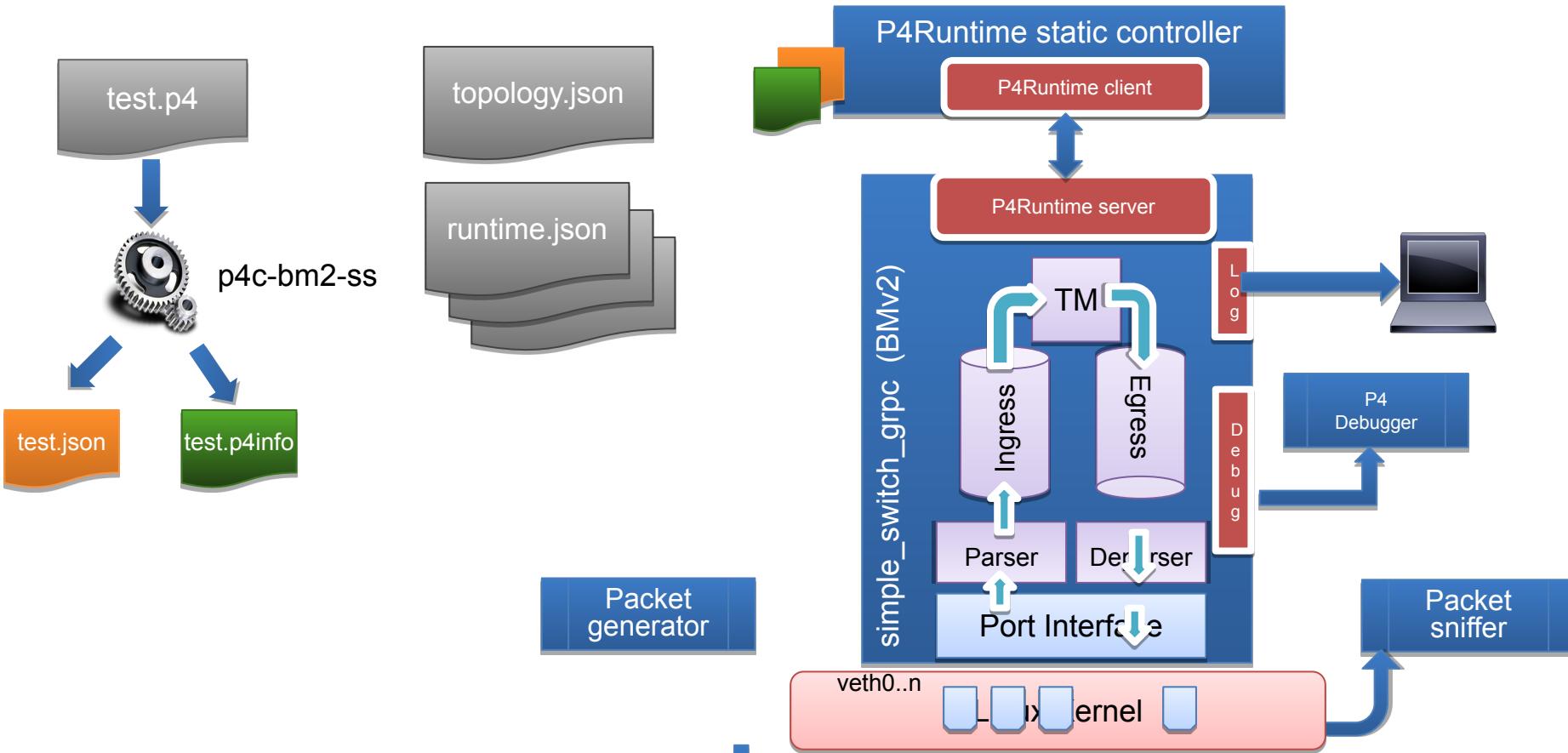
```
control MyIngress(...) {
    table debug {
        key = {
            std_meta.egress_spec : exact;
        }
        actions = { }
    }
    apply {
        ...
        debug.apply();
    }
}
```

- **Bmv2 maintains logs that keep track of how packets are processed in detail**
 - /tmp/p4s.s1.log
 - /tmp/p4s.s2.log
 - /tmp/p4s.s3.log
- **Can manually add information to the logs by using a dummy debug table that reads headers and metadata of interest**
 - [15:16:48.145] [bmv2] [D]
[thread 4090] [96.0] [cxt 0]
Looking up key:
* std_meta.egress_spec : 2

Lab 2: P4Runtime

P4 Software Tools

Makefile: under the hood



Makefile: under the hood (in pseudocode)

```
P4C_ARGS = --p4runtime-file $(basename $@).p4info  
          --p4runtime-format text  
RUN_SCRIPT = ../../utils/run_exercise.py  
TOPO = topology.json
```

dirs:

```
mkdir -p build pcaps logs
```

build: for each P4 program, generate BMv2 json file

```
p4c-bm2-ss --p4v 16 $(P4C_ARGS) -o $@ $<
```

run: build, then [default target]

```
sudo python $(RUN_SCRIPT) -t $(TOPO)
```

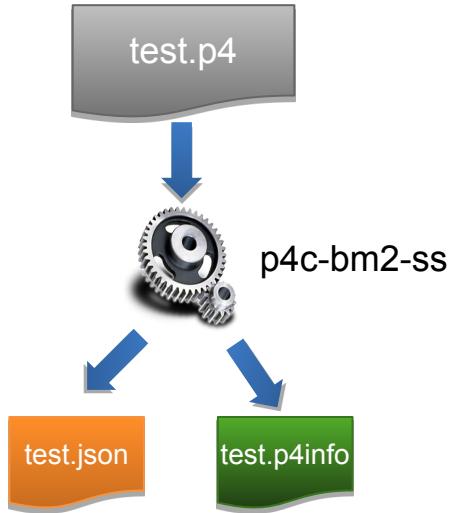
stop: sudo mn -c

clean: stop, then

```
rm -f *.pcap
```

```
rm -rf build pcaps logs
```

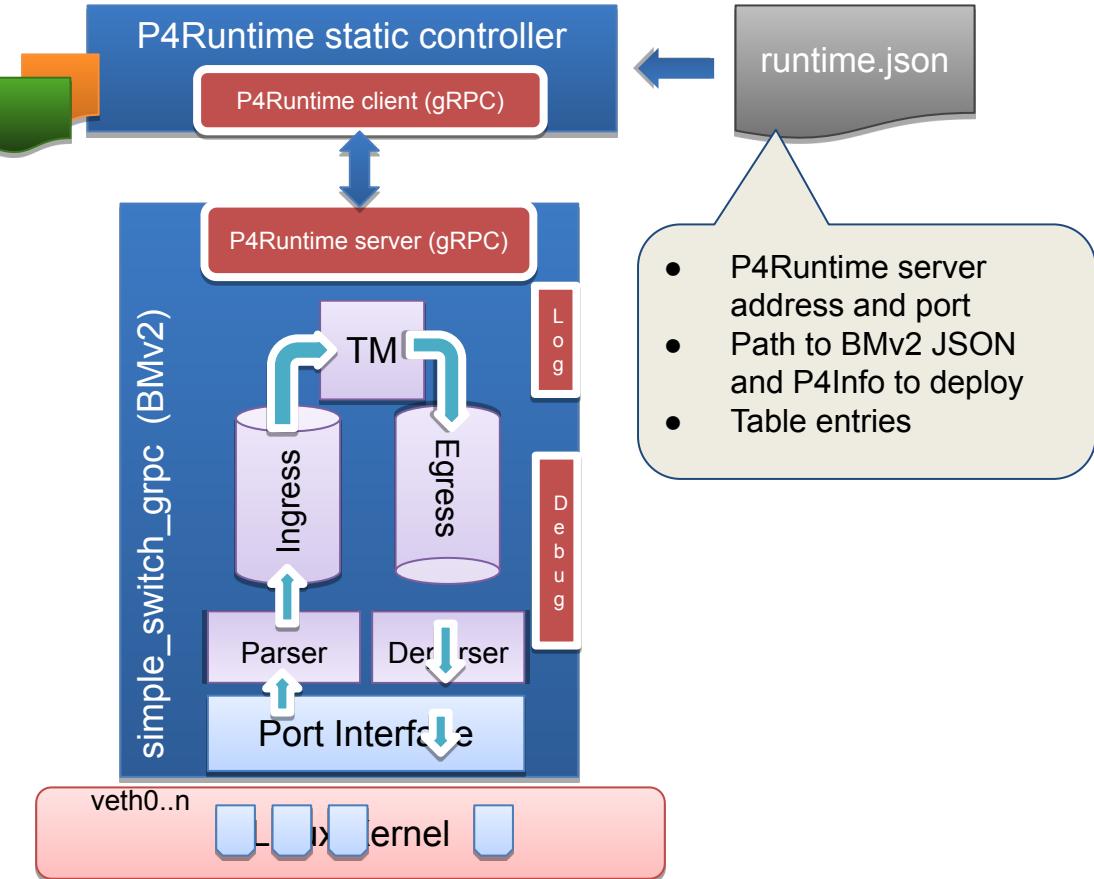
Step 1: P4 Program compilation [build phase]



```
$ p4c-bm2-ss --p4v 16 \
-o test.json \
--p4runtime-file test.p4info \
--p4runtime-format text \
test.p4
```

Step 2: run_exercise.py [run phase]

- Create network based on topology.json
- Start simple_switch_grpc instance for each switch
- Use P4Runtime to push the P4 program (P4Info and BMv2 JSON)
- Add the static rules defined in runtime.json



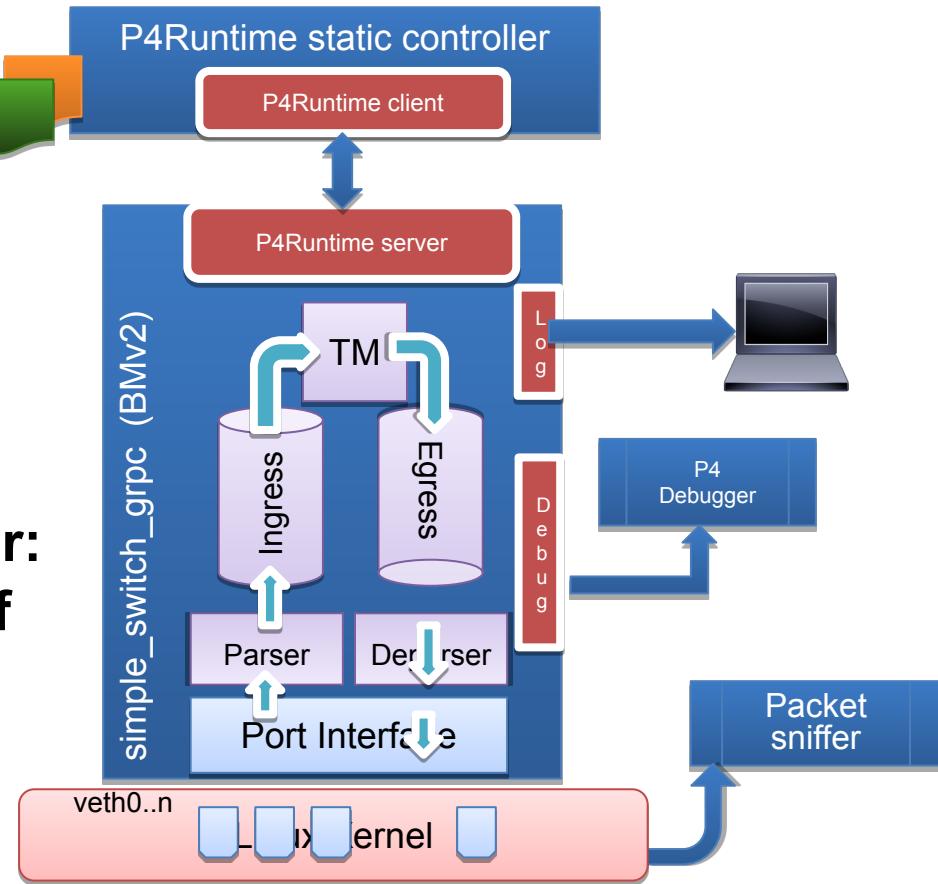
Step 3: Run the traffic generator and sniffer

In some exercises, this is
send.py and receive.py

In others, we use standard
Linux programs, like ping

In the p4runtime exercise,
we also run our own controller:
mycontroller.py instead of
the static one

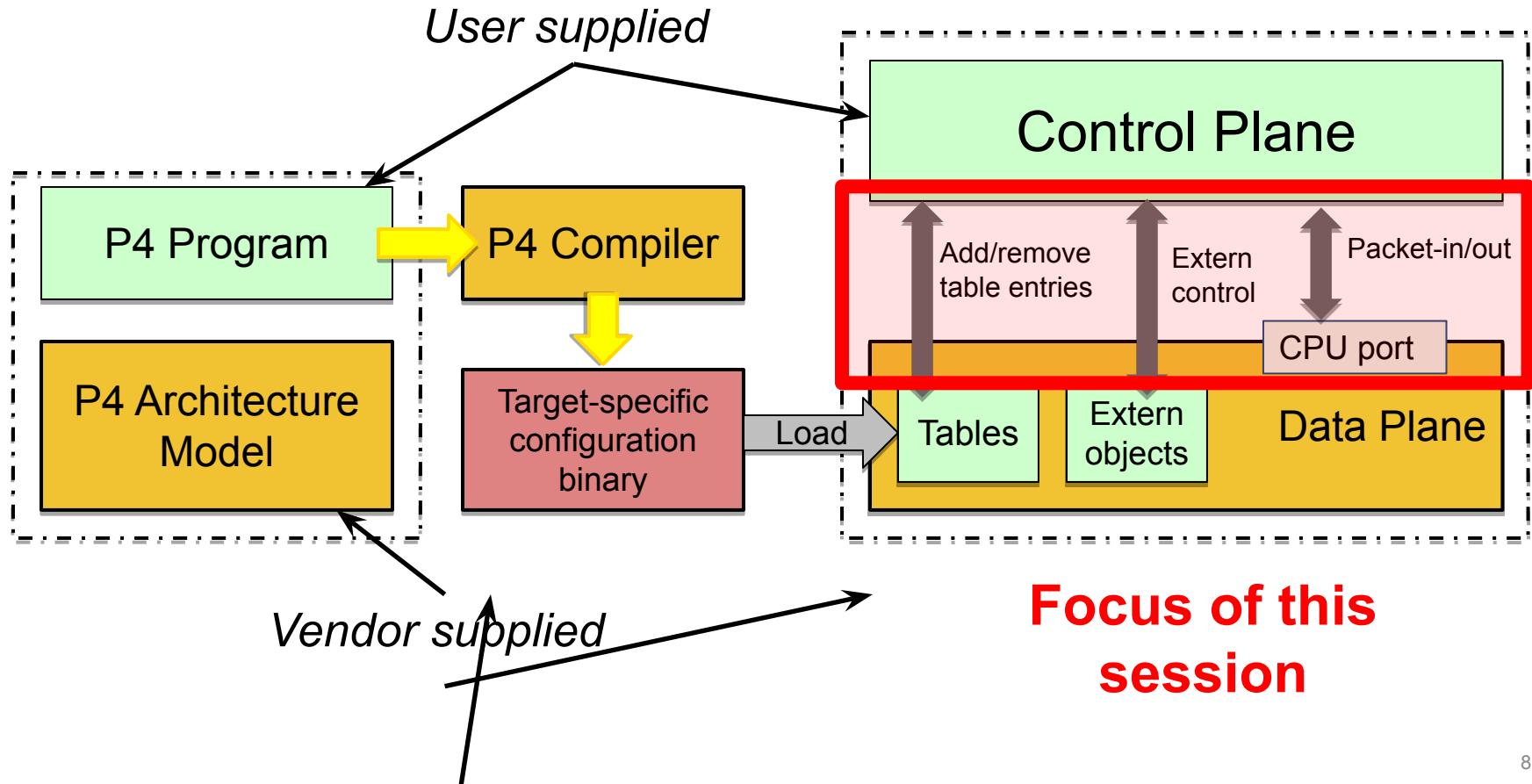
Packet
generator



P4Runtime

- API overview
- Workflow
- Exercise - Tunneling
- Why P4Runtime?

Runtime control of P4 data planes



Existing approaches to runtime control

- **P4 compiler auto-generated runtime APIs**
 - Program-dependent -- hard to provision new P4 program without restarting the control plane!
- **BMv2 CLI**
 - Program-independent, but target-specific -- control plane not portable!
- **OpenFlow**
 - Target-independent, but protocol-dependent -- protocol headers and actions baked in the specification!
- **OCP Switch Abstraction Interface (SAI)**
 - Target-independent, but protocol-dependent

Why do we need another data plane control API?

HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)

SITUATION:
THERE ARE
14 COMPETING
STANDARDS.

14?! RIDICULOUS!
WE NEED TO DEVELOP
ONE UNIVERSAL STANDARD
THAT COVERS EVERYONE'S
USE CASES.

YEAH!



SOON:

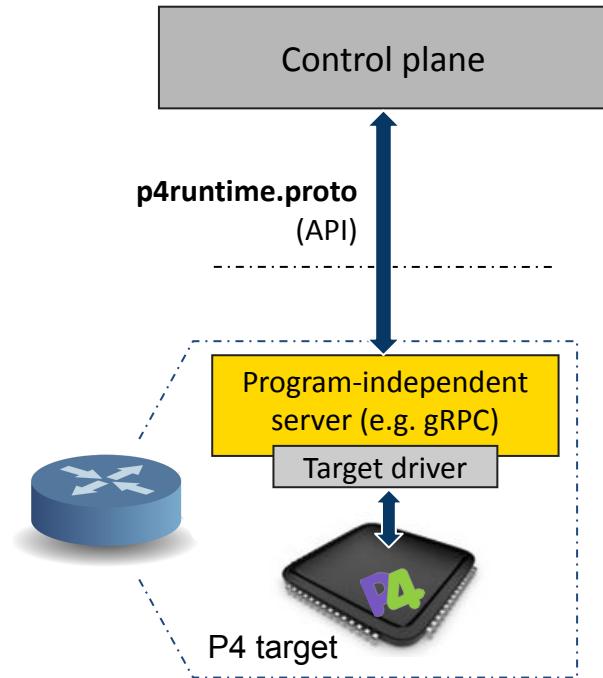
SITUATION:
THERE ARE
15 COMPETING
STANDARDS.

Properties of a runtime control API

API	Target-independent	Protocol-independent
P4 compiler auto-generated	✓	✗
BMv2 CLI	✗	✓
OpenFlow	✓	✗
SAI	✓	✗
P4Runtime	✓	✓

What is P4Runtime?

- **Framework for runtime control of P4 targets**
 - Open-source API + server implementation
 - <https://github.com/p4lang/PI>
 - Initial contribution by Google and Barefoot
- **Work-in-progress by the p4.org API WG**
 - Draft of version 1.0 available
- **Protobuf-based API definition**
 - p4runtime.proto
 - gRPC transport
- **P4 program-independent**
 - API doesn't change with the P4 program
- **Enables field-reconfigurability**
 - Ability to push new P4 program without recompiling the software stack of target switches



Protocol Buffers Basics

- Language for describing data for serialization in a structured way
- Common binary wire-format
- Language-neutral
 - Code generators for: *Action Script, C, C++, C#, Clojure, Lisp, D, Dart, Erlang, Go, Haskell, Java, Javascript, Lua, Objective C, OCaml, Perl, PHP, Python, Ruby, Rust, Scala, Swift, Visual Basic, ...*
- Platform-neutral
- Extensible and backwards compatible
- Strongly typed

```
syntax = "proto3";

message Person {
    string name = 1;
    int32 id = 2;
    string email = 3;

    enum PhoneType {
        MOBILE = 0;
        HOME = 1;
        WORK = 2;
    }

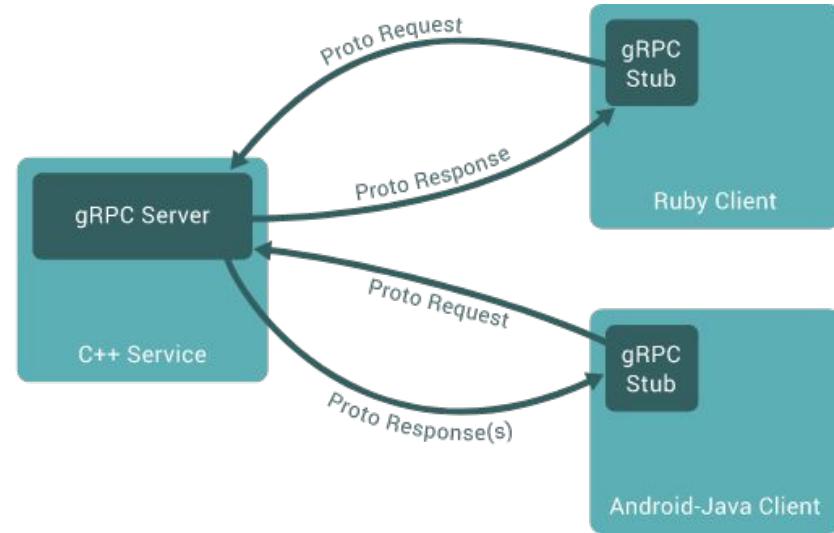
    message PhoneNumber {
        string number = 1;
        PhoneType type = 2;
    }

    repeated PhoneNumber phone = 4;
}
```

gRPC Basics

- Use Protocol Buffers to define service API and messages
- Automatically generate native stubs in:

- C / C++
- C#
- Dart
- Go
- Java
- Node.js
- PHP
- Python
- Ruby



- Transport over HTTP/2.0 and TLS
 - Efficient single TCP connection implementation that supports bidirectional streaming

gRPC Service Example

```
// The greeter service definition.  
service Greeter {  
    // Sends a greeting  
    rpc SayHello (HelloRequest) returns (HelloReply) {}  
}  
  
// The request message containing the user's name.  
message HelloRequest {  
    string name = 1;  
}  
  
// The response message containing the greetings  
message HelloReply {  
    string message = 1;  
}
```

More details here: <https://grpc.io/docs/guides/>

P4Runtime Service

Enables a local or remote entity to load the pipeline/program, send/receive packets, and read and write forwarding table entries, counters, and other chip features.

```
service P4Runtime {
    rpc Write(WriteRequest) returns (WriteResponse) {}
    rpc Read(ReadRequest) returns (stream ReadResponse) {}
    rpc SetForwardingPipelineConfig(SetForwardingPipelineConfigRequest)
        returns (SetForwardingPipelineConfigResponse) {}
    rpc GetForwardingPipelineConfig(GetForwardingPipelineConfigRequest)
        returns (GetForwardingPipelineConfigResponse) {}
    rpc StreamChannel(stream StreamMessageRequest)
        returns (stream StreamMessageResponse) {}
}
```

P4Runtime Service

Protobuf Definition:

<https://github.com/p4lang/PI/blob/master/proto/p4/v1/p4runtime.proto>

Service Specification:

Working draft of version 1.0 is available now

<https://p4.org/p4-spec/docs/P4Runtime-v1.0.0.pdf>

P4Runtime Write Request

```
message WriteRequest {  
    uint64 device_id = 1;  
    uint64 role_id = 2;  
    Uint128 election_id = 3;  
    repeated Update updates = 4;  
}
```

```
message Update {  
    enum Type {  
        UNSPECIFIED = 0;  
        INSERT = 1;  
        MODIFY = 2;  
        DELETE = 3;  
    }  
    Type type = 1;  
    Entity entity = 2;
```

```
message Entity {  
    oneof entity {  
        ExternEntry extern_entry = 1;  
        TableEntry table_entry = 2;  
        ActionProfileMember  
            action_profile_member = 3;  
        ActionProfileGroup  
            action_profile_group = 4;  
        MeterEntry meter_entry = 5;  
        DirectMeterEntry direct_meter_entry = 6;  
        CounterEntry counter_entry = 7;  
        DirectCounterEntry direct_counter_entry = 8;  
        PacketReplicationEngineEntry  
            packet_replication_engine_entry = 9;  
        ValueSetEntry value_set_entry = 10;  
        RegisterEntry register_entry = 11;  
    }  
}
```

P4Runtime Table Entry

p4runtime.proto simplified excerpts:

```
message TableEntry {  
    uint32 table_id;  
    repeated FieldMatch match;  
    Action action;  
    int32 priority;  
    ...  
}
```

```
message Action {  
    uint32 action_id;  
    message Param {  
        uint32 param_id;  
        bytes value;  
    }  
    repeated Param params;  
}
```

```
message FieldMatch {  
    uint32 field_id;  
    message Exact {  
        bytes value;  
    }  
    message Ternary {  
        bytes value;  
        bytes mask;  
    }  
    ...  
    oneof field_match_type {  
        Exact exact;  
        Ternary ternary;  
        ...  
    }  
}
```

Full protobuf definition:

<https://github.com/p4lang/PI/blob/master/proto/p4/p4runtime.proto>

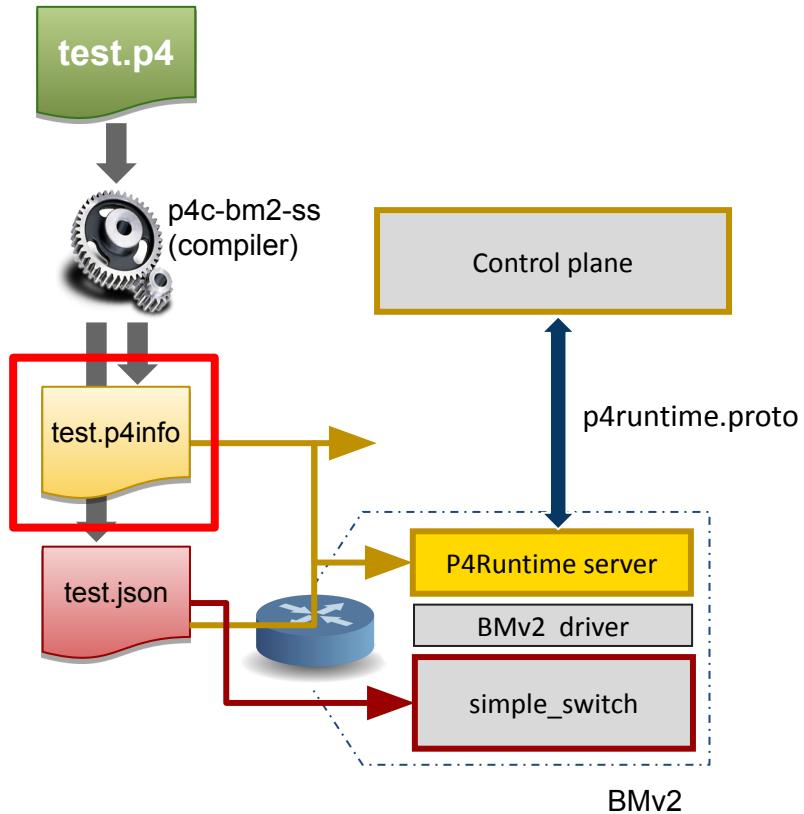
To add a table entry, the control plane needs to know:

- **IDs of P4 entities**
 - Tables, field matches, actions, params, etc.
- **Field matches for the particular table**
 - Match type, bitwidth, etc.
- **Parameters for the particular action**
- **Other P4 program attributes**

P4Runtime workflow

P4Info

- **Captures P4 program attributes needed at runtime**
 - IDs for tables, actions, params, etc.
 - Table structure, action parameters, etc.
- **Protobuf-based format**
- **Target-independent compiler output**
 - Same P4Info for BMv2, ASIC, etc.



Full P4Info protobuf specification:

<https://github.com/p4lang/PI/blob/master/proto/p4/config/v1/p4info.proto>

P4Info example

basic_router.p4

```
...
action ipv4_forward(bit<48> dstAddr,
                    bit<9> port) {
    /* Action implementation */
}

...
table ipv4_lpm {
    key = {
        hdr.ipv4.dstAddr: lpm;
    }
    actions = {
        ipv4_forward;
        ...
    }
    ...
}
```

basic_router.p4info

```
actions {
    id: 16786453
    name: "ipv4_forward"
    params {
        id: 1
        name: "dstAddr"
        bitwidth: 48
        ...
        id: 2
        name: "port"
        bitwidth: 9
    }
}
...
tables {
    id: 33581985
    name: "ipv4_lpm"
    match_fields {
        id: 1
        name: "hdr.ipv4.dstAddr"
        bitwidth: 32
        match_type: LPM
    }
    action_ref_id: 16786453
}
```



P4 compiler

P4Runtime Table Entry Example

basic_router.p4

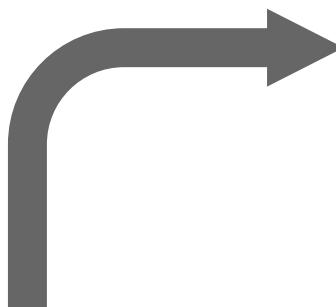
```
action ipv4_forward(bit<48> dstAddr,  
                    bit<9> port) {  
    /* Action implementation */  
}  
  
table ipv4_lpm {  
    key = {  
        hdr.ipv4.dstAddr: lpm;  
    }  
    actions = {  
        ipv4_forward;  
        ...  
    }  
    ...  
}
```



Logical view of table entry

```
hdr.ipv4.dstAddr=10.0.1.1/32  
-> ipv4_forward(00:00:00:00:00:10, 7)
```

Control plane
generates

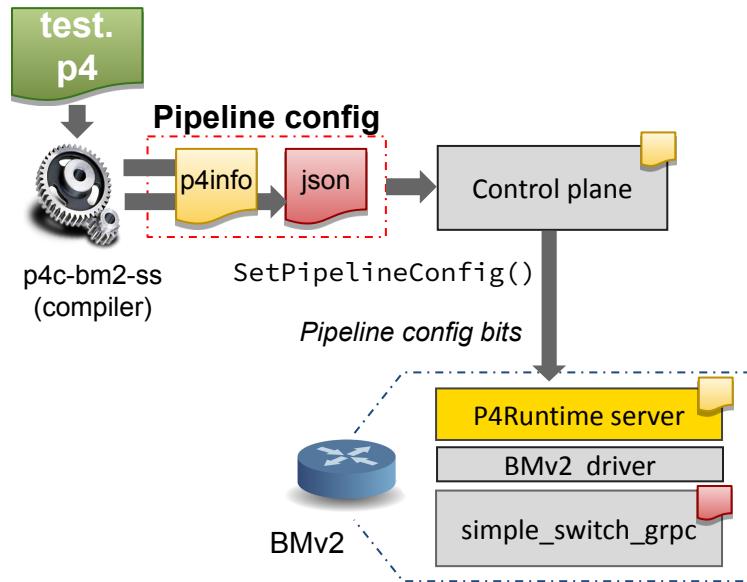


Protobuf message

```
table_entry {  
    table_id: 33581985  
    match {  
        field_id: 1  
        lpm {  
            value: "\n\000\001\001"  
            prefix_len: 32  
        }  
    }  
    action {  
        action_id: 16786453  
        params {  
            param_id: 1  
            value: "\000\000\000\000\000\n"  
        }  
        params {  
            param_id: 2  
            value: "\000\007"  
        }  
    }  
}
```

P4Runtime SetPipelineConfig

```
message SetForwardingPipelineConfigRequest {  
    enum Action {  
        UNSPECIFIED = 0;  
        VERIFY = 1;  
        VERIFY_AND_SAVE = 2;  
        VERIFY_AND_COMMIT = 3;  
        COMMIT = 4;  
        RECONCILE_AND_COMMIT = 5;  
    }  
    uint64 device_id = 1;  
    uint64 role_id = 2;  
    Uint128 election_id = 3;  
    Action action = 4;  
    ForwardingPipelineConfig config = 5;  
}
```



```
message ForwardingPipelineConfig {  
    config.P4Info p4info = 1;  
    // Target-specific P4 configuration.  
    bytes p4_device_config = 2;  
}
```

P4Runtime StreamChannel

```
message StreamMessageRequest {  
    oneof update {  
        MasterArbitrationUpdate  
            arbitration = 1;  
        PacketOut packet = 2;     
    }  
}
```

```
// Packet sent from the controller to the switch.  
message PacketOut {  
    bytes payload = 1;  
    // This will be based on P4 header annotated as  
    // @controller_header("packet_out").  
    // At most one P4 header can have this annotation.  
    repeated PacketMetadata metadata = 2;  
}
```

```
message StreamMessageResponse {  
    oneof update {  
        MasterArbitrationUpdate  
            arbitration = 1;  
        PacketIn packet = 2;     
    }  
}
```

```
// Packet sent from the switch to the controller.  
message PacketIn {  
    bytes payload = 1;  
    // This will be based on P4 header annotated as  
    // @controller_header("packet_in").  
    // At most one P4 header can have this annotation.  
    repeated PacketMetadata metadata = 2;  
}
```

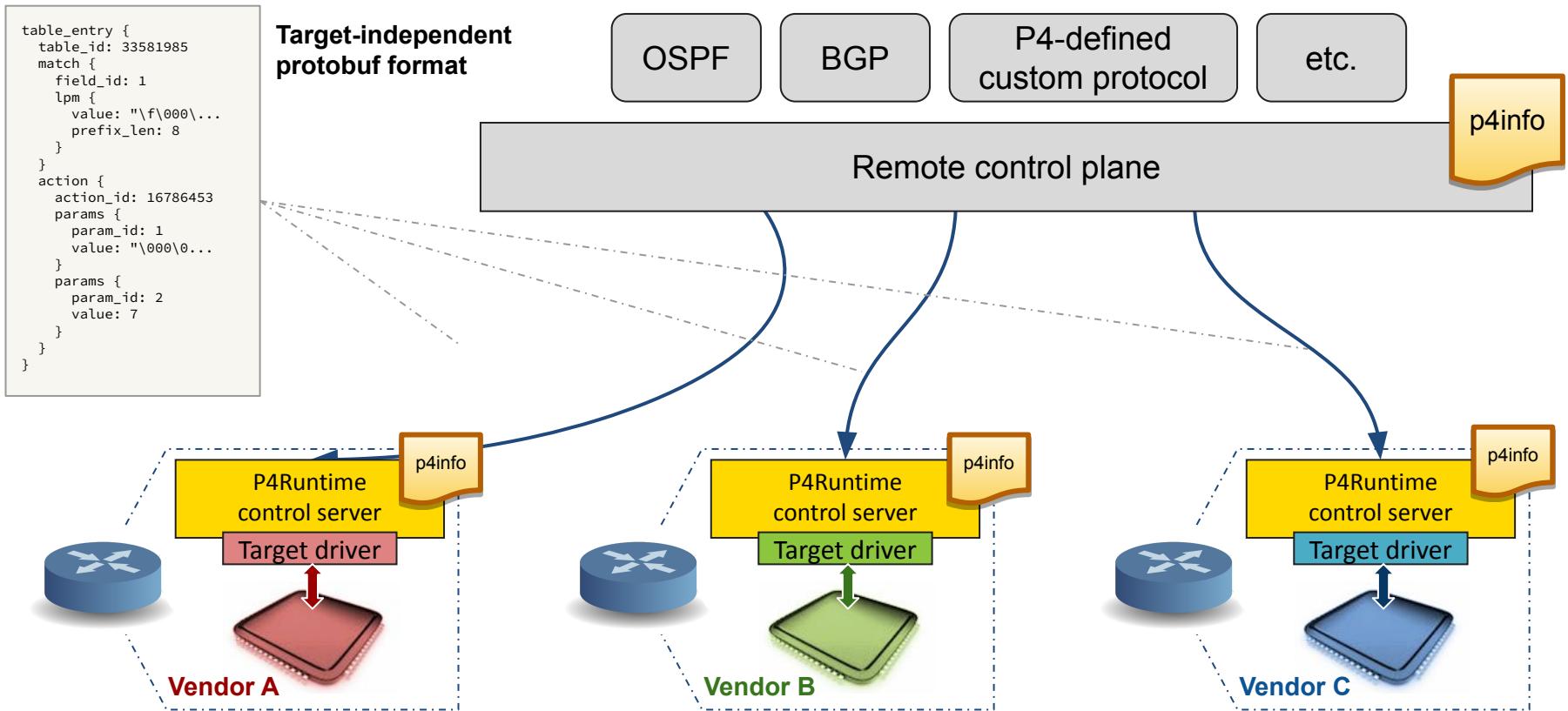
P4Runtime Common Parameters

- **device_id**
 - Specifies the specific forwarding chip or software bridge
 - **Set to 0 for single chip platforms**
- **role_id**
 - Corresponds to a role with specific capabilities (i.e. what operations, P4 entities, behaviors, etc. are in the scope of a given role)
 - Role definition is currently agreed upon between control and data planes offline
 - **Default role_id (0) has full pipeline access**
- **election_id**
 - P4Runtime supports mastership on a per-role basis
 - Client with the highest election ID is referred to as the "master", while all other clients are referred to as "slaves"
 - **Set to 0 for single instance controllers**

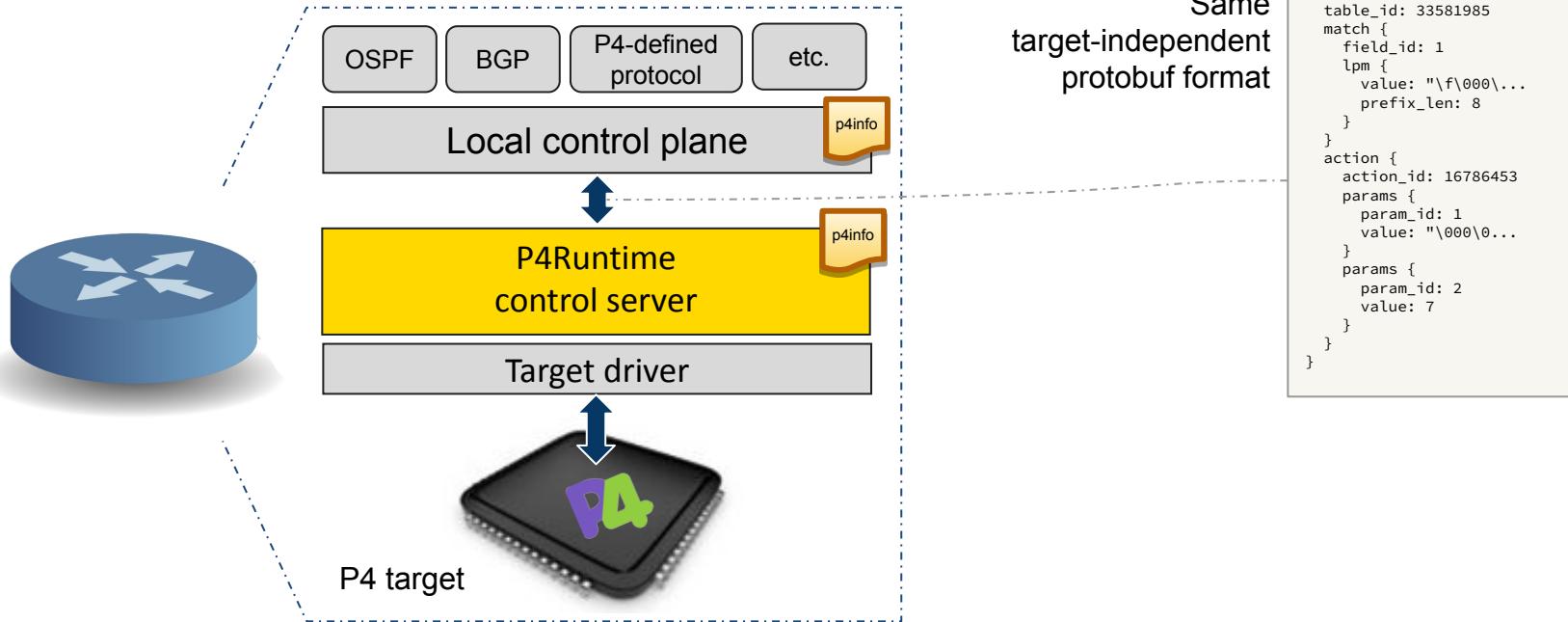
Mastership Arbitration

- Upon connecting to the device, the client (e.g. controller) needs to open a StreamChannel
- The client must advertise its role_id and election_id using a MasterArbitrationUpdate message
 - If role_id is not set, it implies the default role and will be granted full pipeline access
 - The election_id is opaque to the server and determined by the control plane (can be omitted for single-instance control plane)
- The switch marks the client for each role with the highest election_id as master
- Master can:
 - Perform Write requests
 - Receive PacketIn messages
 - Send PacketOut messages

Remote control



Local control



Same
target-independent
protobuf format

```
table_entry {  
    table_id: 33581985  
    match {  
        field_id: 1  
        lpm {  
            value: "\f\000\..."  
            prefix_len: 8  
        }  
    }  
    action {  
        action_id: 16786453  
        params {  
            param_id: 1  
            value: "\000\0..."  
        }  
        params {  
            param_id: 2  
            value: 7  
        }  
    }  
}
```

P4Runtime can be used equally well
by a remote or local control plane

P4Runtime API recap

Things we covered:

- **P4Runtime definition**
- **P4Info**
- **Table entries**
- **Set pipeline config**
- **Packet-in/out support**
- **Controller replication**
 - Via master-slave arbitration

What we didn't cover:

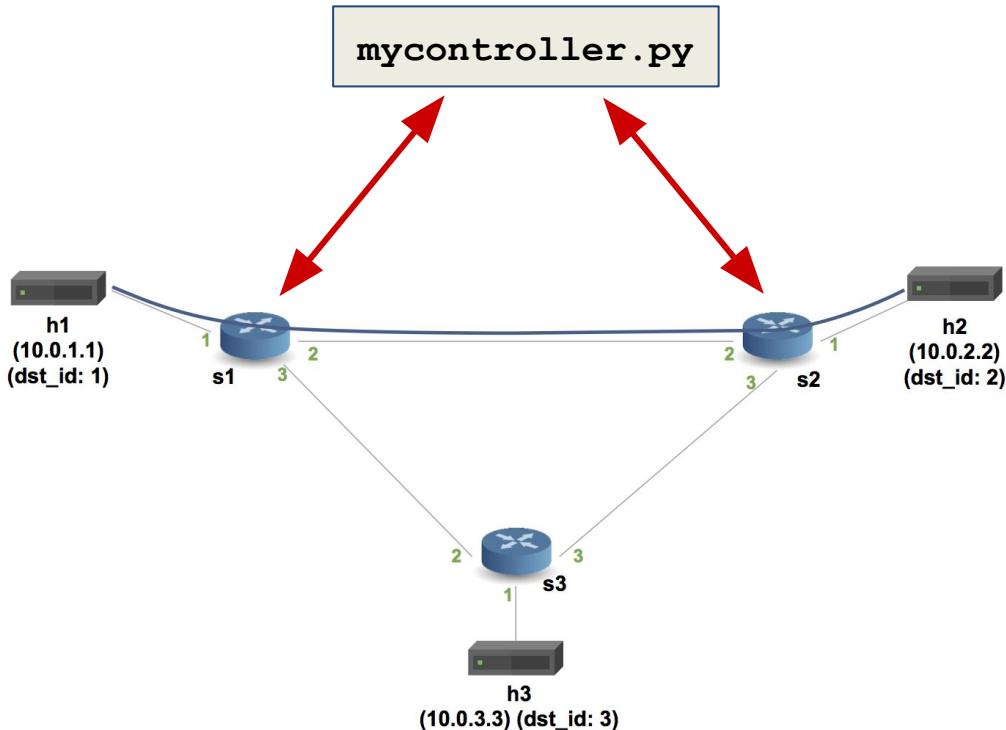
- **How to control other P4 entities**
 - Externs, counters, meters
- **Batched reads/writes**
- **Switch configuration**
 - Outside the P4Runtime scope
 - Achieved with other mechanisms
 - e.g., OpenConfig and gNMI

P4Runtime exercise

Exercise Overview

Controller's responsibilities:

1. Establish a gRPC connection to the switches for the P4Runtime service
2. Push the P4 program to each switch
3. Write the tunnel forwarding rules:
 - a. **myTunnel_ingress** rule to encapsulate packets on the ingress switch
 - b. **myTunnel_forward** rule to forward packets on the ingress switch
 - c. **myTunnel_egress** rule to decapsulate and forward packets on the egress switch
4. Read the tunnel ingress and egress counters every 2 seconds



Getting started

The source code has already been downloaded on your VM:

`~/tutorials/exercises/p4runtime`

You should start by reading the `README.md`

In this exercise, you will need to complete the implementation of `writeTunnelRules` in `mycontroller.py`

You will need two Terminal windows: one for your dataplane network (Mininet) that you will start using `make`, and the other is for your controller program.

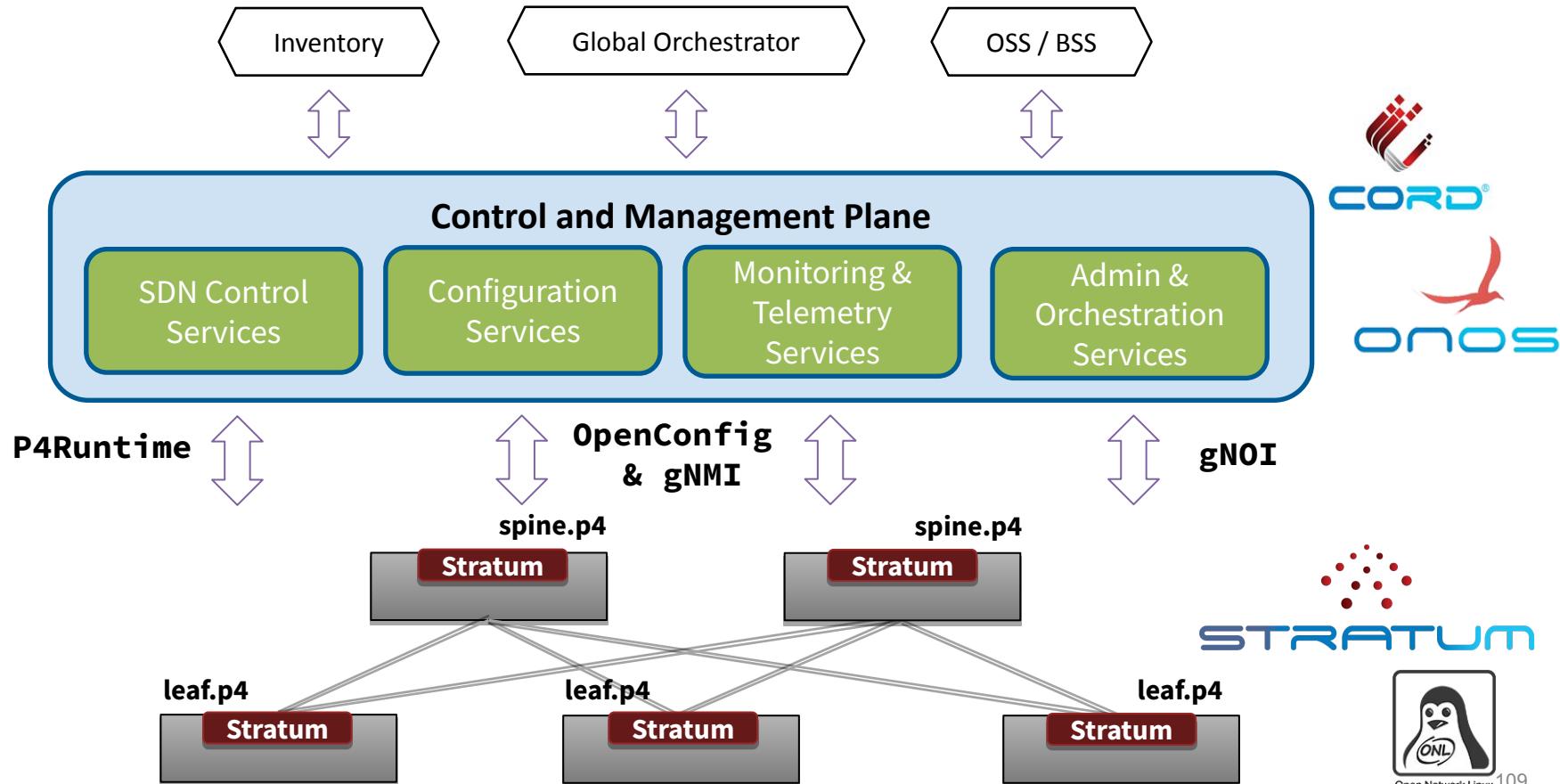
To find the source code:

<https://github.com/p4lang/tutorials/>

The screenshot shows a portion of a `README.md` file. At the top, there's a header for "Implementing a Control Plane using P4 Runtime". Below it is a section titled "Introduction". A note states: "In this exercise, we will be using P4 Runtime to send flow entries to the switch instead of using the switch's CLI. We will be building on the same P4 program that you used in the `basic_tunnel` exercise. The P4 program has been renamed to `advanced_tunnel.py` and has been augmented with two counters (`ingressTunnelCounter`, `egressTunnelCounter`) and two new actions (`myTunnel_ingress`, `myTunnel_egress`). You will use the starter program, `mycontroller.py`, and a few helper libraries in the `p4runtime_lib` directory to create the table entries necessary to tunnel traffic between host 1 and 2." A "Spoiler alert" note says: "There is a reference solution in the `solution` sub-directory. Feel free to compare your implementation to the reference." Below this is a "Step 1: Run the (incomplete) starter code" section. It says: "The starter code for this assignment is in a file called `mycontroller.py`, and it will install only some of the rules that you need to tunnel traffic between two hosts." It also says: "Let's first compile the new P4 program, start the network, use `mycontroller.py` to install a few rules, and look at the `ingressTunnelCounter` to see that things are working as expected." At the bottom, there's a step: "1. In your shell, run: `make`".

P4Runtime in the wild

P4Runtime-enabled Open Source SDN Stack



Some Pointers

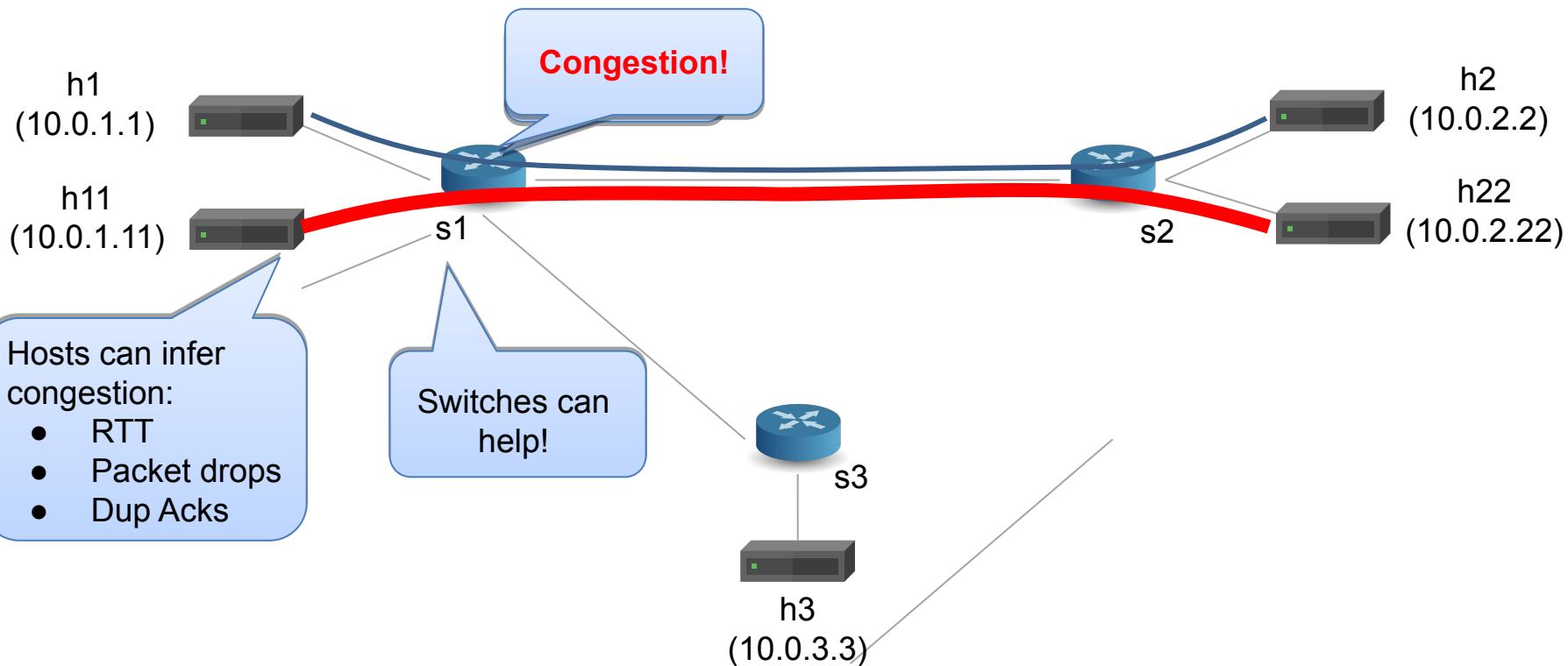
<https://stratumproject.org/>

<https://onosproject.org/>

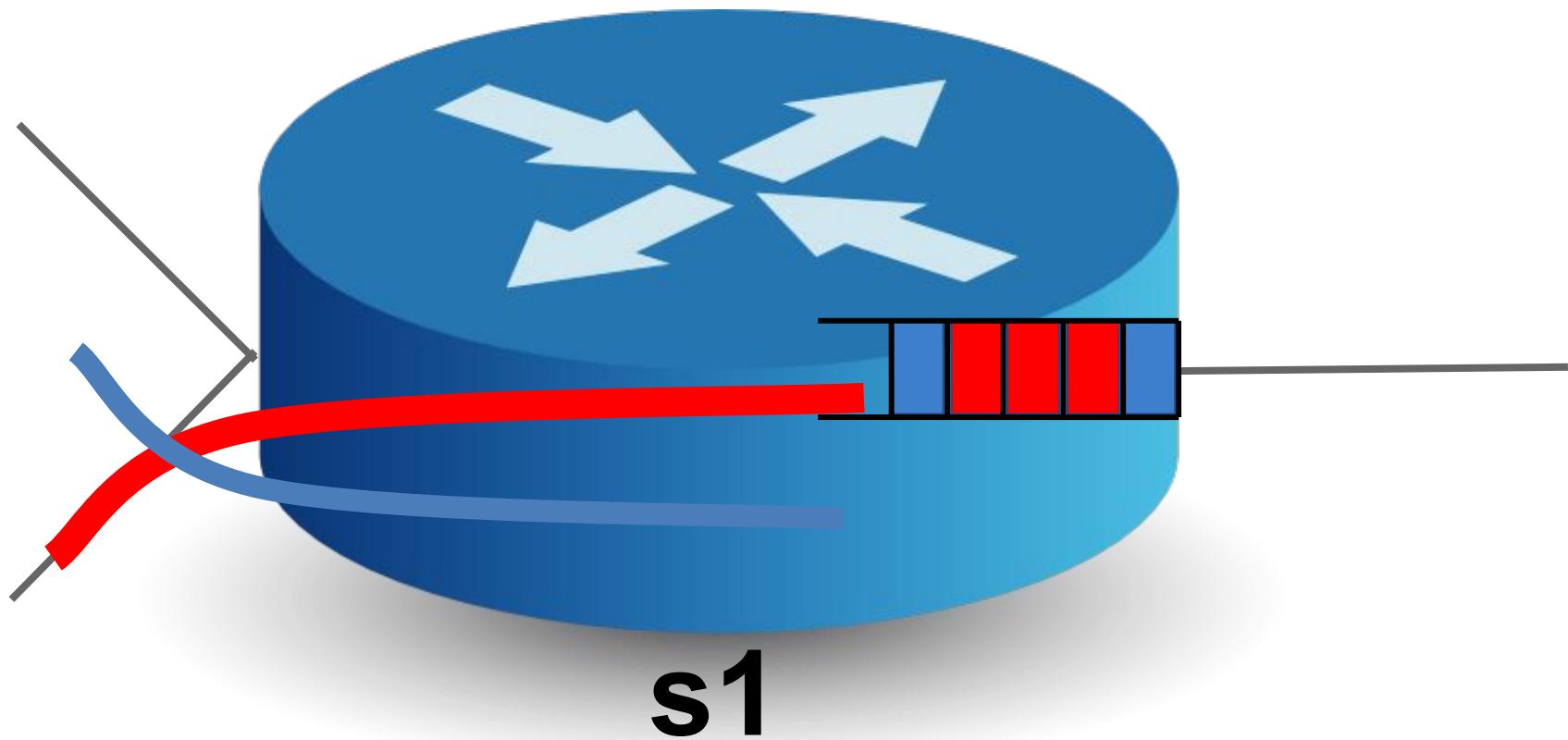
<https://wiki.onosproject.org/display/ONOS/P4+brigade>

Lab 3: Monitoring & Debugging

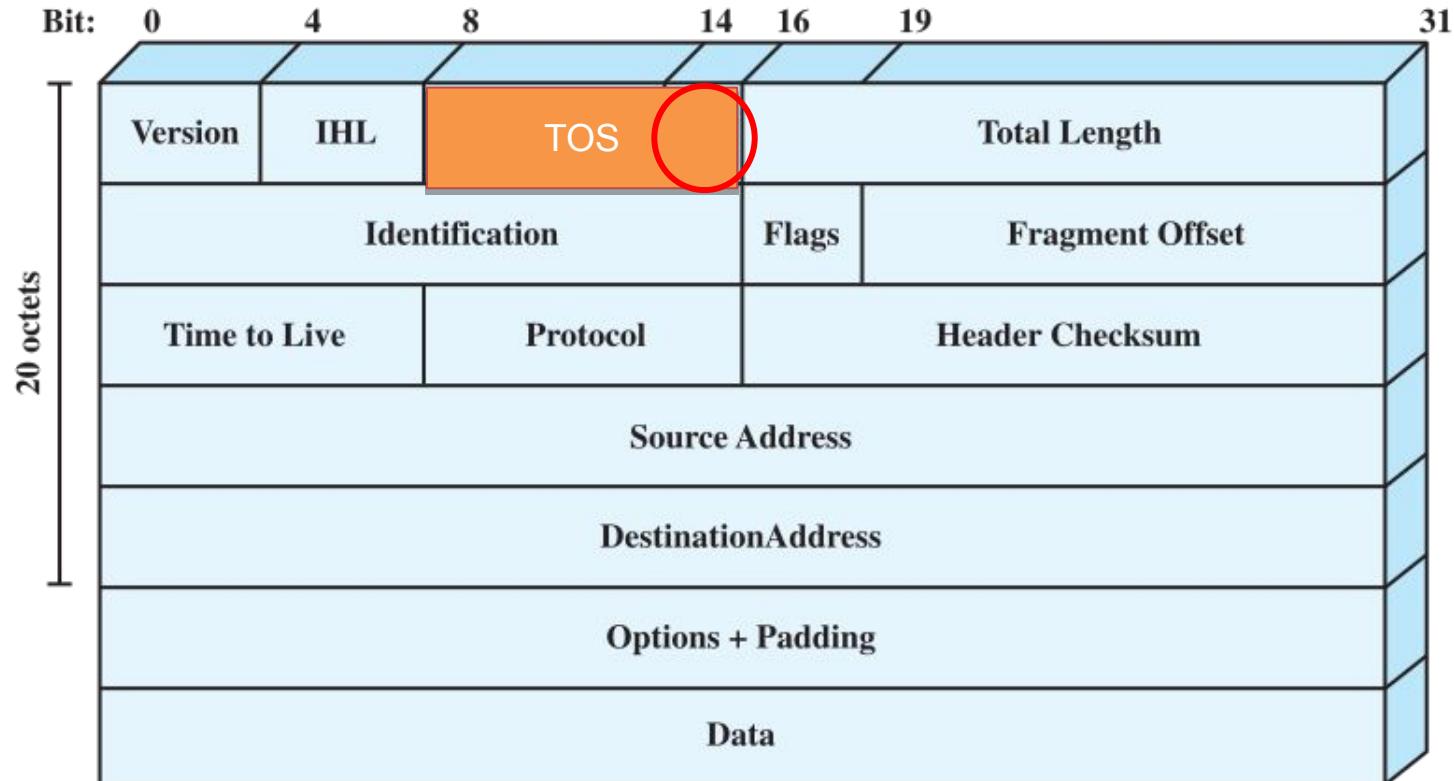
Monitoring & Debugging



Monitoring & Debugging



Explicit Congestion Notification



Explicit Congestion Notification

- **Explicit Congestion Notification**

- 00: Non ECN-Capable Transport, Non-ECT
- 10: ECN Capable Transport, ECT(0)
- 01: ECN Capable Transport, ECT(1)
- 11: Congestion Encountered, CE

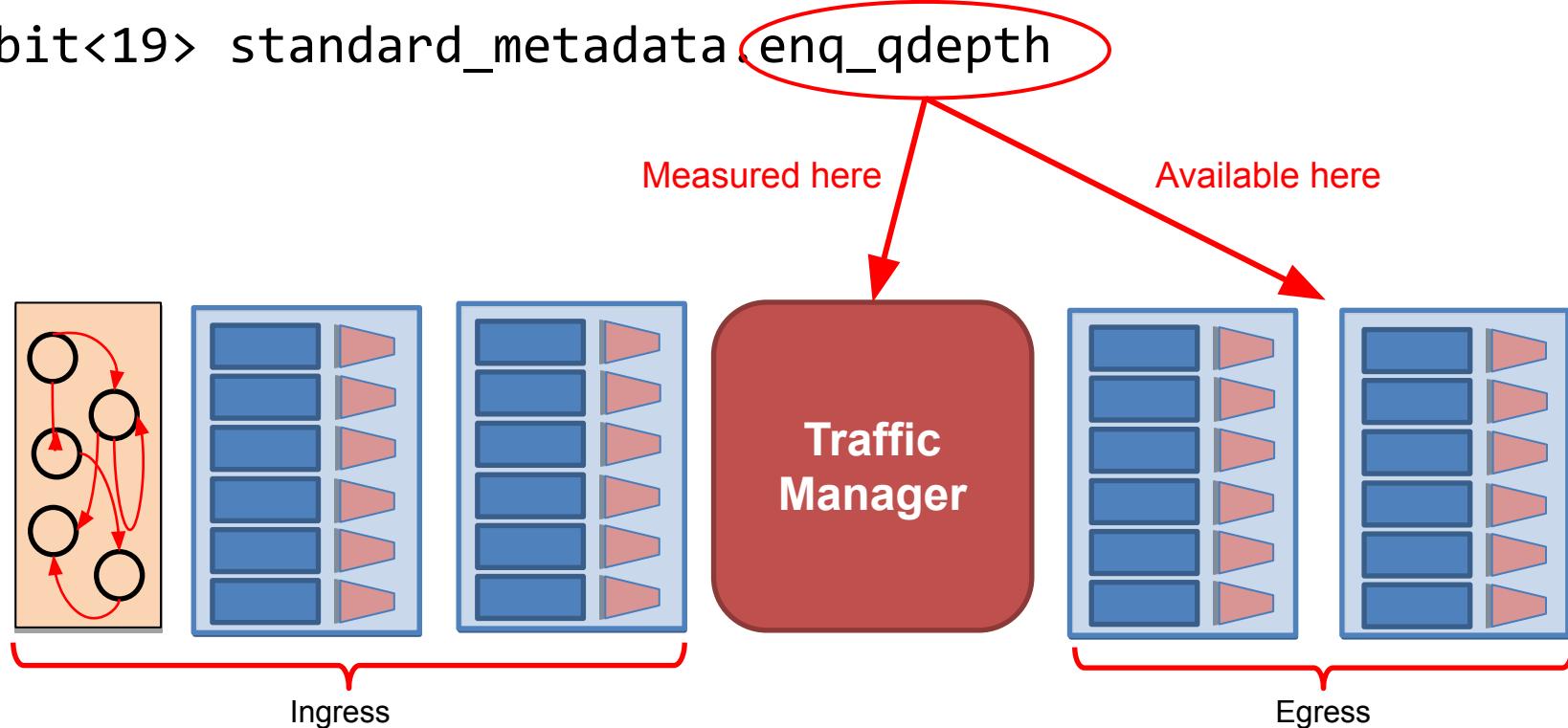
- **For packets originating from ECT, ECN-capable switches set the CE bit upon congestion**

- E.g., observed queue depth > threshold

Explicit Congestion Notification in P4

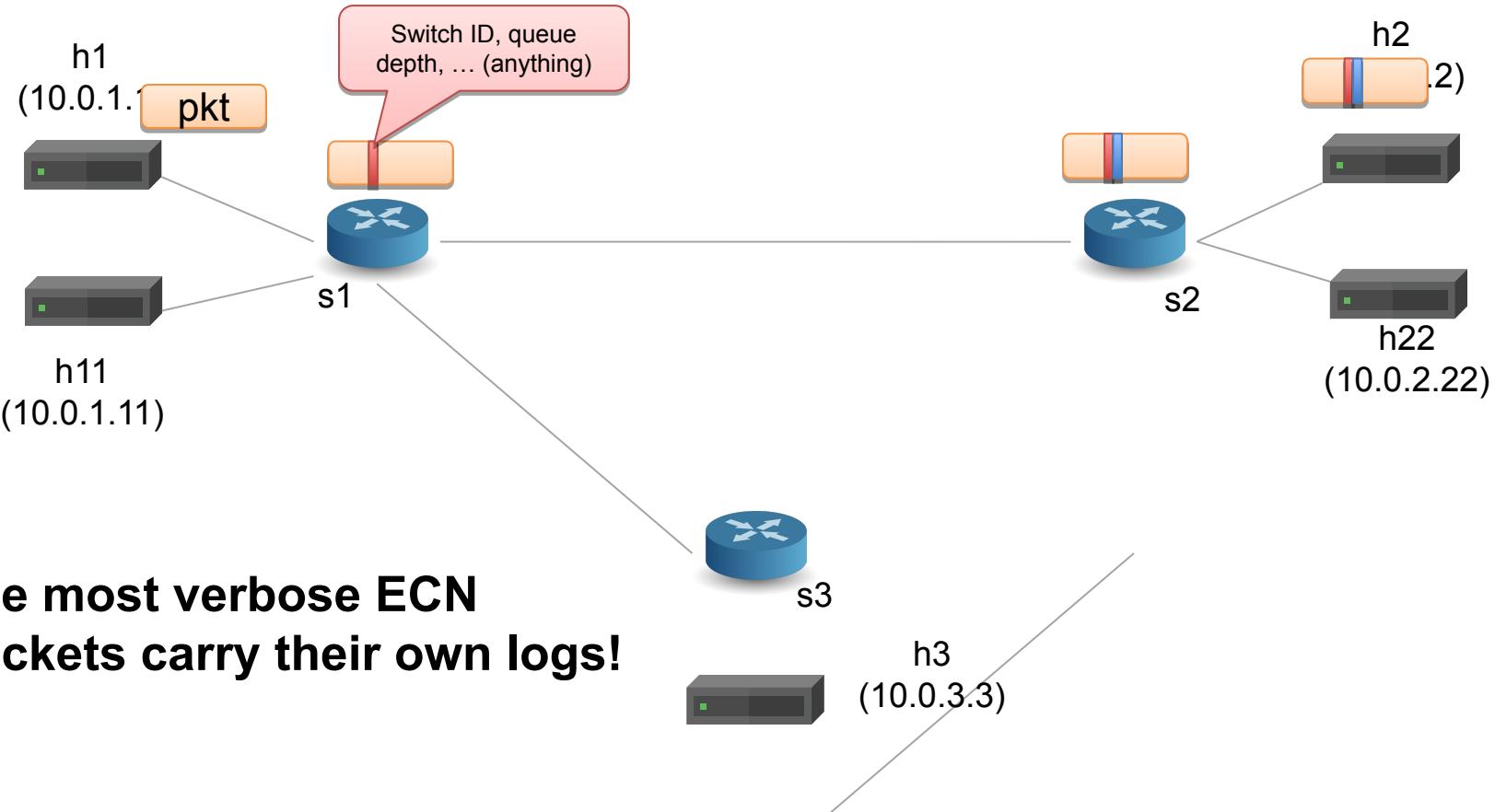
- The standard data for the V1Model includes the queue depth:

bit<19> standard_metadata.enq_qdepth



Coding Break

Multi-Route Inspection



Multi-Route Inspect: Packet Format

```
header mri_t {  
    bit<16> count;  
}
```

```
header switch_t {  
    switchID_t swid;  
    qdepth_t qdepth;  
}
```

```
struct headers {  
    ethernet_t          ethernet;  
    ipv4_t              ipv4;  
    ipv4_option_t       ipv4_option;  
    mri_t               mri;  
    switch_t[MAX_HOPS] swtraces;  
}
```

- **Header validity operations:**

- `hdr.setValid() : add_header`
- `hdr.setInvalid() : remove_header`
- `hdr.isValid() : test validity`

- **Header Stacks**

- `hdr[CNT] stk;`

- **Header Stacks in Parsers**

- `stk.next`
- `stk.last`
- `stk.lastIndex`

- **Header Stacks in Controls**

- `stk[i]`
- `stk.size`
- `stk.push_front(int count)`
- `stk.pop_front(int count)`

Header verification

```
/* Standard errors, defined in core.p4 */
error {
    NoError,          // no error
    PacketTooShort,   // not enough bits in packet for extract
    NoMatch,          // match expression has no matches
    StackOutOfBounds, // reference to invalid element of a header stack
    OverwritingHeader, // one header is extracted twice
    HeaderTooShort,   // extracting too many bits in a varbit field
    ParserTimeout     // parser execution time limit exceeded
}

/* Additional error added by the programmer */
error { IPv4BadHeader }

...
state parse_ipv4 {
    packet.extract(hdr.ipv4);
    verify(hdr.ipv4.version == 4, error.IPV4BadHeader);
    transition accept;
}
```

Coding Break

Questions and Answers



<https://bit.ly/join-p4-lang-slack>

Join #d2-2018-spring channel



<https://pigeonhole.at>

Event code: P4D2



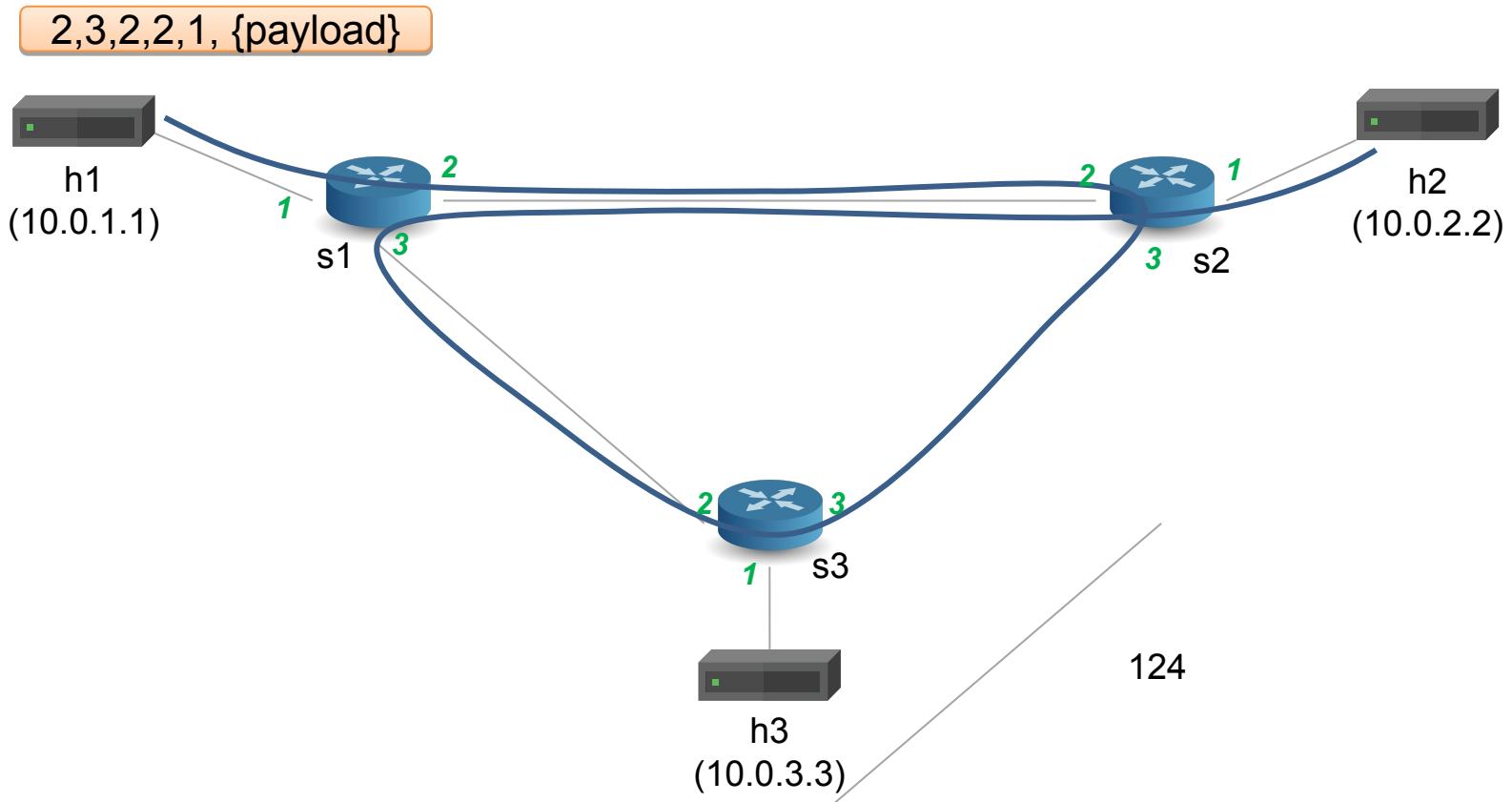
Questions for TAs



Questions for Panel

Lab 4: Advanced Behavior

Source Routing



Source Routing: Packet Format

```
#define MAX_HOPS 9

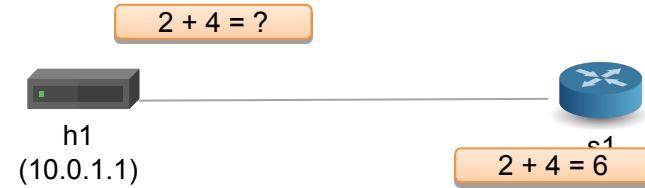
const bit<16> TYPE_IPV4 = 0x800;
const bit<16> TYPE_SRCROUTING = 0x1234;
header srcRoute_t {
    bit<1>    bos;
    bit<15>   port;
}

struct headers {
    ethernet_t          ethernet;
    srcRoute_t[MAX_HOPS] srcRoutes;
    ipv4_t               ipv4;
}
```

- Parse source routes only if etherType is **0x1234**
- The special value **bos == 1** indicates the “bottom of stack”
- Forward packets using source routes, and also decrement IPv4 TTL
- Drop the packet if source routes are not valid
- Hint: Use the next, pop_front primitives
`packet.extract(hdr.srcRoutes.next)`
`hdr.srcRoutes.pop_front(1)`

Coding Break

Calculator



Calculator: Packet Format

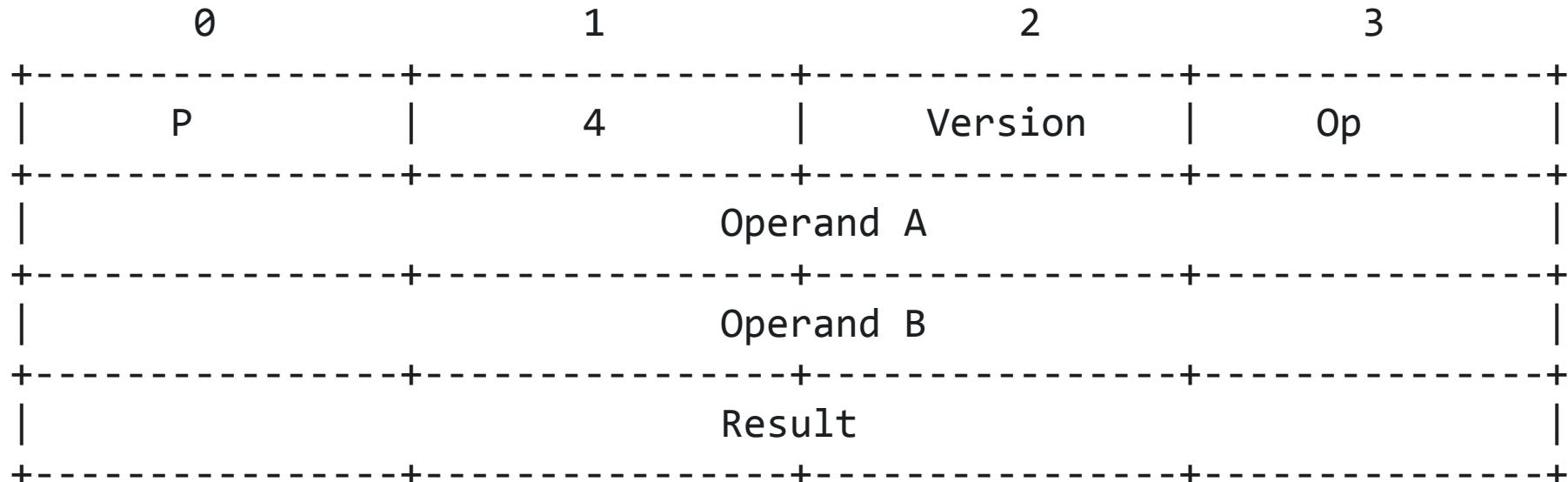


Table Initializers

```
table tbl {  
    key = { hdr.h.f : exact }  
    actions = { a1; a2; a3 }  
    entries = {  
        { 0x01 } : a1(1);  
        { 0x02 } : a1(2);  
        { _ } : NoAction();  
    }  
}
```

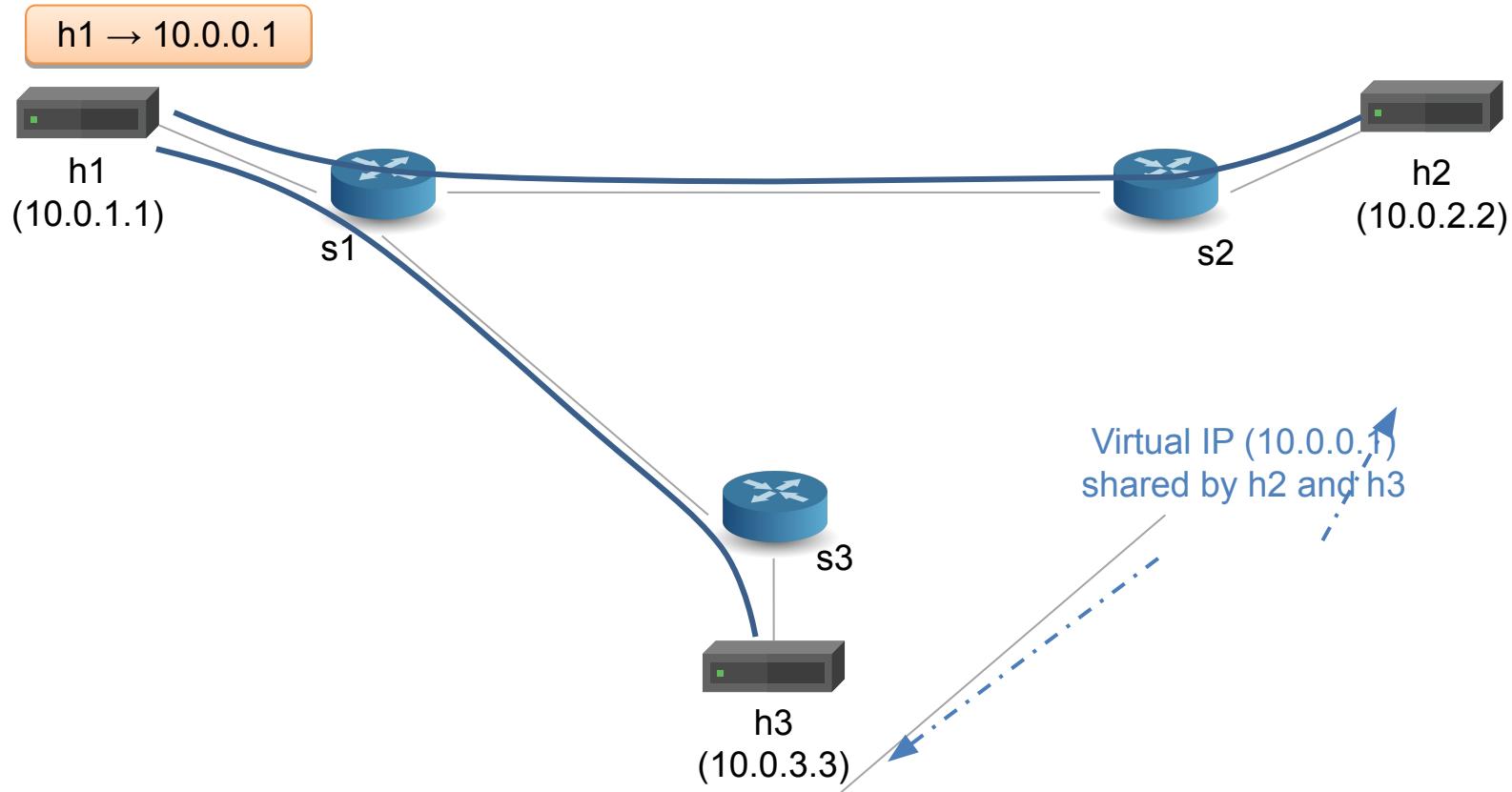
Can initialize tables with constant entries

Must fully specify the value of all action data, including values that are normally supplied by the control-plane

Hint: for the calculator, use a table that matches on the op-code

Coding Break

Simple Load Balancing



Hashing (V1Model)

```
enum HashAlgorithm {  
    csum16,  
    xor16,  
    crc32,  
    crc32_custom,  
    crc16,  
    crc16_custom,  
    random,  
    identity  
}  
  
extern void hash<O, T, D, M>(  
    out O result,  
    in HashAlgorithm algo,  
    in T base,  
    in D data,  
    in M max);
```

Computes the hash of data (using algo) modulo max and adds it to base

Uses type variables (like C++ templates / Java Generics) to allow hashing primitive to be used with many different types.

Wrapping up & Next Steps

Why P4₁₆?

- **Clearly defined semantics**
 - You can describe what your data plane program is doing
- **Expressive**
 - Supports a wide range of architectures through standard methodology
- **High-level, Target-independent**
 - Uses conventional constructs
 - Compiler manages the resources and deals with the hardware
- **Type-safe**
 - Enforces good software design practices and eliminates “stupid” bugs
- **Agility**
 - High-speed networking devices become as flexible as any software
- **Insight**
 - Freely mixing packet headers and intermediate results

Things we covered

- **The P4 "world view"**
 - Protocol-Independent Packet Processing
 - Language/Architecture Separation
 - If you can interface with it, it can be used
- **Key data types**
- **Constructs for packet parsing**
 - State machine-style programming
- **Constructs for packet processing**
 - Actions, tables and controls
- **Packet deparsing**
- **Architectures & Programs**

Things we didn't cover

- **Mechanisms for modularity**
 - Instantiating and invoking parsers or controls
- **Details of variable-length field processing**
 - Parsing and deparsing of options and TLVs
- **Architecture definition constructs**
 - How these “templated” definitions are created
- **Advanced features**
 - How to do learning, multicast, cloning, resubmitting
 - Header unions
- **Other architectures**
- **Control plane interface**



The P4 Language Consortium

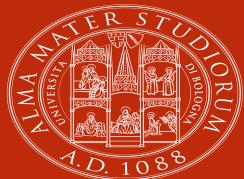
- Consortium of academic and industry members
- Open source, evolving, domain-specific language
- Permissive Apache license, code on GitHub today
- Membership is free: contributions are welcome
- Independent, set up as a California nonprofit

The screenshot shows the homepage of the P4 Language Consortium at https://p4.org. The page features a large banner image of a person working on a computer. Overlaid on the banner are four sections: "Protocol Independent" (describing how P4 programs specify packet processing), "Target Independent" (describing its use from ASICs to software switches), "Field Reconfigurable" (allowing network engineers to change switch behavior after deployment), and a code snippet for a "routing" table. A "TRY IT! GET THE CODE ON GITHUB" button is visible at the bottom right.

P4 programs specify how a switch processes packets.

```
table routing {  
    key = { ipv4.dstAddr : lpm; }  
    actions = { drop; route; }  
    size : 2048;  
}  
control.ingress() {  
    apply {  
        routing.apply();  
    }  
}
```

TRY IT! GET THE CODE ON GITHUB



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Laboratory of Network Programmability and Automation

ONOS+P4 Tutorial Controlling P4 data planes with ONOS

Andrea Melis

Dipartimento di Informatica – Scienza e
Ingegneria

Copyright

The following slides belong to:

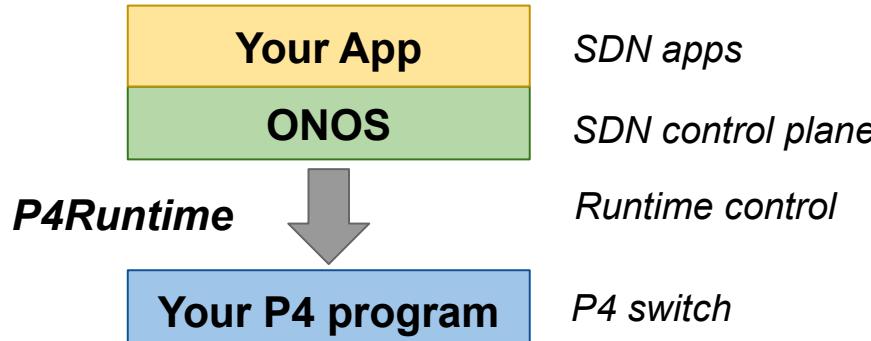
The Open Network Foundtion

Link to slides:

<http://bit.ly/onos-p4-tutorial-slides>

Goal of this session

- Learn the basics of P4, P4Runtime and ONOS
- Show you the “big picture” of P4
 - Acquire enough knowledge to build full-stack network applications
 - Go from a P4 idea to an end-to-end solution
- Learn the tools to practically experiment with it

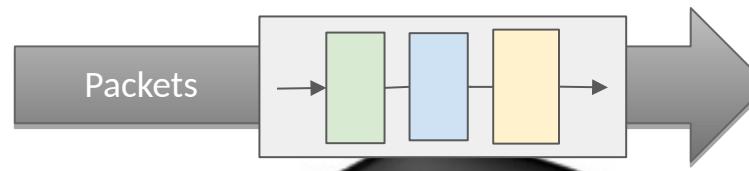


P4

The Data Plane Programming Language

What is a pipeline?

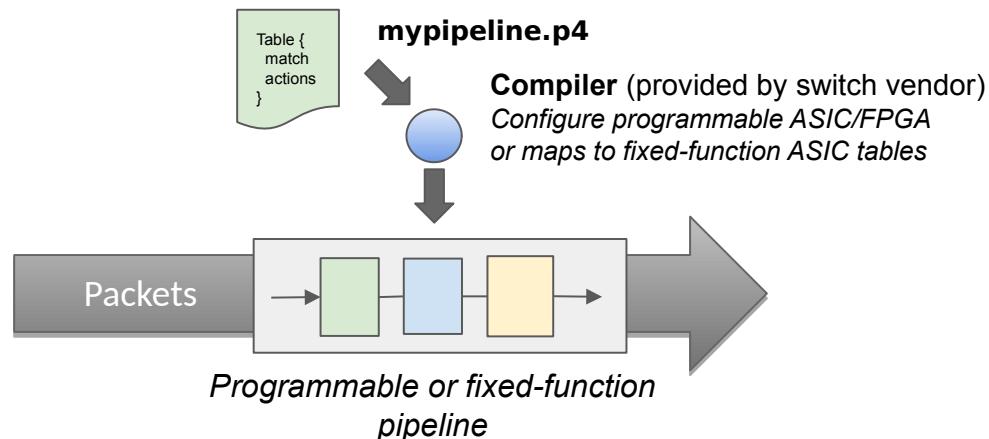
Pipeline of match-action tables



ASIC, FPGA, NPU, or CPU



- **Domain-specific language to formally define the data plane pipeline**
 - Describe protocol headers, lookup tables, actions, counters, etc.
 - Can describe fast pipelines (e.g ASIC, FPGA) as well as a slower ones (e.g. SW switch)
- **Good for programmable switches, as well as fixed-function ones**
 - Defines “contract” between the control plane and data plane for runtime control



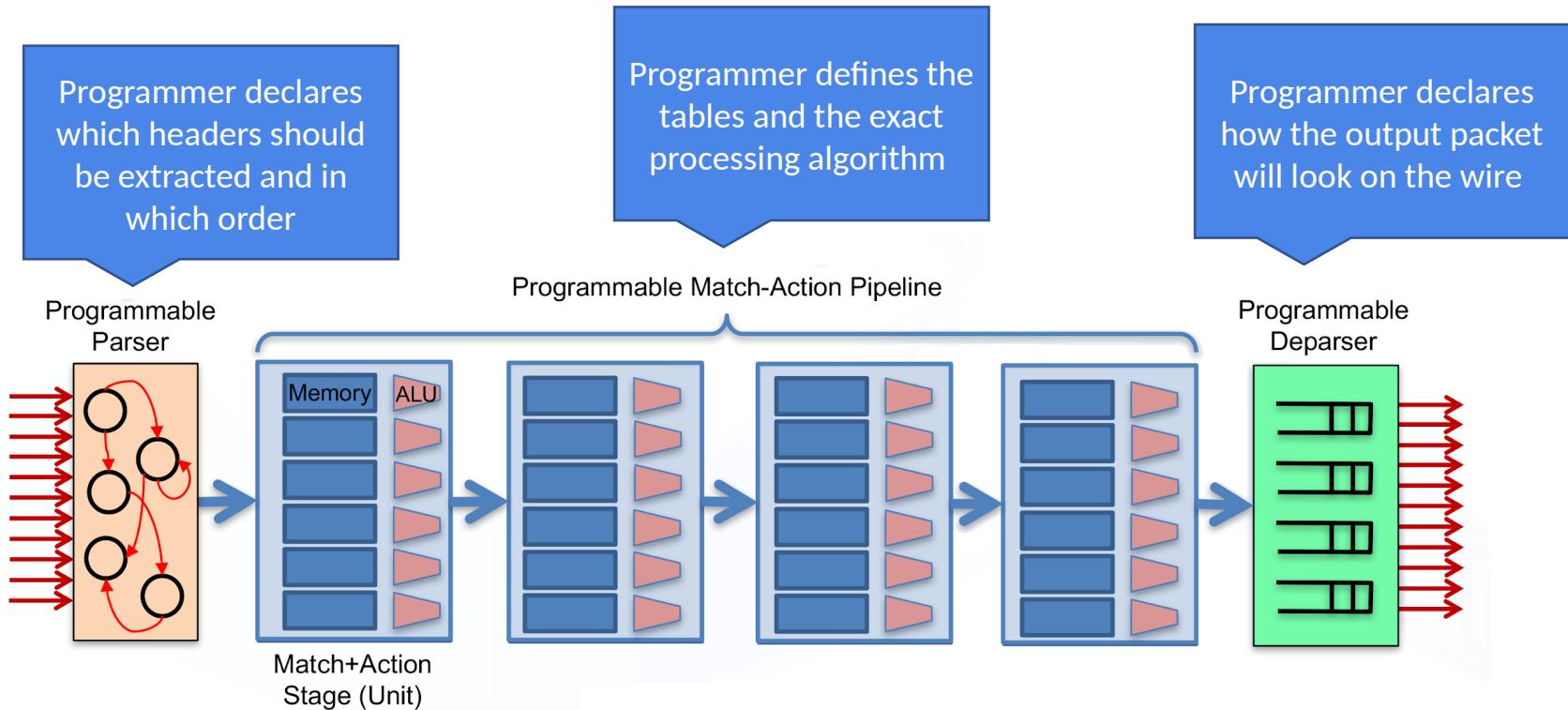
Evolution of the language

- **P4₁₄**
 - Original version of the language
 - Assumed specific device capabilities
 - Good only for a subset of programmable switch/targets
- **P4₁₆ (Focus of this tutorial)**
 - More mature and stable language definition
 - Does not assume device capabilities
 - Defined via external libraries/architecture definition
 - Good for many targets
 - E.g., switches or NICs, programmable or fixed-function

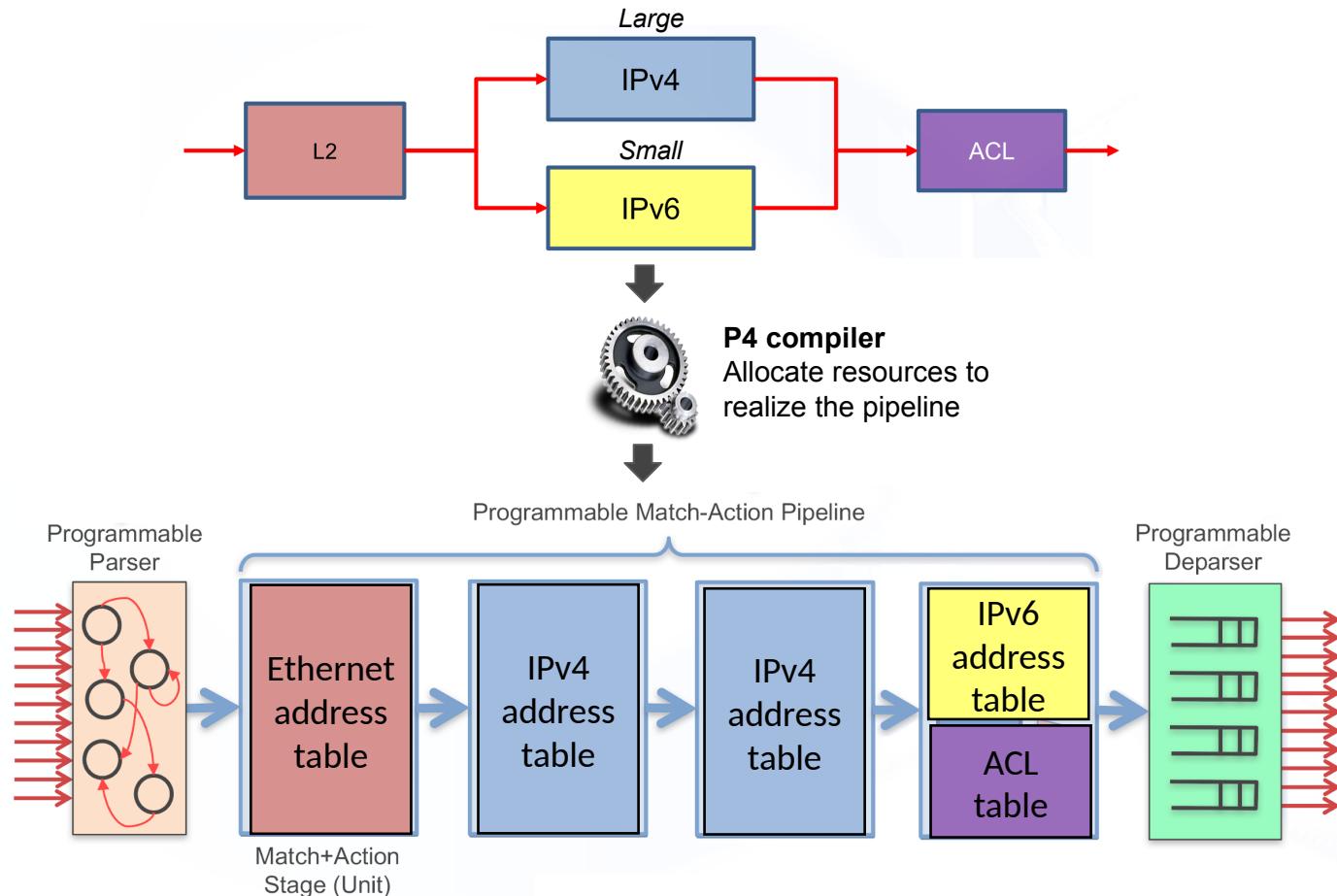
PISA: Protocol-Independent Switch Architecture

8

Abstract machine model of a high-speed programmable switch architecture

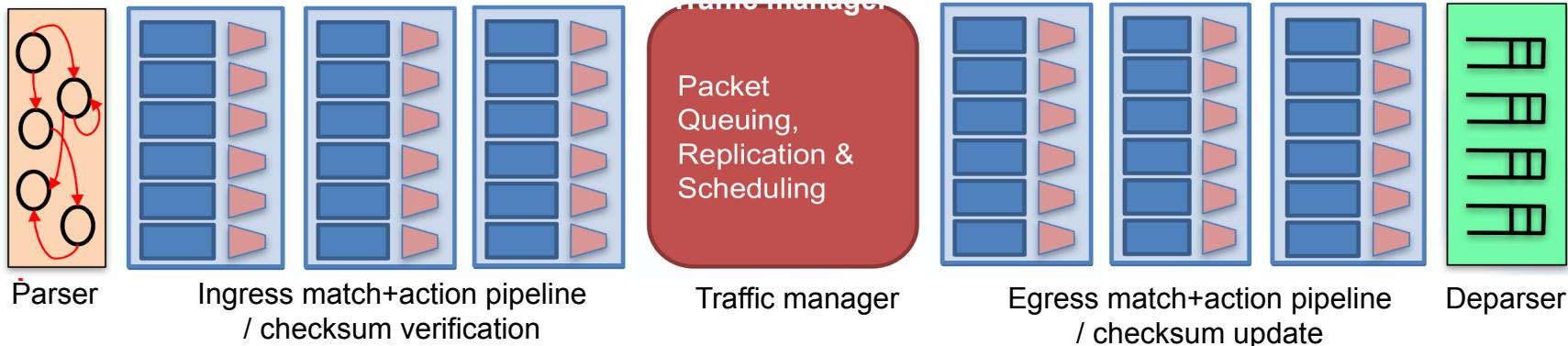


Compiling a simple logical pipeline on PISA

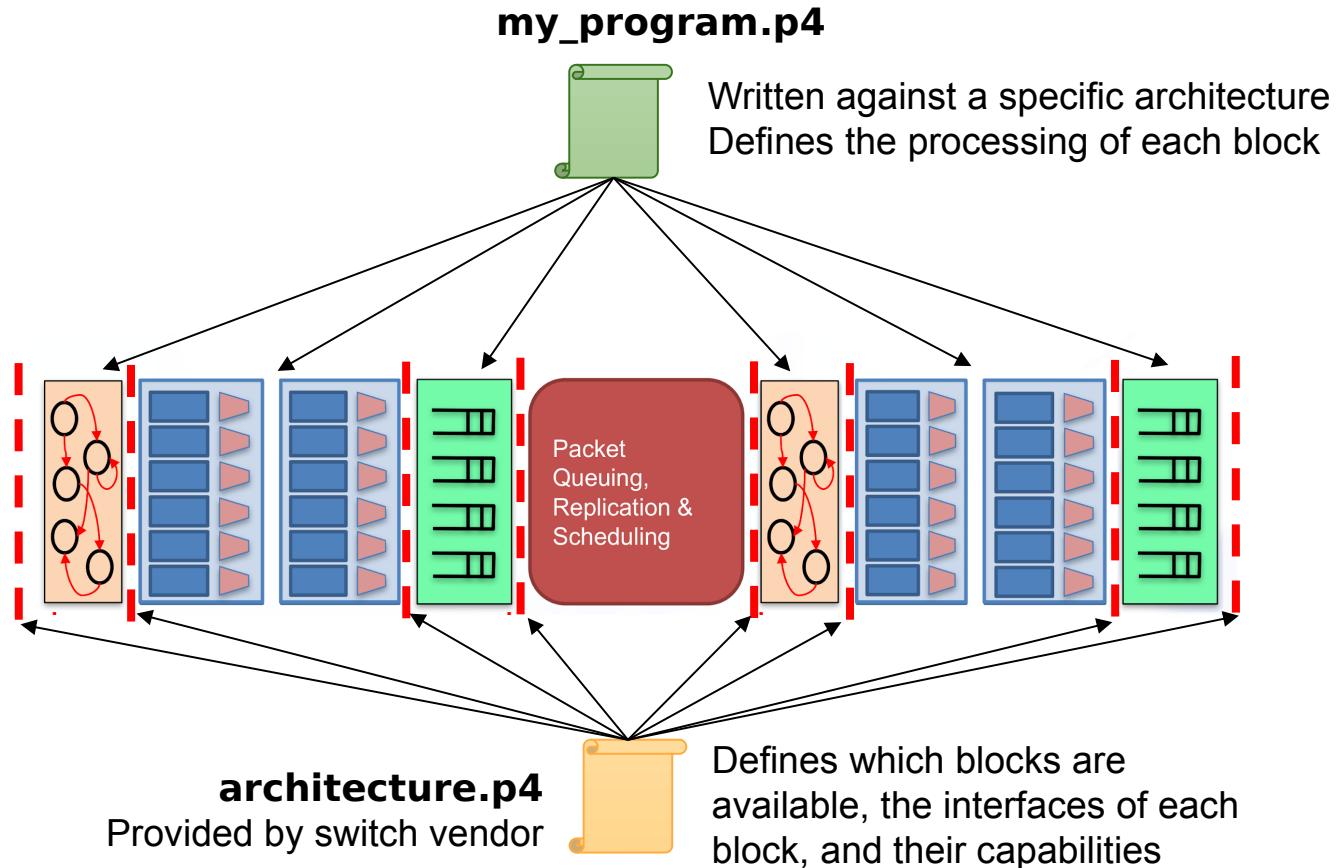


V1Model P4 Switch Architecture

- Parser/deparser → **P4 programmable**
- Checksum verification/update → **P4 programmable**
- Ingress Pipeline → **P4 programmable**
- Egress Pipeline → **P4 programmable**
 - Match on egress port
- **Traffic Manager** → **Fixed function**
 - Queueing, Replication (multicast), scheduling



P4 architectures



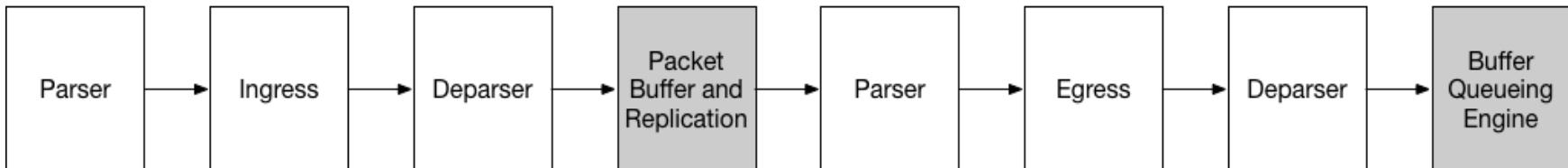
Preliminary takeaways

- **Can I implement/describe this or that feature with P4?**
 - It depends on the architecture
 - The P4 language is flexible enough to express almost any behavior based on match-action tables
 - Specific capabilities depend on the architecture
 - e.g. ternary match vs. longest-prefix match vs. exact match
- **Can I execute my P4 program on a switch X from vendor Y?**
 - Probably yes, if vendor Y provides a P4 compiler for the specific architecture used

Architectures enable portability of P4 programs across different HW and SW targets

PSA - Portable Switch Architecture

- Community-developed architecture (P4.org Arch WG)
 - <https://github.com/p4lang/p4-spec/tree/master/p4-16/psa>
- Describes common capabilities of a network switch
- 6 programmable P4 blocks + 2 fixed-function blocks
- Defines capabilities beyond match+action tables
 - Counters, meters, stateful registers, hash functions, etc.



P4 program template (V1Model)

```
#include <core.p4>
#include <v1model.p4>

/* HEADERS */
struct metadata { ... }
struct headers {
    ethernet_t    ethernet;
    ipv4_t        ipv4;
}

/* PARSER */
parser MyParser(packet_in packet,
                 out headers hdr,
                 inout metadata meta,
                 inout standard_metadata_t smeta) {
    ...
}

/* CHECKSUM VERIFICATION */
control MyVerifyChecksum(in headers hdr,
                         inout metadata meta) {
    ...
}

/* INGRESS PROCESSING */
control MyIngress(inout headers hdr,
                  inout metadata meta,
                  inout standard_metadata_t std_meta) {
    ...
}
```

```
/* EGRESS PROCESSING */
control MyEgress(inout headers hdr,
                  inout metadata meta,
                  inout standard_metadata_t std_meta) {
    ...
}

/* CHECKSUM UPDATE */
control MyComputeChecksum(inout headers hdr,
                         inout metadata meta) {
    ...
}

/* DEPARSER */
control MyDeparser(inout headers hdr,
                    inout metadata meta) {
    ...
}

/* SWITCH */
V1Switch(
    MyParser(),
    MyVerifyChecksum(),
    MyIngress(),
    MyEgress(),
    MyComputeChecksum(),
    MyDeparser()
) main;
```

P4 program example: simple_router.p4

```
header ethernet_t {  
    bit<48> dst_addr;  
    bit<48> src_addr;  
    bit<16> eth_type;  
}  
  
header ipv4_t {  
    bit<4> version;  
    bit<4> ihl;  
    bit<8> diffserv;  
    ...  
}  
  
parser parser_impl(packet_in pkt, out headers_t hdr) {  
    /* Parser state machine to extract header fields */  
}
```

```
action set_next_hop(bit<48> dst_addr) {  
    ethernet.dst_addr = dst_addr;  
    ipv4.ttl = ipv4.ttl - 1;  
}  
...  
table ipv4_routing_table {  
    key = {  
        ipv4.dst_addr : LPM; // longest-prefix match  
    }  
    actions = {  
        set_next_hop();  
        drop();  
    }  
    size = 4096; // table entries  
}
```

Runtime: simple_router.p4

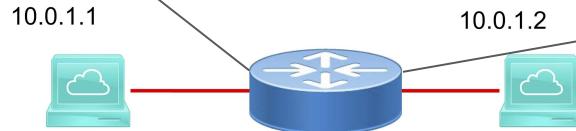
- **Data plane program (P4)**
 - Defines the match-action tables
 - Performs the lookup
 - Executes the chosen action
- **Control plane (runtime)**
 - Populates table entries with specific information
 - Based on configuration, automatic discovery, protocol calculations

```

action ipv4_forward(bit<48> dst_addr, bit<9> port) {
    ethernet.dst_addr = dst_addr;
    standard_metadata.egress_spec = port;
    ipv4.ttl = ipv4.ttl - 1;
}

table ipv4_routing_table {
    key = {
        ipv4.dst_addr : LPM; // longest-prefix match
    }
    actions = {
        ipv4_forward();
        drop();
    }
}

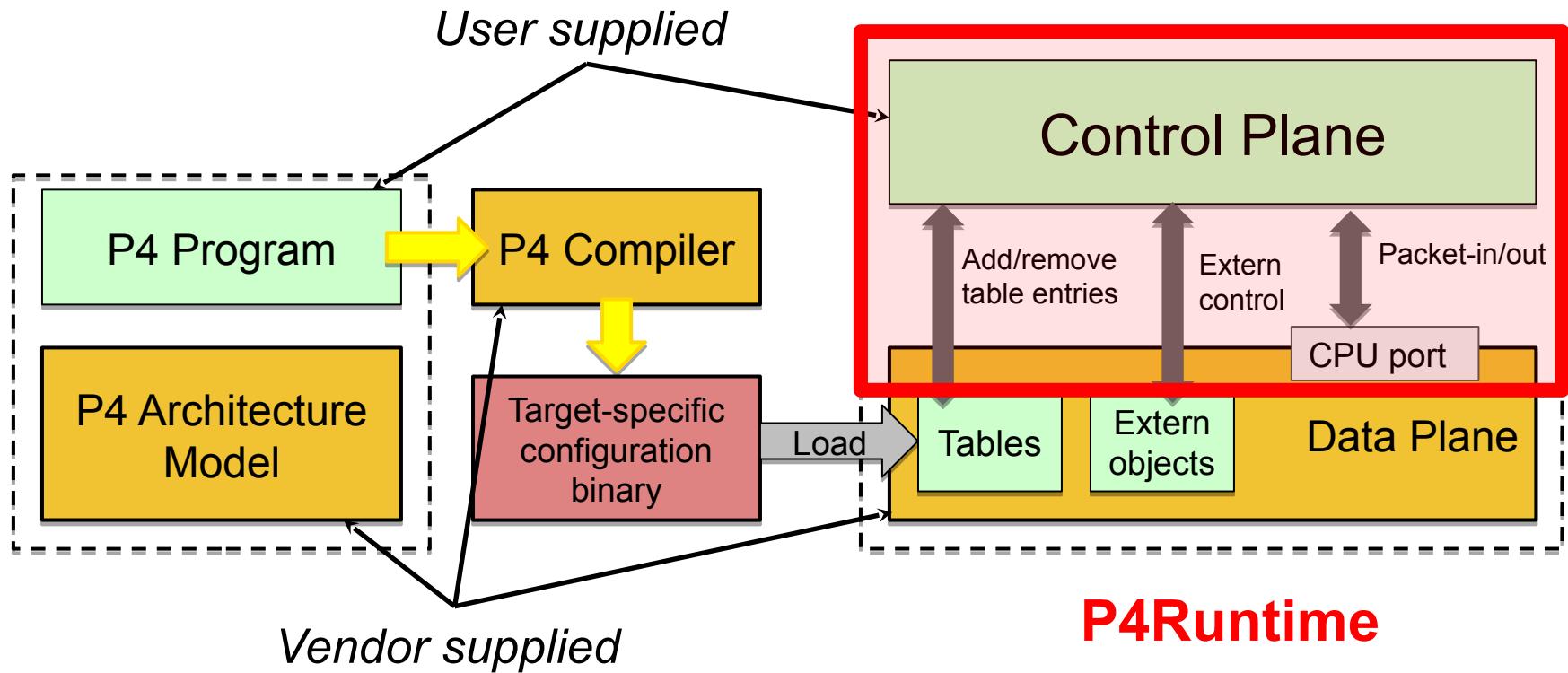
```



Control plane populates table entries

Key	Action	Action Data
10.0.1.1/32	ipv4_forward	dstAddr=00:00:00:00:01:01 port=1
10.0.1.2/32	drop	
*`	NoAction	

P4 workflow summary

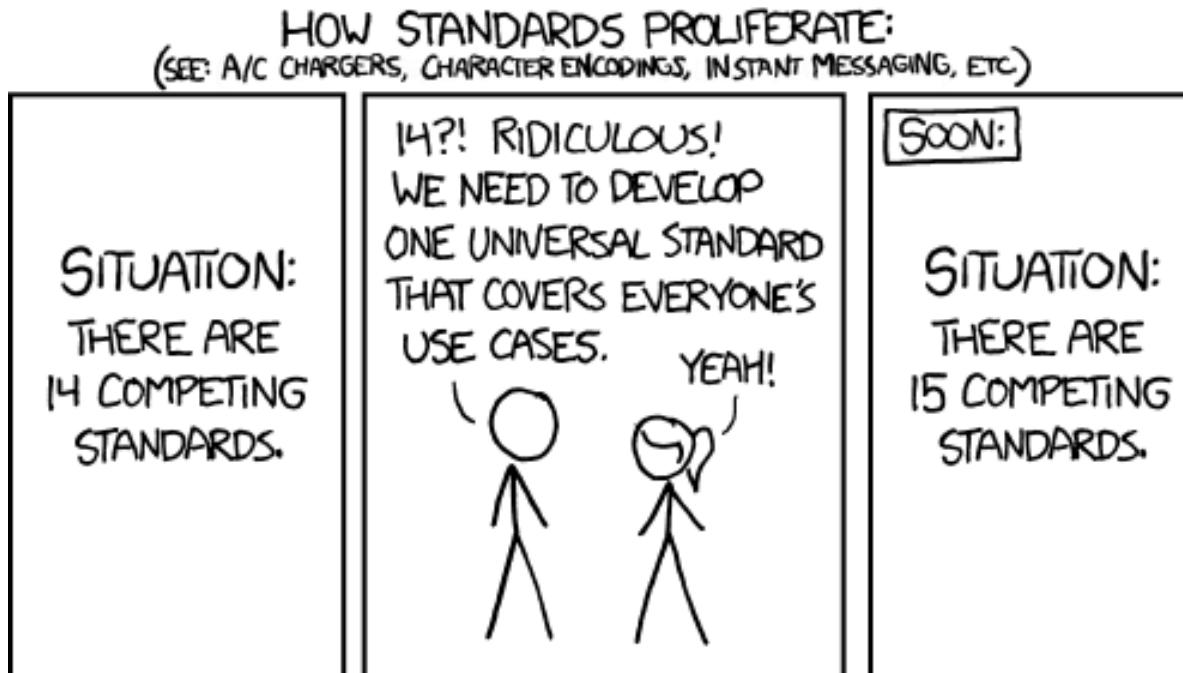


P4Runtime

Runtime control API for P4-defined data planes

Do we need yet another data plane control API?

Can't we re-use existing APIs such as OpenFlow or SAI?



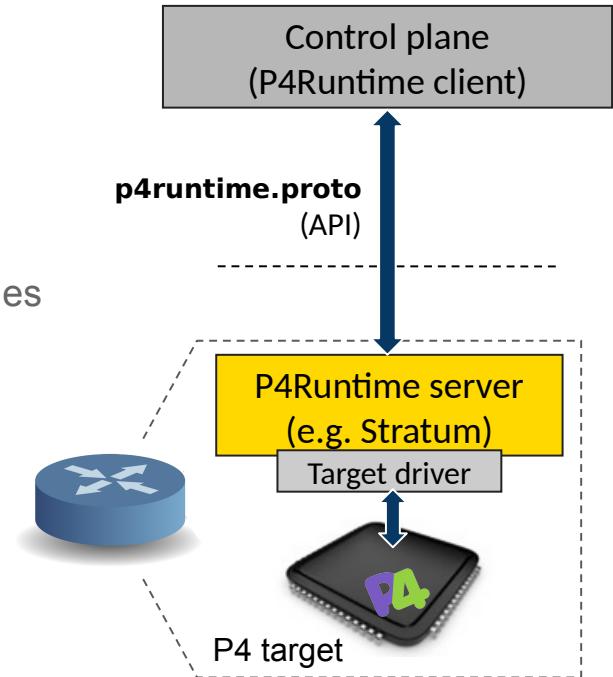
Credits XKCD

Yes, we need P4Runtime

API	Target-independent	Protocol-independent	Pipeline-independent
OpenFlow	Same API works with different switches from different vendors	Same API allows control of any data plane protocol, standard or custom ✖ Protocol headers and actions hard-coded in the spec	Same API allows control of many arbitrary pipelines formally specified (with TTP)
Switch Abstraction Interface (SAI)		✖ Designed for legacy forwarding pipelines (L2/L3/ACL)	✖ Implicit fixed-function pipeline
P4Runtime			(with P4)

P4Runtime overview

- API for runtime control of P4-defined switches
 - Designed around PSA architecture but can be extended to others
- Community-developed (p4.org API WG)
 - Initial contribution by Google and Barefoot
 - RC of version 1.0 available: <https://p4.org/p4-spec/>
- gRPC/protobuf-based API definition
 - Automatically generate client/server code for many languages
- P4 program-independent
 - API doesn't change with the P4 program
- Enables field-reconfigurability
 - Ability to push new P4 program, i.e. re-configure the switch pipeline, without recompiling the switch software stack



Protocol Buffers (protobuf) Basics

- Language for describing data for serialization in a structured way
- Strongly typed
- Auto-generate code to serialize/deserialize messages in:
 - Code generators for: *Action Script, C, C++, C#, Clojure, Lisp, D, Dart, Erlang, Go, Haskell, Java, Javascript, Lua, Objective C, OCaml, Perl, PHP, Python, Ruby, Rust, Scala, Swift, Visual Basic, ...*
- Platform-neutral
- Extensible and backwards compatible

```
syntax = "proto3";

message Person {
    string name = 1;
    int32 id = 2;
    string email = 3;

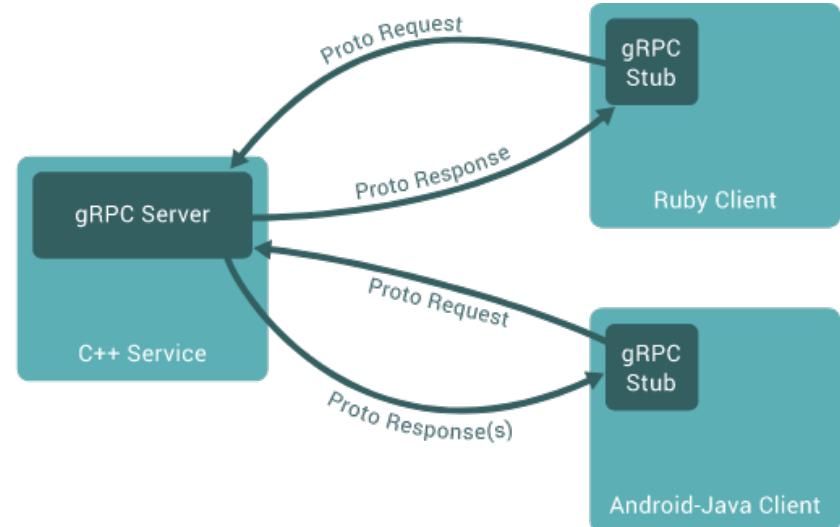
    enum PhoneType {
        MOBILE = 0;
        HOME = 1;
        WORK = 2;
    }

    message PhoneNumber {
        string number = 1;
        PhoneType type = 2;
    }

    repeated PhoneNumber phone = 4;
}
```

gRPC Basics

- Use Protobuf to define service API and messages
- Automatically generate native client and server code in:
 - C / C++, C#, Dart, Go, Java, Node.js, PHP, Python, Ruby
- Transport over HTTP/2.0 and TLS
 - Efficient single TCP connection implementation that supports bidirectional streaming



p4runtime.proto (gRPC service)

Enables a local or remote control plane to

- Load the pipeline/P4 program
- Write/read pipeline state (e.g. table entries, meters, groups, etc.)
- Send/receive packets to/from controller (packet-in/out)
- Arbitrate mastership (for distributed control planes)

```
service P4Runtime {
    rpc Write(WriteRequest) returns (WriteResponse) {}
    rpc Read(ReadRequest) returns (stream ReadResponse) {}

    SetForwardingPipelineConfig(SetForwardingPipelineConfigRequest)
        returns (SetForwardingPipelineConfigResponse) {}
    rpc

    GetForwardingPipelineConfig(GetForwardingPipelineConfigRequest)
        returns (GetForwardingPipelineConfigResponse) {}
    rpc StreamChannel(stream StreamMessageRequest)
        returns (stream StreamMessageResponse) {}
```

From: <https://github.com/p4lang/p4runtime/blob/master/proto/p4/v1/p4runtime.proto>

P4Runtime Write Request

```
message WriteRequest {  
    uint64 device_id = 1;  
    uint64 role_id = 2;  
    Uint128 election_id = 3;  
    repeated Update updates = 4;  
}
```

```
message Update {  
    enum Type {  
        UNSPECIFIED = 0;  
        INSERT = 1;  
        MODIFY = 2;  
        DELETE = 3;  
    }  
    Type type = 1;  
    Entity entity = 2;
```

```
message Entity {  
    oneof entity {  
        ExternEntry extern_entry = 1;  
        TableEntry table_entry = 2;  
        ActionProfileMember  
            action_profile_member = 3;  
        ActionProfileGroup  
            action_profile_group = 4;  
        MeterEntry meter_entry = 5;  
        DirectMeterEntry direct_meter_entry = 6;  
        CounterEntry counter_entry = 7;  
        DirectCounterEntry direct_counter_entry = 8;  
        PacketReplicationEngineEntry  
            packet_replication_engine_entry = 9;  
        ValueSetEntry value_set_entry = 10;  
        RegisterEntry register_entry = 11;  
    }  
}
```

P4Runtime TableEntry

p4runtime.proto simplified excerpts:

```
message TableEntry {
    uint32 table_id;
    repeated FieldMatch match;
    Action action;
    int32 priority;
    ...
}
```

```
message Action {
    uint32 action_id;
    message Param {
        uint32 param_id;
        bytes value;
    }
    repeated Param params;
}
```

```
message FieldMatch {
    uint32 field_id;
    message Exact {
        bytes value;
    }
    message Ternary {
        bytes value;
        bytes mask;
    }
    message LPM {
        bytes value;
        uint32 prefix_length;
    }
    ...
    oneof field_match_type {
        Exact exact;
        Ternary ternary;
        LPM lpm;
        ...
    }
}
```

To add a table entry, the control plane needs to know:

- **IDs of P4 entities**
 - Tables, field matches, actions, params, etc.
- **Field matches for the particular table**
 - Match type, bit width, etc.
- **Parameters for the particular action**
- **Other P4 program attributes**

P4 compiler workflow

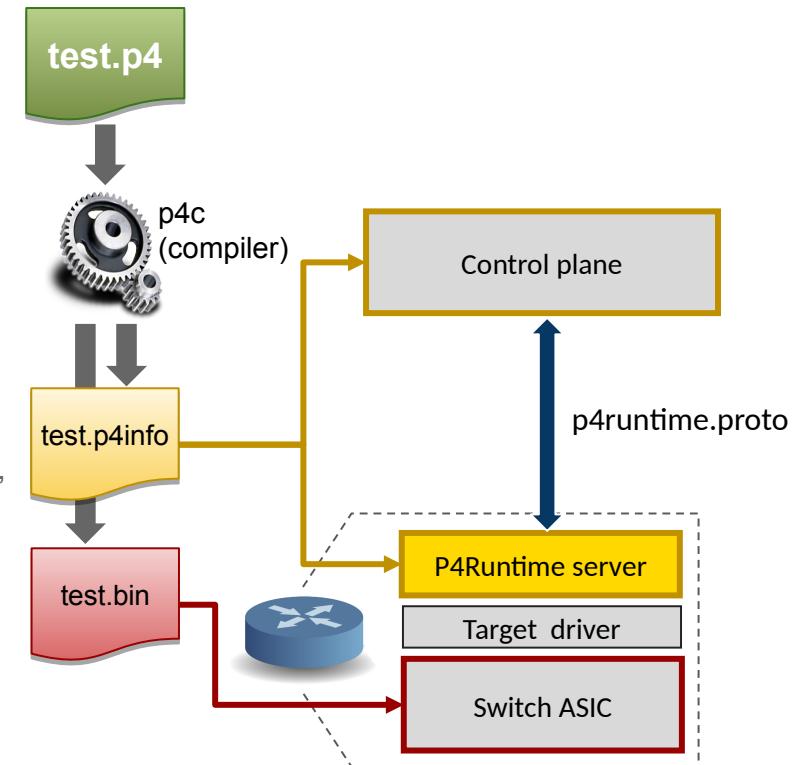
P4 compiler generates 2 outputs:

1. Target-specific binaries

- Used to realize switch pipeline
(e.g. binary config for ASIC, bitstream for FPGA, etc.)

2. P4Info file

- Describes “schema” of pipeline for runtime control
 - Captures P4 program attributes such as tables, actions, parameters, etc.
- Protobuf-based format
- Target-independent compiler output
 - Same P4Info for SW switch, ASIC, etc.



Full P4Info protobuf specification:

<https://github.com/p4lang/p4runtime/blob/master/proto/p4/config/v1/p4info.proto>

P4Info example

basic_router.p4

```

...
action ipv4_forward(bit<48> dstAddr,
                    bit<9> port) {
    /* Action implementation */
}

...
table ipv4_lpm {
    key = {
        hdr.ipv4.dstAddr: lpm;
    }
    actions = {
        ipv4_forward;
    ...
}
...
}
```



P4 compiler

basic_router.p4info

```

actions {
    id: 16786453
    name: "ipv4_forward"
    params {
        id: 1
        name: "dstAddr"
        bitwidth: 48
        ...
        id: 2
        name: "port"
        bitwidth: 9
    }
}
...
tables {
    id: 33581985
    name: "ipv4_lpm"
    match_fields {
        id: 1
        name: "hdr.ipv4.dstAddr"
        bitwidth: 32
        match_type: LPM
    }
    action_ref_id: 16786453
}
```

P4Runtime table entry example

basic_router.p4

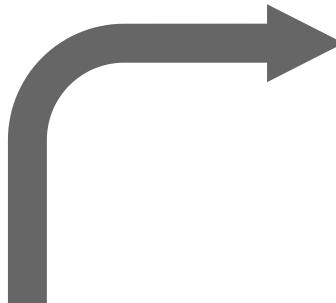
```
action ipv4_forward(bit<48> dstAddr,
                    bit<9> port) {
    /* Action implementation */
}

table ipv4_lpm {
    key = {
        hdr.ipv4.dstAddr: lpm;
    }
    actions = {
        ipv4_forward;
        ...
    }
    ...
}
```

Logical view of table entry

hdr.ipv4.dstAddr=10.0.1.1/32
-> ipv4_forward(00:00:00:00:00:10, 7)

Control plane generates

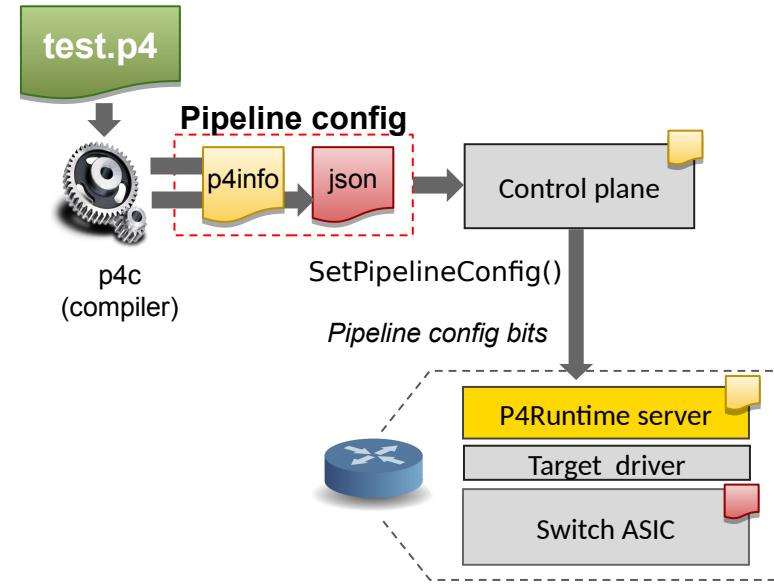


Protobuf message

```
table_entry {
    table_id: 33581985
    match {
        field_id: 1
        lpm {
            value: "\n\000\001\001"
            prefix_len: 32
        }
    }
    action {
        action_id: 16786453
        params {
            param_id: 1
            value: "\000\000\000\000\000\n"
        }
        params {
            param_id: 2
            value: "\000\007"
        }
    }
}
```

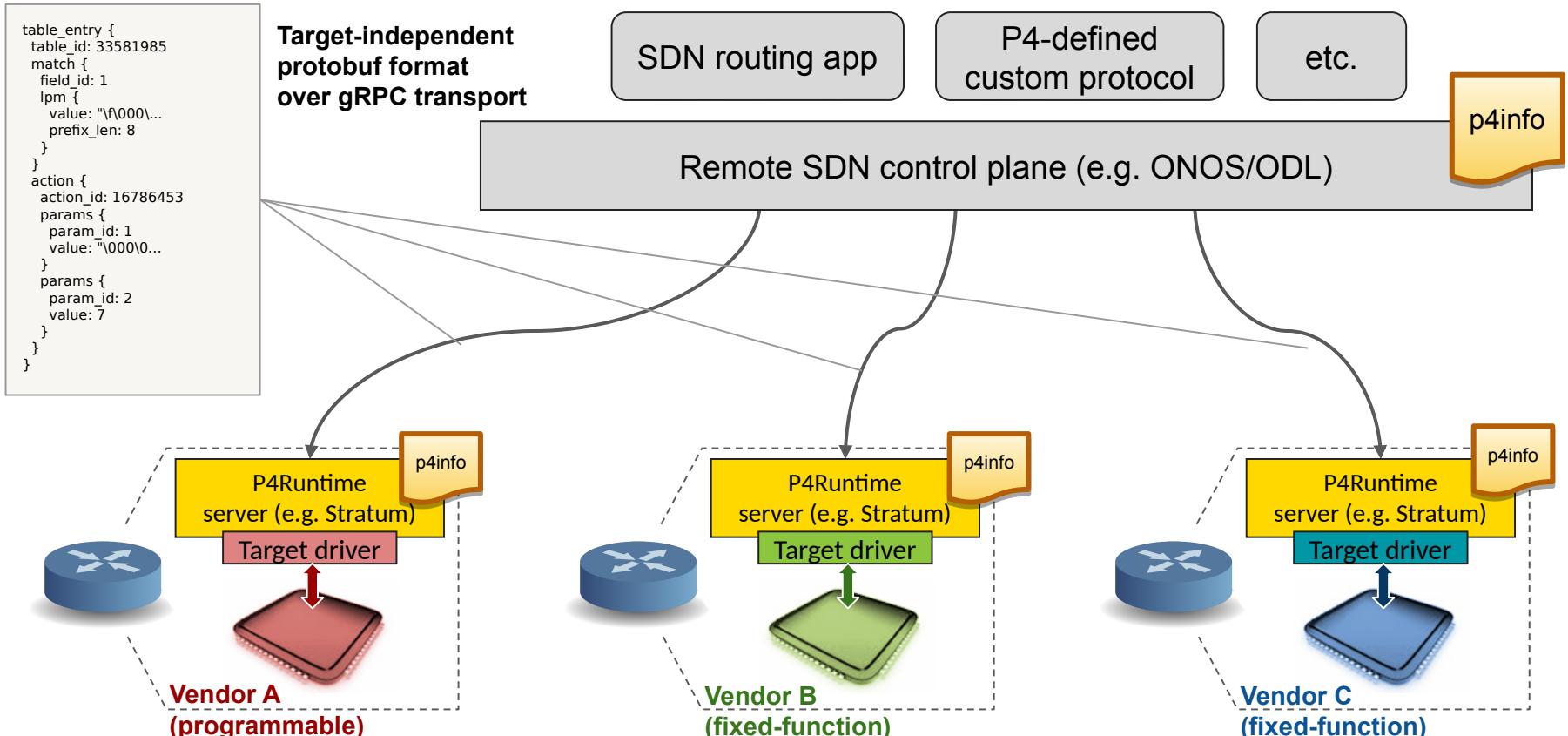
P4Runtime SetPipelineConfig

```
message
SetForwardingPipelineConfigRequest {
    enum Action {
        UNSPECIFIED = 0;
        VERIFY = 1;
        VERIFY_AND_SAVE = 2;
        VERIFY_AND_COMMIT = 3;
        COMMIT = 4;
        RECONCILE_AND_COMMIT = 5;
    }
    uint64 device_id = 1;
    uint64 role_id = 2;
    Uint128 election_id = 3;
    Action action = 4;
    ForwardingPipelineConfig config = 5;
}
```

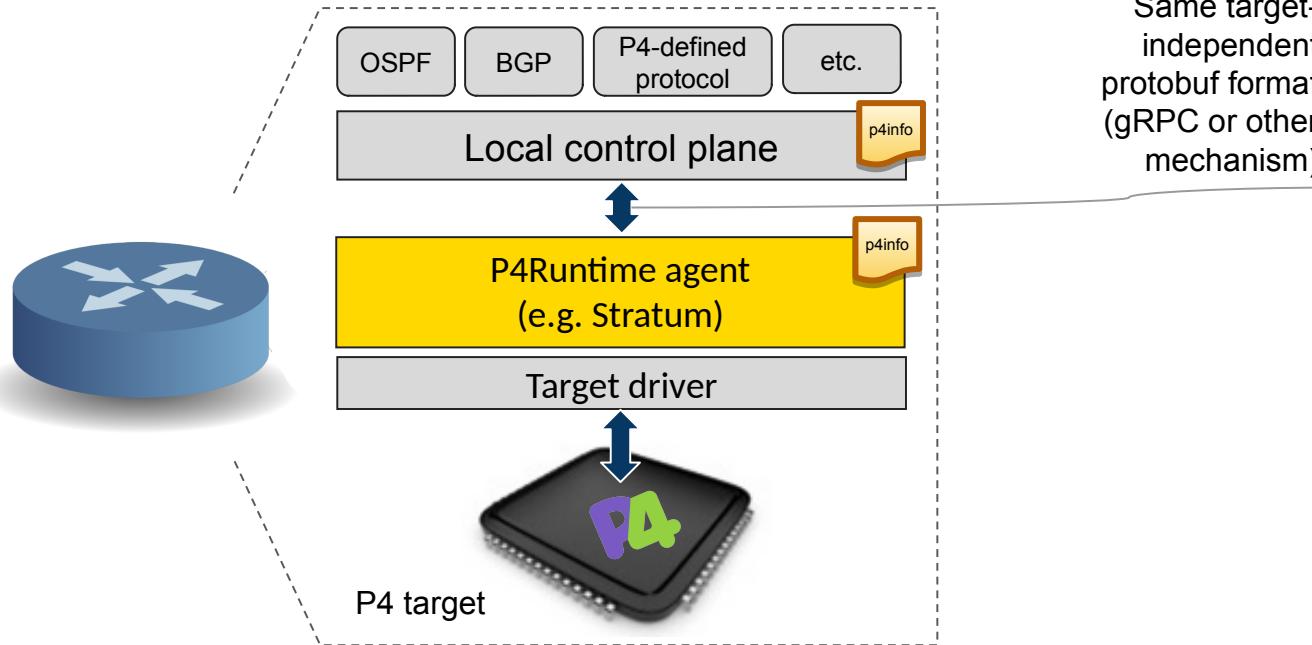


```
message ForwardingPipelineConfig {
    config.P4Info p4info = 1;
    // Target-specific P4
    configuration.
    bytes p4_device_config = 2;
}
```

Silicon-independent remote control



Portability of local control plane



Same target-independent
protobuf format
(gRPC or other
mechanism)

```
table_entry {
    table_id: 33581985
    match {
        field_id: 1
        lpm {
            value: "\f\000\0...
            prefix_len: 8
        }
    }
    action {
        action_id: 16786453
        params {
            param_id: 1
            value: "\000\0...
        }
        params {
            param_id: 2
            value: 7
        }
    }
}
```

P4Runtime summary

- **P4Runtime is an improvement over previous data plane APIs**
 - Realizes the vision of OpenFlow 2.0
 - Provides protocol and pipeline-independence
 - Protocols supported and pipeline are formally specified using P4
- **Based on protobuf and gRPC**
 - Makes it easy to implement a P4Runtime client/server by auto-generating code
- **P4Info as a contract between control and data plane**
 - Generated by P4 compiler
 - Needed by the control plane to format the body of P4Runtime messages (e.g. to add table entry)

ONOS

A control plane for P4Runtime devices

What is ONOS?

- **Open Network Operating System (ONOS)**
- **Provides the control plane for a software-defined network**
 - Logically centralized remote controller
 - Provides APIs to make it easy to create apps to control a network
- **Focus on service provider for access/edge applications**
- **Runs as a distributed system across many servers**
 - For scalability, high-availability, and performance
- **Open source project**
 - Created by ON.Lab and currently hosted by the Linux Foundation
 - Active community: 8,152 commits and 276 authors in last 2 years

ONOS releases

4-month release cycles

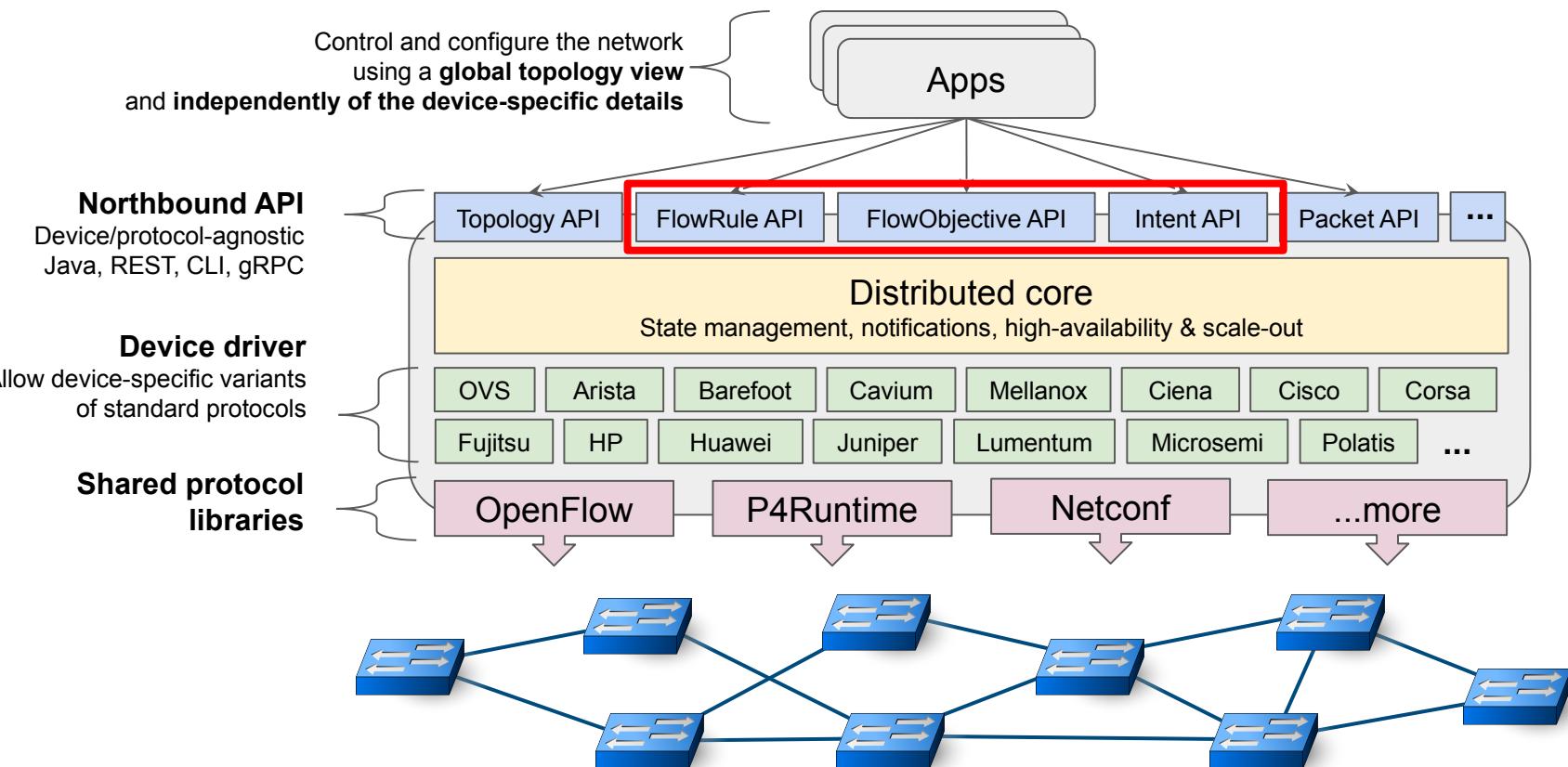
- **Avocet (1.0.0)** 2014-12
- ...
- **Loon (1.11.0)** 2017-08 (*Initial P4Runtime support*)
- **Magpie (1.12.0)** 2017-12
- **Nightingale (1.13.0)** 2018-04
- **Owl (1.14.0)** 2018-08
- **Peacock (1.15.0)** 2018-12 (*latest release*)

Deployments

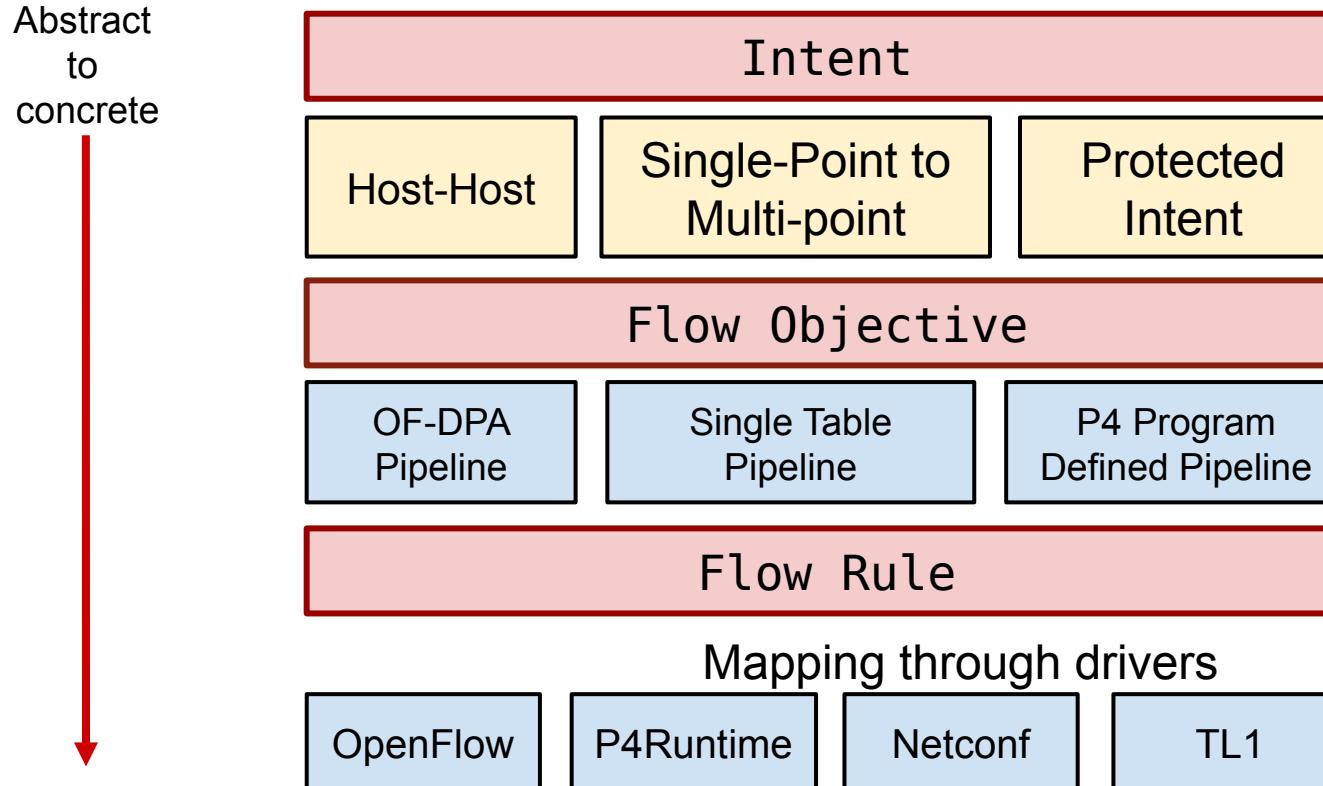
- **Access network for residential customers**
 - In production with a major US telecom provider
 - Edge leaf-spine fabric: ~10K clients, ~150K routes, ~1.5M flows
 - OpenFlow
- **Research & Education**
 - SDN-IP, VPLS apps
- **SDN in Air-Traffic Management**
 - Safety-critical, ATM-grade deployment in Brazil (~22M km²)
 - Radar relays, remote control towers, pilot voice, etc.
 - NetBroker from Frequentis developed on ONOS
 - Brown-field & OpenFlow



ONOS architecture

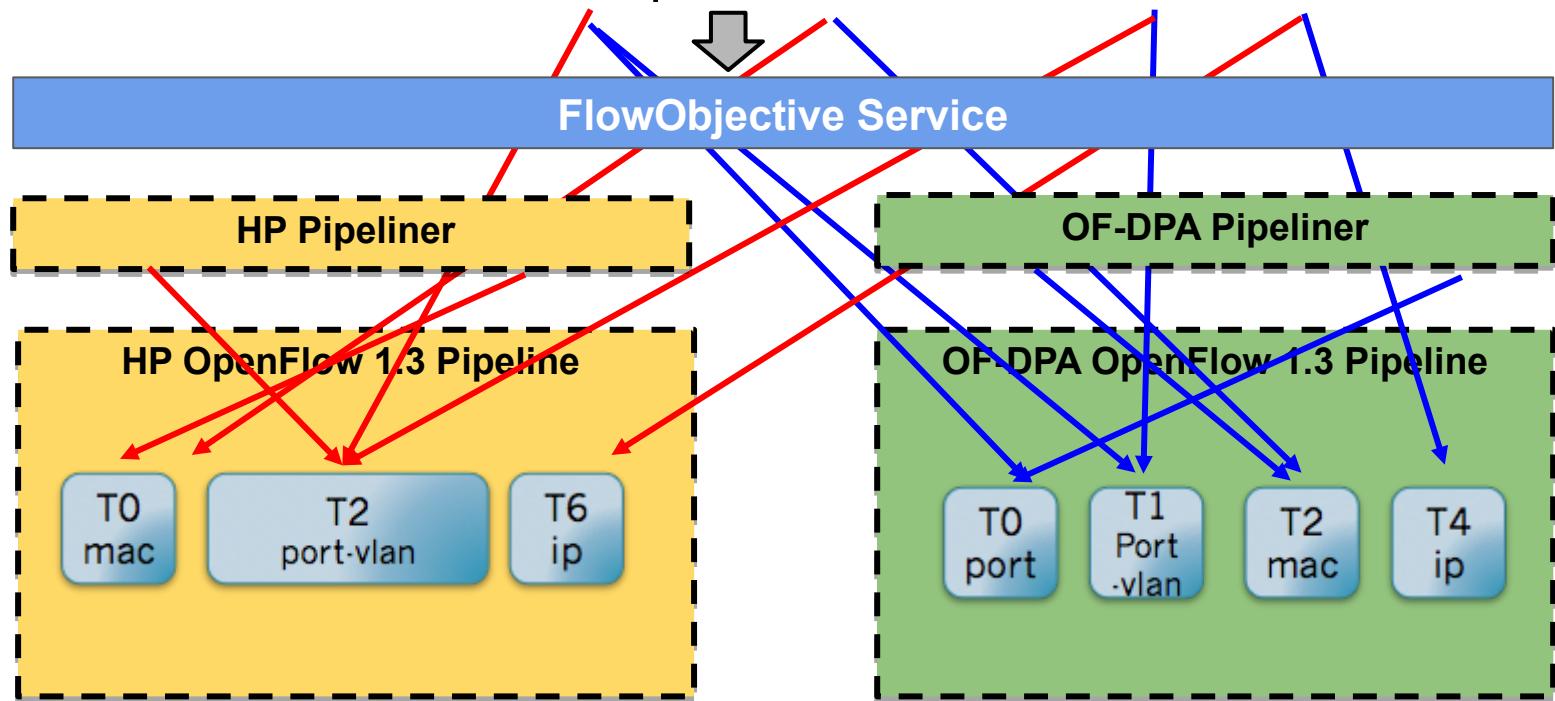


Network programming API



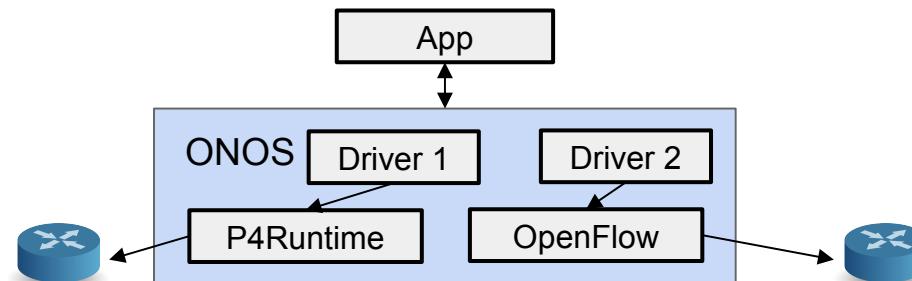
Flow objective example

Peering Router Match on Switch port, MAC address, VLAN, IP



Driver behaviors in ONOS

- ONOS defines APIs to interact with device called “behaviors”
 - [DeviceDescriptionDiscovery](#) → Read device information and ports
 - [FlowRuleProgrammable](#) → Write/read flow rules
 - [PortStatisticsDiscovery](#) → Statistics of device ports (e.g. packet/byte counters)
 - [Pipeliner](#) → FlowObjective-to-FlowRules mapping logic
 - Etc.
- Behavior = Java interface
- Driver = collection of one or more behavior implementations
 - Implementations use ONOS protocol libraries to interact with device



ONOS key takeaways

- **Apps are independent from switch control protocols**
 - Same app can work with OpenFlow and P4Runtime devices
- **Different network programming APIs**
 - FlowRule API – pipeline-dependent
 - FlowObjective API – pipeline-independent
 - Drivers translate 1 FlowObjective to many FlowRule
- **FlowObjective API enables application portability**
 - App using FlowObjectives can work with switches with different pipelines
 - For example, switches with different P4 programs

P4 and P4Runtime support in ONOS

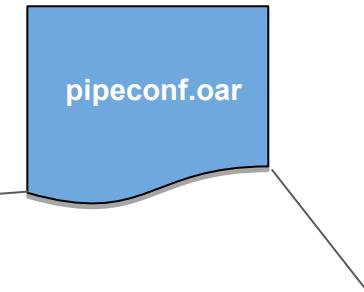
P4 and P4Runtime support in ONOS

Goals:

1. Allow ONOS users to bring their own P4 program
2. Allow *existing* apps to control *any* P4 pipeline without changing the app
 - i.e. enable app portability across many P4 pipelines
3. Allow apps to control custom/new protocols as defined in the P4 program

Pipeconf - Bring your own pipeline!

- Package together everything necessary to let ONOS understand, control, and deploy an arbitrary pipeline
- Provided to ONOS as an app
 - Can use .oar binary format for distribution



1. Pipeline model

- Description of the pipeline understood by ONOS
- Automatically derived from P4Info

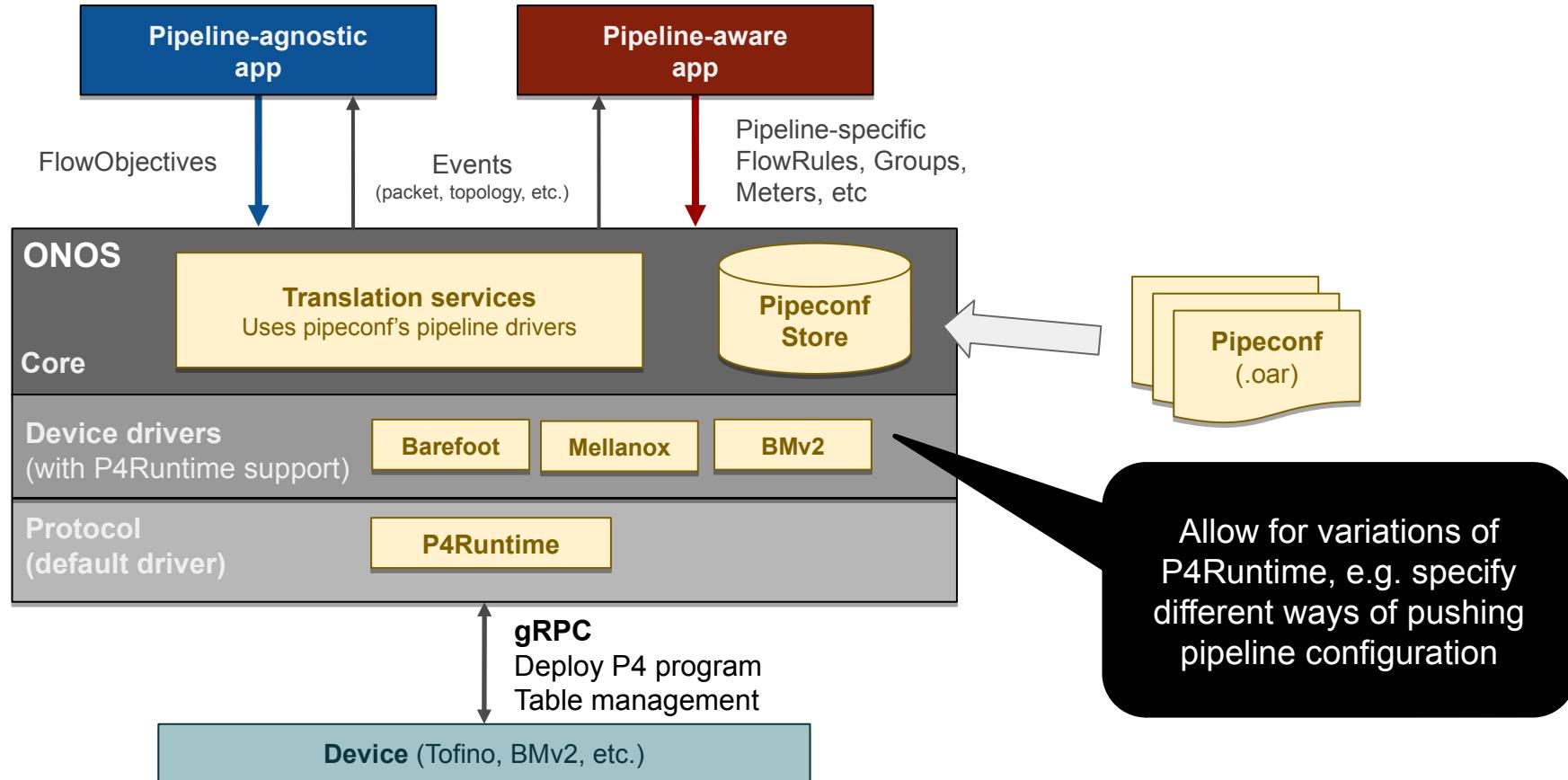
2. Target-specific binaries to deploy pipeline to device

- E.g. BMv2 JSON, Tofino binary, FPGA bitstream, etc.

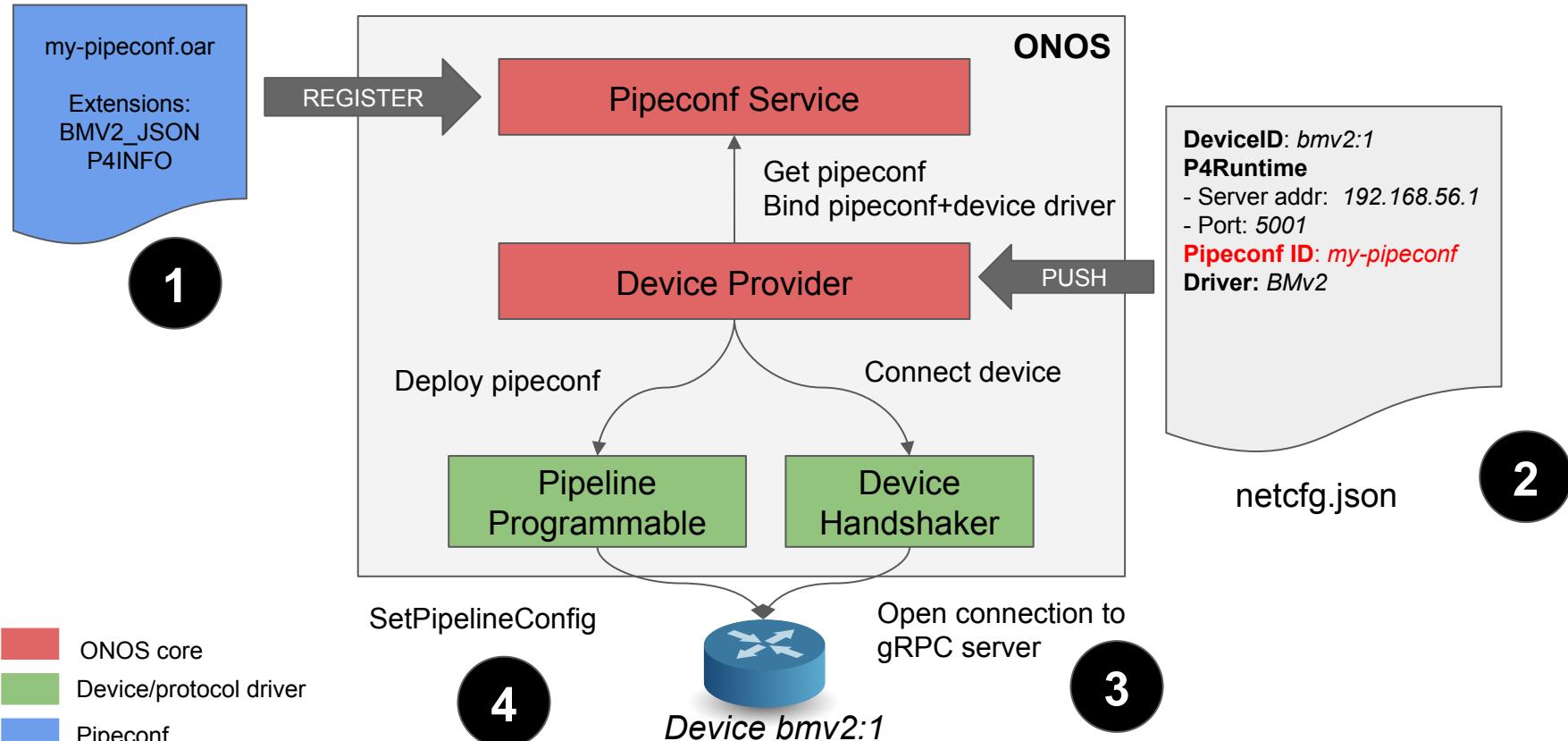
3. Pipeline-specific driver behaviors

- E.g. “Pipeliner” implementation: logic to map FlowObjectives to P4 pipeline

Pipeconf support in ONOS



Pipeconf deploy and device discovery



Flow operations

48

Define flow rules using same headers/action names as in the P4 program. E.g match on "hdr.my_protocol.my_field"

Pipeconf-based 3 phase translation:

1. Flow Objective → Flow Rule

- Maps 1 flow objective to many flow rules

2. Flow Rule → Table entry

- Maps standard headers/actions to P4-defined ones
E.g. ETH_DST→“hdr.ethernet.dst_addr”

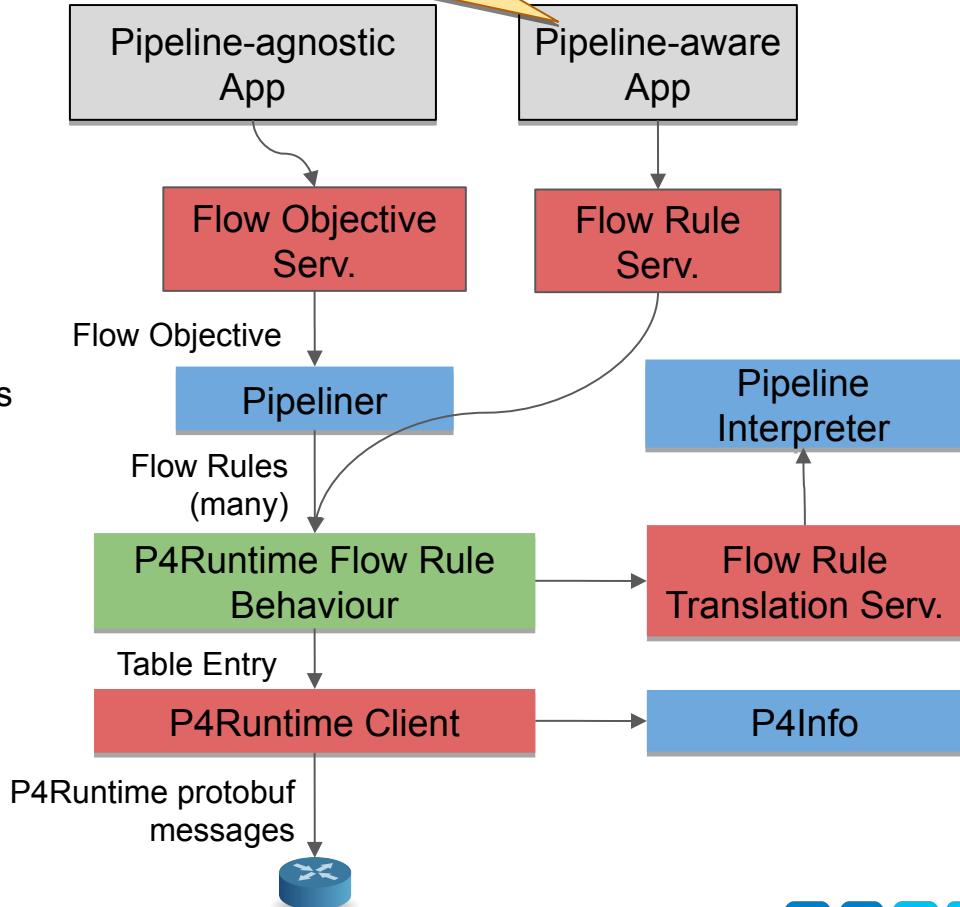
3. Table Entry → P4Runtime message

- Maps P4 names to P4Info numeric IDs

 ONOS Core

 Device/protocol driver

 Pipeconf driver



Pipeline Interpreter

- **Driver behavior**
 - Provides mapping between ONOS well-known headers/actions and P4 program-specific entities
- **Example: flow rule mapping**
 - Match
 - 1:1 mapping between ONOS known headers and P4 header names
 - E.g. ETH_DST → “ethernet.dst_addr” (name defined in P4 program)
 - Action
 - ONOS defines standard actions as in OpenFlow (output, set field, etc)
 - Problem: P4 allows only one action per table entry, ONOS many (as in OpenFlow)
 - E.g. header rewrite + output: 2 actions in ONOS, 1 action with 2 parameters in P4
 - How to map many actions to one? Need interpretation logic (i.e. Java code)!

P4Runtime support in ONOS 1.14 (Owl)

P4Runtime control entity	ONOS API
Table entry	Flow Rule Service , Flow Objective Service Intent Service
Packet-in/out	Packet Service
Action profile group/members, PRE multicast groups	Group Service
Meter	Meter Service (indirect meters only)
Counters	Flow Rule Service (direct counters) P4Runtime Client (indirect counters)
Pipeline Config	Pipeconf

Unsupported features - community help needed!

Parser value sets, registers, digests, clone sessions

ONOS+P4 workflow recap

- **Write P4 program and compile it**
 - Obtain P4Info and target-specific binaries to deploy on device
- **Create pipeconf**
 - Implement pipeline-specific driver behaviours (Java):
 - Pipeliner (optional - if you need FlowObjective mapping)
 - Pipeline Interpreter (to map ONOS known headers/actions to P4 program ones)
 - Other driver behaviors that depend on pipeline
- **Use existing pipeline-agnostic apps**
 - Apps that program the network using FlowObjectives
- **Or... write new pipeline-aware apps**
 - Apps can use same string names of tables, headers, and actions as in the P4 program
- **Enjoy!**