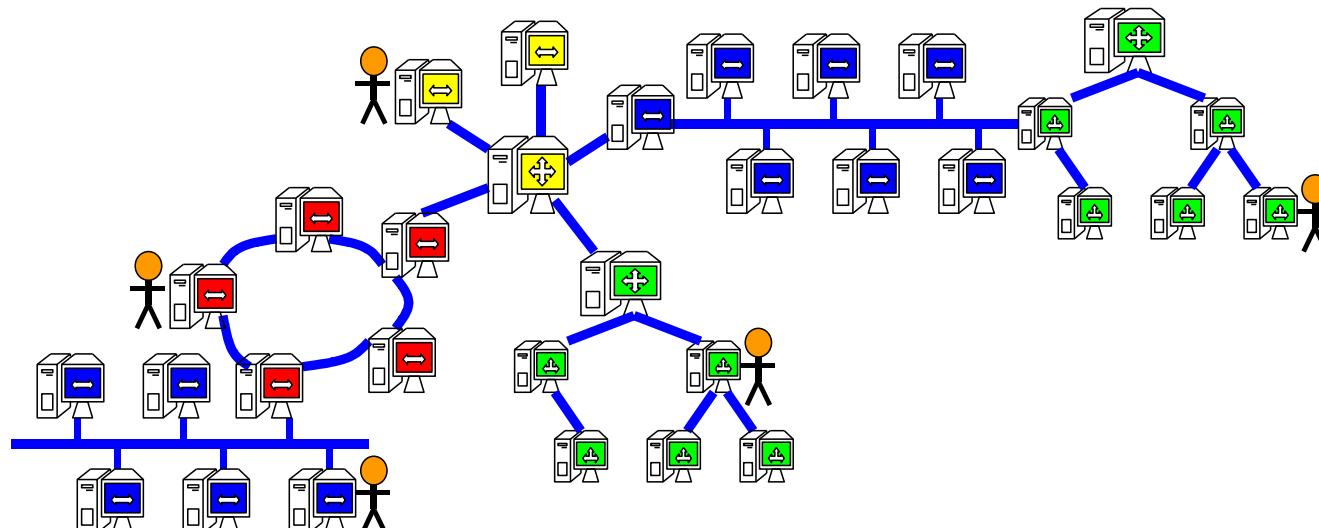


Cos'è una rete di calcolatori?

- Definizione: insieme di dispositivi di calcolo, **autonomi e interconnessi**
- Motivazioni:
 - supporto alla comunicazione tra utenti
 - es. nuovi servizi di comunicazione (es. e-mail, WWW)
 - comunicazione tra calcolatori elettronici
 - Condivisione di informazione (es. pagine web, basi di dati)
 - Condivisione di dispositivi e risorse (es. stampanti, supporti di memorizzazione)
 - Accesso a calcolatori remoti
 - Calcolo distribuito, sistemi scalabili



Una generica rete di calcolatori con dispositivi autonomi e utenti.

L'immagine mostra una rete composta da svariati dispositivi, ognuno dei quali è autonomo e connesso ad altri dispositivi, fino a formare un'unica rete di calcolatori. Esistono alcuni utenti che stanno comunicando, attraverso i rispettivi calcolatori, interconnessi attraverso il supporto fisico della rete di comunicazione.

Una rete di calcolatori è un insieme di dispositivi autonomi, cioè in grado di eseguire e svolgere autonomamente i compiti programmati di calcolo e di comunicazione, interconnessi tra loro da supporti fisici alla trasmissione di segnali.

Non sono considerate reti di calcolatori, ad esempio, né le reti di comunicazione telefonica (i cui terminali telefonici non sono dispositivi autonomi), né le reti di distribuzione televisiva (in quanto i televisori non sono dispositivi autonomi in grado di comunicare informazione).

Nel prosieguo della presentazione, con il generico termine di "rete" o "rete di comunicazione" intenderemo implicitamente solo le reti di calcolatori elettronici.

Con l'avvento dei calcolatori elettronici, e con la loro diffusione tra comunità sempre più grandi di utenti, è emersa l'esigenza e l'utilità di fornire un supporto alla comunicazione tra utenti, attraverso l'uso del calcolatore, supportando innovativi servizi di comunicazione per l'utente, quali ad esempio il World Wide Web e la posta elettronica. In tempi più recenti si sono sviluppati ulteriormente i sistemi di rete includendo Internet of Things (IoT), reti senza fili (Wireless), ecc.

Le necessità di comunicare e condividere informazione sono tra i principali motivi che favoriscono la nascita e lo sviluppo di reti di calcolatori. La fruizione dell'informazione contenuta in questo corso rappresenta un esempio.

Un ulteriore aspetto che ha favorito la nascita e la diffusione delle reti di calcolatori è legato alla possibilità di condividere dispositivi costosi, altrimenti sotto-utilizzati, come ad esempio stampanti o capienti dispositivi di memorizzazione dei dati, e la possibilità di accedere e lavorare sui dati di un calcolatore, senza doversi spostare fisicamente sul calcolatore stesso.

Una rete di calcolatori può consentire di eseguire calcoli complessi in parallelo e in maniera distribuita, aumentando le prestazioni per l'ottenimento dei risultati. In tal senso, le reti rendono possibile la scalabilità dei sistemi di comunicazione e di calcolo: il numero di dispositivi usati, e l'investimento relativo, possono essere dimensionati dinamicamente in funzione delle richieste di servizio. In tempi recenti, le reti sono utilizzate in particolare per supportare la comunicazione utente, secondo svariate forme e applicazioni, oppure per supportare la comunicazione diretta tra dispositivi pervasivi e mobili (es. Internet of Things, Wireless Networks), ecc.

Classificazione delle Reti

Le reti di calcolatori si possono classificare in base alla dimensione geografica:

- Reti personali (Personal Area Network), PAN
 - Connessione tra dispositivi in una stanza, o di una scrivania
- Reti locali (Local Area Network), LAN
 - Connessione dispositivi di un ufficio, un laboratorio, un edificio, un campus
- Reti metropolitane (Metropolitan Area Network), MAN
 - Connessione di aree urbane, reti civiche
- Reti geografiche (Wide Area Network), WAN
 - Connessione di aree molto ampie, reti nazionali e internazionali
- Internet: è la rete globale composta dall'unione di reti di vari tipi, connesse tra loro e conformi a un determinato insieme di regole di comunicazione comuni (i protocolli di Internet).

Una prima classificazione delle reti di calcolatori si basa sulla dimensione delle reti stesse.

Non esiste in generale un criterio ben definito per tale classificazione, ma ci si basa su considerazioni generali, riguardanti la dimensione dell'area di copertura geografica della rete, ovvero l'area entro la quale possano esistere dispositivi connessi.

Le reti personali (PAN) sono reti di comunicazione per connettere dispositivi vicini tra loro, ad esempio sul corpo di una persona o entro una stanza.

Un esempio potrebbe essere dato dalla connessione di due dispositivi indossabili, sensori e smartphone, oppure calcolatori, una stampante e un agenda elettronica. Le reti personali sono di solito finanziate e gestite dal singolo utente che le utilizza.

Le reti locali (LAN) sono molto spesso reti gestite e mantenute da organizzazioni, università, enti o aziende.

Esse connettono calcolatori nel raggio di qualche centinaio di metri, ad esempio su interi edifici o campus universitari. Ad esempio, la rete delle aule del Dipartimento.

Le reti metropolitane hanno connessioni in un raggio dell'ordine delle decine di chilometri, e possono connettere intere aree urbane. Esse sono mantenute e gestite da fornitori di servizi di comunicazione (provider) e gestori di servizi telefonici.

Le reti geografiche sono reti in grado di coprire distanze internazionali e addirittura planetarie. Tali reti sono mantenute e gestite da enti nazionali e internazionali, oppure da grossi enti o gestori delle comunicazioni. L'organizzazione e la struttura di tali reti può essere molto complessa, e può risultare composta da diverse parti, e da diverse tecnologie, eterogenee e integrate (ad esempio, molte reti collegate tra loro con tecnologie cablate o in fibra, fino a reti basate su comunicazione satellitare senza fili).

Internet è una rete di reti, composta da molte reti diverse connesse tra loro, integrate grazie a un insieme di regole comuni: i protocolli della rete Internet.

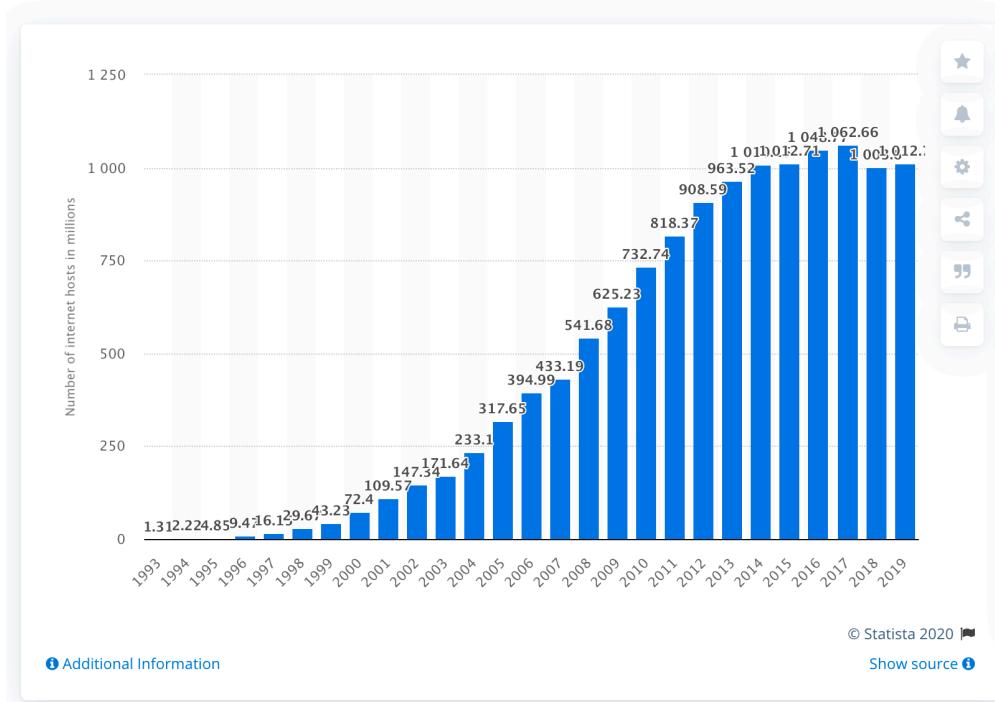
Reti di calcolatori: evoluzione e costi

Lo sviluppo delle reti di calcolatori ha permesso la nascita di Internet, resa possibile da una distribuzione dei costi di realizzazione e gestione delle infrastrutture tra molte entità.

- storicamente la prima rete di [Internet](#) nasce da un esperimento nel 1969, connettendo solo 4 calcolatori di 4 università americane.
- da allora molte entità hanno dato il loro contributo per lo sviluppo e la diffusione delle reti
- all'inizio del 2003 Internet ha oltre 172 milioni di calcolatori connessi (fonte: [ISC](#))
- oggi (2020) si stima siano oltre 4 Miliardi i dispositivi connessi (dotati di indirizzo IP) e oltre 25 miliardi considerando i dispositivi dell'Internet of Things
- Entro il 2022-2025 si pensa a oltre 60 miliardi di dispositivi connessi (Internet of Things)
- costo di realizzazione, mantenimento, gestione e uso: chi paga e mantiene le reti?
 - di norma le tecnologie e infrastrutture a prestazioni più elevate costano di più
 - esistono molte piccole reti: costi e gestione limitati e distribuiti su piccola scala (locale)
 - consorzi e gruppi multinazionali di fornitori di servizi di comunicazione: costi e gestione su larga scala (globale)
 - integrazione di reti: reti di reti
- costo per l'utente: tariffe a tempo, a quantità di dati, oppure tariffe "tutto incluso"

Reti di calcolatori - Intro

(in millions)



credits: statista.com

Una curva che mostra il numero di Host connessi a Internet dotati di nome logico nei servizi DNS.
L'immagine mostra una curva del numero di host connessi a Internet dotati di nome logico nei servizi DNS, crescente in modo esponenziale, che tende a stabilizzarsi nel 2017. In realtà il numero di dispositivi connessi sta ancora crescendo esponenzialmente, ma da questo indice non lo si percepisce.

Lo sviluppo delle reti di calcolatori, che ha permesso la nascita di Internet, non sarebbe stato possibile senza una distribuzione dei costi di realizzazione e gestione delle infrastrutture tra molte entità.

Storicamente, la prima rete di Internet nasce da un esperimento nel 1969, connettendo solo 4 calcolatori di 4 università americane. Da allora molte entità hanno dato il loro contributo per lo sviluppo e la diffusione delle reti di calcolatori.

All'inizio del 2003 Internet contava oltre 172 milioni di calcolatori (fonte [Internet Software Consortium](#)). Già nel 2017 si parla di oltre 4 miliardi di dispositivi connessi (anche se non tutti connessi allo stesso momento). Entro 5-7 anni si raggiungeranno i 60 miliardi di dispositivi connessi, realizzando l'avvento dell'Internet of Things.

Per quello che riguarda i costi, la realizzazione, mantenimento e gestione dell'infrastruttura di una rete molto ampia richiede investimenti economici elevati, che possono essere maggiori a seconda del grado di avanzamento delle tecnologie e delle prestazioni richieste. Alcune delle infrastrutture principali delle reti estese MAN e WAN (e di Internet) hanno costi affrontabili solo attraverso un consistente investimento e una pianificazione delle ricadute commerciali da parte di consorzi o fornitori di servizi di comunicazione nazionali e multinazionali. Tuttavia, la maggior percentuale del complesso delle infrastrutture di rete che compongono Internet risultano essere mantenute e gestite capillarmente da piccoli gestori e piccoli gruppi, con investimenti relativamente modesti per la realizzazione di piccole reti locali (LAN). L'integrazione di un insieme molto vasto di reti grandi e soprattutto piccole reti locali, eterogenee e distribuite su tutto il pianeta, ha permesso la crescita incrementale, il successo commerciale e la esplosiva diffusione delle reti su scala globale, fino a Internet. L'utente delle reti paga tipicamente per i servizi di trasmissione offerti dalle reti, con tariffe che possono essere basate sul tempo di collegamento, sulla quantità di dati

Reti di calcolatori: prestazioni

- Uno degli aspetti di interesse per gli utenti delle reti riguarda le prestazioni: come si misurano?
 - capacità di trasmissione: numero di bit o byte trasmessi o ricevuti in un secondo
 - i dati si misurano in bit o in byte, cioè gruppi di 8 bit
 - K=Kilo (migliaia), M=Mega (milioni), G=Giga (miliardi), T=Tera (1000 G)
 - es. 10 Mbyte/sec = circa 10 milioni di gruppi da 8 bit al secondo
 - migliori prestazioni sono determinate da migliori tecnologie (a costi elevati).
 - ritardo del collegamento: tempo richiesto ai dati per transitare da mittente a destinatario
 - il ritardo è determinato dalla distanza fisica, ma non solo
 - il ritardo può essere determinato in misura rilevante anche dai tempi di gestione dovuti alle leggi dei processi di comunicazione (protocolli)

Per ciò che riguarda le prestazioni delle reti di calcolatori, l'utente è principalmente interessato a due indici: la capacità di trasmissione (impropriamente detta velocità della rete) e il ritardo del collegamento di rete.

La capacità di trasmissione si misura sulla base della quantità di dati che è possibile comunicare in un secondo mediante la rete. I dati digitali del calcolatore si misurano in bit o in byte (gruppi di 8 bit), e di conseguenza l'unità di misura usata tipicamente per misurare la capacità di trasmissione dei dati di una rete è il numero di bit oppure di byte trasmessi al secondo (bit/sec, byte/sec). Spesso si usano i prefissi Kilo (K) per le migliaia, Mega (M) per i milioni e Giga (G) per i miliardi di bit o byte al secondo, Tera (T) per le migliaia di miliardi, ecc. (esempio Kbit/sec, Kbyte/sec).

Il ritardo del collegamento di rete indica il tempo necessario ai dati per transitare dal mittente al destinatario finale sulla rete.

Il ritardo è determinato da vari fattori, tra i quali la distanza fisica del collegamento, i tempi necessari alla gestione delle regole dei processi di comunicazione in rete (protocolli) che i dati devono subire durante il loro tragitto.

Ovviamente sono da preferire reti dotate di basso ritardo, in quanto ciò favorisce la rapidità e l'interattività del processo di comunicazione.

Per fare un parallelo intuitivo, pensando alle reti come a tubi che trasportano bit, la capacità di trasmissione equivale al diametro del tubo, mentre il ritardo equivale al tempo che i bit impiegano ad attraversare una serie di tubi in tutta la loro lunghezza.

Il calcolatore e la rete: componenti

La connessione di un calcolatore a una rete di calcolatori richiede un insieme minimo di componenti, sia hardware che software, in aggiunta al calcolatore elettronico di base:

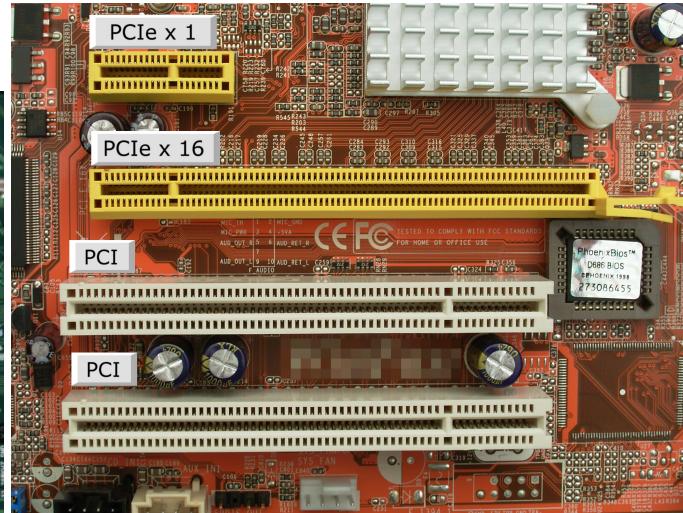
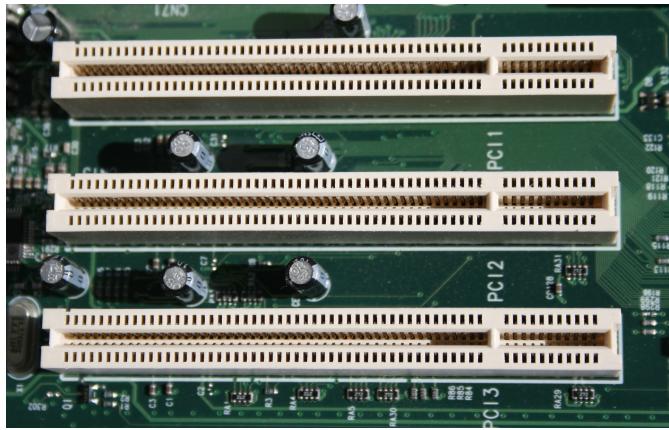
- **dispositivi o schede di rete**: sono dispositivi hardware per codificare, trasmettere, ricevere e decodificare dati dal calcolatore alla rete, e viceversa
 - amministrati da componenti software del sistema operativo
- **mezzo di trasmissione**: supporto fisico alla propagazione e trasmissione di segnali: es. cavi elettrici, fibre ottiche. Esso realizza l'infrastruttura fisica di rete.
- **connettore di rete**: interfaccia o connettore standard per il collegamento del dispositivo di rete al mezzo di trasmissione della rete
- **protocolli di rete**: insieme di regole implementate sotto forma di software del calcolatore, univocamente definite per garantire compatibilità, e corretta gestione della comunicazione



Reti di calcolatori - Intro



Reti di calcolatori - Intro



Schema delle componenti necessarie alla connessione di un calcolatore a una rete di calcolatori. Nella seconda immagine, una scheda Ethernet per connessione a bus Peripheral Component Interconnect (PCI), e una scheda di rete wireless 802.11n con due antenne e connessione PCI express (PCIe).

La figura mostra un calcolatore, sul quale è installato il software dei protocolli di rete, nel quale viene inserita una scheda o dispositivo di rete, dotata a sua volta di un connettore di rete. Al connettore di rete viene collegato il mezzo di trasmissione utilizzato per la rete di calcolatori.

La connessione di un calcolatore a una rete di calcolatori richiede un insieme essenziale di componenti, hardware e software, in aggiunta al calcolatore elettronico di base.

L'elemento primario da aggiungere al calcolatore è il dispositivo (o scheda) di rete: si tratta di un dispositivo hardware di comunicazione, fisicamente collegato al calcolatore, in grado di codificare e trasmettere, oppure ricevere e decodificare i dati inviati dal calcolatore alla rete, e dalla rete al calcolatore.

I mezzi di trasmissione sono supporti fisici alla propagazione e trasmissione di segnali, quali cavi o fili elettrici, fibre ottiche, o semplicemente lo spazio tridimensionale nel quale si propagano le onde radio. Tali mezzi di trasmissione realizzano l'infrastruttura fisica della rete. Il costo di realizzazione dell'infrastruttura di rete rappresenta spesso un fattore rilevante e critico per la diffusione e l'implementazione di reti di calcolatori.

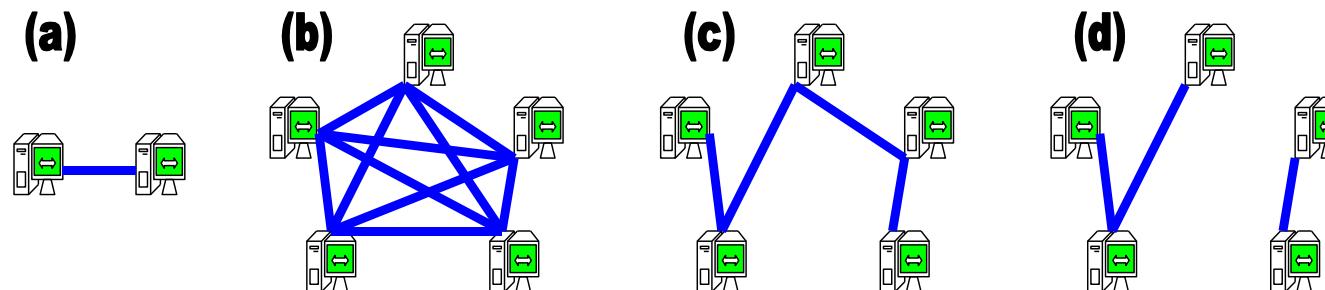
Il connettore di rete è semplicemente un'interfaccia standard presente sul dispositivo di rete, per il collegamento del dispositivo di rete al mezzo di trasmissione. Esistono vari connettori, diversi a seconda del tipo di tecnologia impiegata per la rete di comunicazione. I connettori possono avere varie forme, e tipicamente permettono il collegamento solo quando le tecnologie dei dispositivi di rete, dei protocolli di gestione, e dei mezzi di trasmissione sono tra loro compatibili.

I dispositivi di rete sono amministrati da componenti software del sistema operativo, e devono rispettare un insieme di regole standard per la gestione dei processi di comunicazione, definite dai protocolli di rete.

I protocolli di rete sono un insieme di regole, univocamente definite, per garantire la compatibilità e la corretta configurazione e gestione delle fasi della comunicazione tra i dispositivi di rete. Essi saranno trattati in maggior dettaglio successivamente.

Collegamenti e infrastrutture di rete

- **Connessione o collegamento di rete:**
 - un mezzo di trasmissione condiviso tra calcolatori in rete (nodi, host)
 - Il mezzo di trasmissione supporta fisicamente la trasmissione di segnali
- **Infrastruttura di rete:** la struttura di connessione dei collegamenti della rete
 - Punto a punto (a)
 - strutture di rete a connessioni multiple
 - Completamente connesse (b), cammini ridondanti
 - parzialmente connesse (c)
 - partizioni di rete (d): gruppo di componenti isolati da altri
- **Cammino dei segnali:** diretto oppure indiretto attraverso connessioni in sequenza



Alcune classi di infrastrutture di rete

La figura illustra quattro casi relativi a possibili classi di infrastrutture di connessione di rete. Connessione punto a punto, con due soli dispositivi connessi direttamente. Connessione multipla completamente connessa, con cinque dispositivi tutti connessi direttamente tra loro. Connessione multipla parzialmente connessa, con cinque dispositivi connessi tra loro non completamente, ma in modo da garantire l'esistenza di un cammino per trasferire i segnali scambiati da ogni possibile coppia della rete. Connessione multipla con partizione di rete, nella quale due gruppi da tre e da due nodi sono isolati tra loro, anche se connessi internamente.

Reti di calcolatori - Intro

Un collegamento o connessione fisica di rete è fornita da un mezzo di trasmissione (ad esempio un cavo, una fibra ottica oppure lo spazio per la propagazione di onde radio) che sia condiviso tra due o più dispositivi ad esso collegati, e che permetta il trasferimento di segnali, e quindi informazione, tra i dispositivi stessi.

Un'infrastruttura di rete rappresenta l'insieme dei collegamenti o connessioni fisiche esistenti tra tutti i dispositivi di una rete.

La comunicazione tra una coppia qualsiasi di calcolatori in rete, detti nodi (oppure host) della rete, è possibile se esiste un collegamento diretto tra i nodi, oppure se esiste una sequenza di collegamenti, detta cammino, che permetta la comunicazione dei segnali passando per eventuali nodi e collegamenti intermedi.

Sono possibili diverse classi di strutture di connessione della rete.

Le connessioni di rete punto a punto, come nell'esempio (a), sono connessioni che possono essere instaurate tra una coppia di calcolatori, senza coinvolgerne altri. Esse rappresentano il caso più semplice di infrastruttura di rete, e sono semplici da gestire.

Le connessioni di rete multiple permettono di connettere contemporaneamente un dispositivo a molti altri dispositivi.

Nell'esempio (b) viene mostrata una infrastruttura di rete nella quale ogni nodo è connesso attraverso un linea dedicata ad ogni altro nodo. Questa infrastruttura di rete viene detta completamente connessa, ed è molto ridondante: infatti esistono molti cammini, oltre al collegamento diretto, per connettere ogni coppia di nodi passando per nodi intermedi.

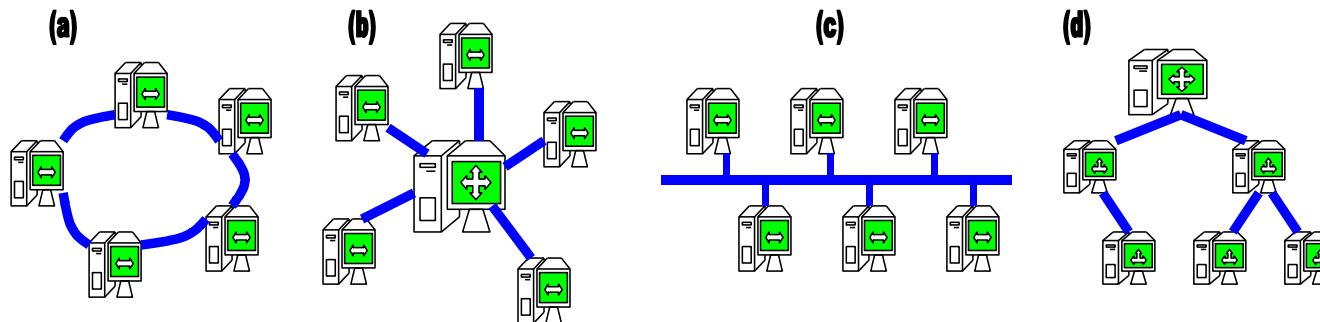
Una simile infrastruttura di rete si può ritenere a volte troppo complessa e costosa. Ad esempio, sarebbe impensabile disporre di una connessione dedicata (cioè un filo diretto) da ogni calcolatore ad ogni altro calcolatore sulle reti mondiali.

Nell'esempio c, malgrado il ridotto numero di collegamenti rispetto al caso b, è comunque possibile per ogni dispositivo trasferire segnali, cioè comunicare, verso ogni altro dispositivo. In altre parole esiste un cammino, attraverso le connessioni disponibili, per trasferire informazione tra ogni coppia di dispositivi della rete. Nella rete (c) esiste però un fattore di rischio: in seguito a un guasto di una connessione, potrebbe risultare un insieme di componenti separato da tutti gli altri, detto "partizione" della rete (esempio d). Le partizioni della rete limitano il grado di comunicazione possibile, e possono essere dovute a cause fisiche (guasti fisici della connessione) oppure a cause che dipendono da cattive applicazioni delle regole di utilizzo (ovvero dei protocolli, che vedremo in seguito).

Topologia di rete

Le infrastrutture di rete possono assumere vari schemi di connessione dei dispositivi (**topologia**)

- esempi di topologie:
 - (a) Anello (ring)
 - (b) Stella (star)
 - (c) Bus
 - (d) Albero (tree)
- In generale le reti più piccole (reti locali LAN e PAN) rispettano le topologie citate
- Reti più grandi e complesse possono assumere topologie ibride, dette a maglia
 - reti con connessioni multiple riducono il rischio di **partizioni** della rete



Quattro esempi delle principali topologie per infrastrutture di rete

La figura mostra quattro esempi relativi alle principali topologie di infrastrutture di rete. Nello scenario (a) vediamo 5 elementi connessi con topologia ad anello: ogni elemento è connesso solo con l'elemento precedente e successivo. Nello scenario (b) vediamo 6 elementi connessi a stella: ogni elemento periferico ha una connessione (bi-direzionale) verso l'unico elemento centrale. L'elemento centrale è l'unico elemento completamente connesso a tutti gli altri. Nello scenario (c) vediamo 6 elementi connessi con tipologia a bus: ognuno degli elementi ha una connessione verso il bus centrale, condiviso da tutti. Nello scenario (d) vediamo 6 elementi connessi con tipologia ad albero: se pensiamo all'analogia con un albero genealogico, esiste un nodo (nonno) che connette direttamente due nodi (figli). Il figlio di sinistra connette un elemento nipote, mentre il figlio di destra connette due elementi nipoti.

Diversi schemi di connessione sono possibili per creare le infrastrutture di rete: tali schemi si dicono topologie della rete.

Tra gli schemi possibili, la topologia ad anello (esempio a) è basata sull'organizzazione delle connessioni tra i dispositivi, in modo da creare un anello chiuso. Ogni componente può comunicare con ogni altro componente inviando i segnali attraverso la sequenza di connessioni in senso orario o antiorario.

La topologia a stella (esempio b) prevede un componente centrale direttamente connesso a tutti gli altri. Ogni componente periferico può comunicare con ogni altro componente periferico passando attraverso il componente centrale.

La topologia a bus (esempio c) prevede che ogni componente abbia una connessione verso un bus condiviso (cioè una connessione condivisa da tutti). Questo tipo di connessione permette di introdurre una delle problematiche fondamentali che saranno trattate in seguito: la gestione dell'accesso al bus, ovvero il decidere chi possa trasmettere tra tutti i possibili dispositivi, per evitare sovrapposizioni delle trasmissioni.

La topologia ad albero (esempio d) prevede un'organizzazione gerarchica delle connessioni.

Se pensiamo all'analogia con un albero genealogico, esiste un dispositivo (nonno) che connette direttamente due o più dispositivi (figli), ognuno dei quali a sua volta connette direttamente un numero variabile di dispositivi (nipoti), e così via.

In generale, le topologie suddette vengono di norma adottate solo nel contesto di reti di comunicazione molto piccole, come le reti personali PAN e locali LAN. I motivi per cui ciò accade saranno illustrati in seguito, quando si parlerà dei problemi di condivisione del mezzo di trasmissione.

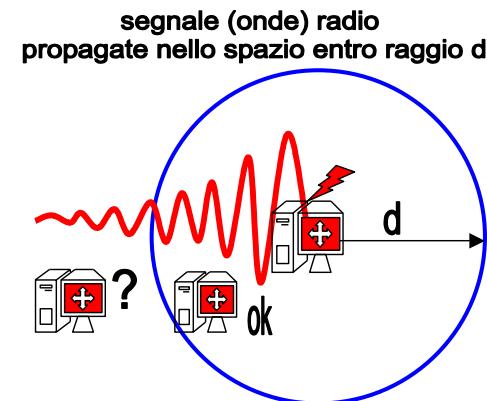
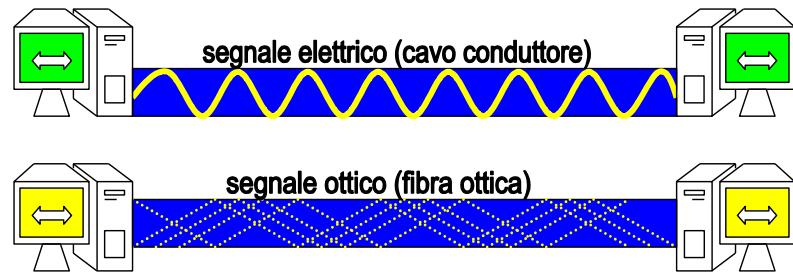
Mano a mano che le reti più piccole vengono collegate tra loro e organizzate in strutture di rete più grandi, la topologia della rete globale può diventare incredibilmente complessa, e quindi uno schema topologico generalizzato non è quasi mai applicabile. In questo caso si parla di rete con topologia a grafo complesso, oppure a maglia. In tali topologie a grafo, possono essere presenti cammini multipli che connettono coppie di nodi, dando luogo a possibili alternative per la connessione dei dispositivi.

Questo fatto può ridurre il rischio di incorrere in partizioni della rete, in quanto un certo grado di ridondanza dei cammini di connessione permette di aggirare i collegamenti soggetti a eventuali guasti.

Il mezzo fisico di trasmissione

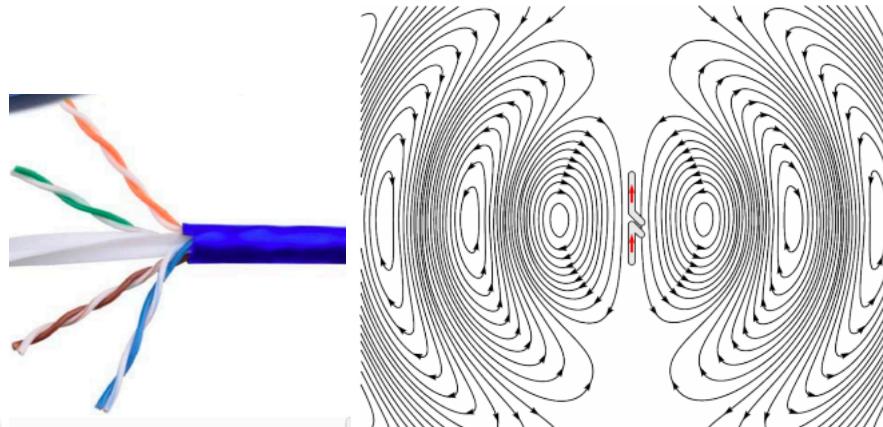
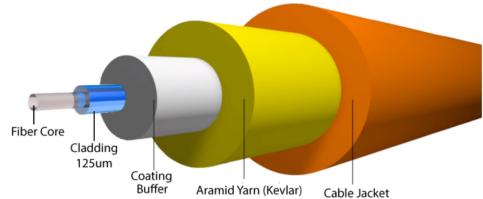
Il **mezzo di trasmissione** usato per la connessione fisica dei calcolatori nelle reti, e il relativo segnale usato per le trasmissioni, possono essere di tre tipi:

- **cavetti o fili metallici**: trasmettono segnali elettrici (corrente e variazione di tensione)
 - ad esempio: doppino intrecciato, cavo coassiale
 - metodo più usato, affidabile, buon rapporto costo/prestazioni (fino a 1-2 Gbit/sec)
- **Fibre ottiche**: trasmissione segnali luminosi vincolati entro fibre di vetro
 - offre le migliori prestazioni di capacità della rete: varie migliaia di Gbit/sec
 - infrastruttura di rete e collegamento (giunzione) della fibra è costoso
- **senza fili (wireless)**: radiazione elettromagnetica nello spazio (anche nel vuoto)
 - onde radio (frequenze radio), raggi infrarossi (frequenze luce infrarossa)
 - permettono mobilità dei calcolatori, e riducono le infrastrutture di rete fisse
 - collegamento non sempre affidabile, e capacità (oggi) da 1 a 54 Mbit/sec
 - distanza limitata



Reti di calcolatori - Intro

SEZIONE DEL CAVO FIBRA OTTICA



https://it.wikipedia.org/wiki/Onde_radio#/media/File:Dipole_xmtng_antenna_animation_4_408x318x150ms.gif

Tre esempi di mezzo trasmissivo per la connessione di rete: cavo conduttore, fibra ottica e onde radio
Due calcolatori verdi sono connessi attraverso un cavo conduttore entro il quale passa un segnale elettrico sinusoidale. Due calcolatori gialli sono connessi attraverso una fibra ottica entro la quale passano fotoni di luce. Due calcolatori rossi sono connessi attraverso onde radio, in quanto si trovano entro la distanza massima di propagazione del segnale radio (d). Un terzo calcolatore rosso è troppo distante e risulta quindi non connesso.

Il mezzo di trasmissione è l'elemento fisico che supporta la propagazione dei segnali trasmessi tra i dispositivi della rete.

Le connessioni di rete possono essere realizzate mediante tre mezzi di trasmissione diversi: cavi conduttori, fibre ottiche e connessioni senza fili.

I cavi di materiale conduttore (cavetti, doppino intrecciato o cavo coassiale), sono in grado di propagare segnali elettrici, cioè variazioni di tensione e corrente elettrica. Questi mezzi fisici sono i più utilizzati nelle reti locali, e nelle brevi distanze, per il loro buon rapporto tra costo e prestazioni. Oggi tale mezzo trasmissivo è in grado di supportare trasmissioni dati con una capacità dell'ordine del miliardo di bit al secondo (1-2 Gbit/sec).

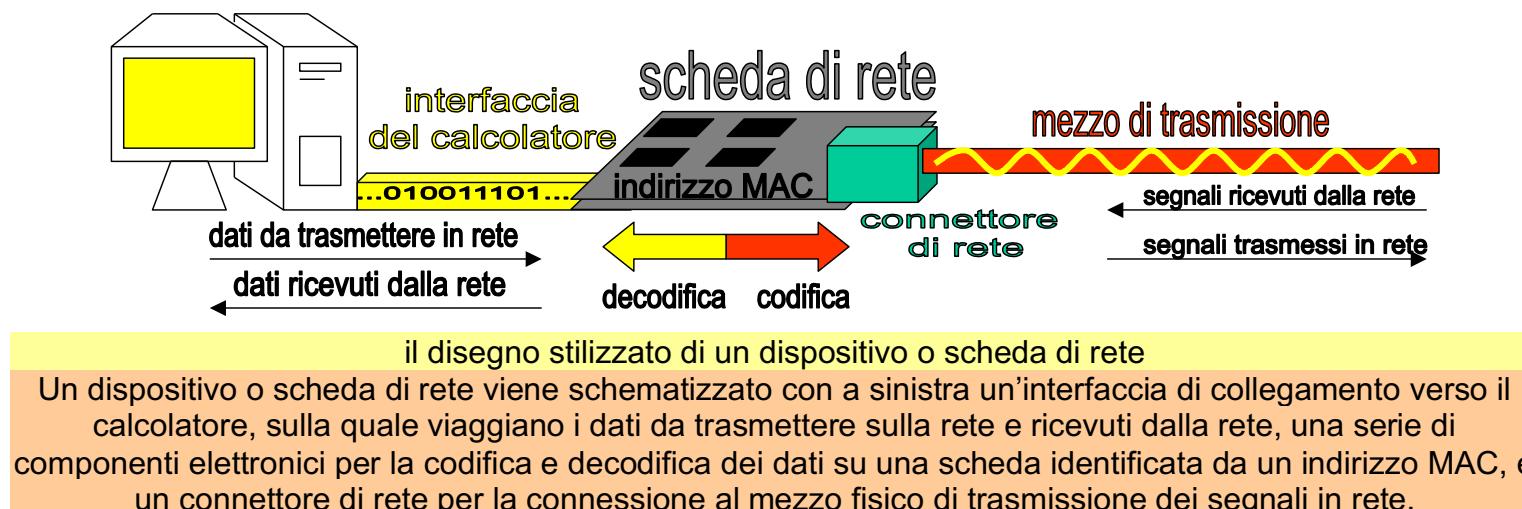
La tecnologia a fibre ottiche è tecnologicamente avanzata, e si basa sulla trasmissione di segnali ottici, cioè di luce, vincolata all'interno di una sottile fibra di vetro purissimo. La fibra è sottile come un capello, è elastica ed è protetta da una guaina esterna per facilitare il suo impiego. Il costo della fibra non è molto elevato, tuttavia la fibra è molto delicata per quanto riguarda la connessione degli estremi (giunzione) e questo influisce molto sui costi di distribuzione e di realizzazione dell'infrastruttura di rete. La capacità di una fibra ottica può arrivare oggi a qualche decina di migliaia di miliardi di bit al secondo (oltre 10000 Gbit/sec).

La terza tecnologia disponibile per il mezzo trasmissivo è fornita dalle onde elettromagnetiche e dalla loro propagazione nello spazio. Esempi in tal senso sono forniti dalle onde radio e dalla luce infrarossa. Tali tecnologie vengono dette senza fili (wireless). Le reti senza fili sono molto interessanti, e la loro diffusione è oggi esplosiva, in quanto permettono la mobilità dei dispositivi e degli utenti. La capacità dei collegamenti senza fili può arrivare oggi facilmente a qualche decina di milioni di bit al secondo (1-54 Mbit/sec per 802.11g). Le tecnologie WiFi più recenti superano i 300 Mb/s, es. 802.11n e ci sono tecnologie pronte a superare il Gb/s, es. 802.11ac). Il limite della tecnologia senza fili è dato dalla vulnerabilità del segnale rispetto ad errori e interferenza dei segnali, e dai limiti fisici della propagazione dei segnali. Due dispositivi possono essere connessi senza fili solo se rimangono entro un limite di distanza d che dipende dalla potenza del segnale radio emesso dal trasmettitore, e da eventuali ostacoli intermedi per il segnale.

Ogni rete può essere realizzata attraverso un singolo mezzo fisico di trasmissione, oppure attraverso la composizione di mezzi fisici eterogenei. Vedremo in seguito come possa avvenire la traduzione dei dati da un mezzo trasmissivo ad altri mezzi trasmissivi.

Dispositivo o scheda di rete

- **Dispositivo o scheda di rete** (hardware)
 - È un componente dell'architettura del calcolatore dotato di
 - un interfaccia di collegamento al calcolatore
 - un connettore di rete
 - trasmette e riceve i dati, opportunamente codificati e decodificati
 - dal calcolatore al mezzo di trasmissione: trasmissione
 - dal mezzo di trasmissione al calcolatore: ricezione
 - Dipende dal mezzo di trasmissione e dalla tecnologia di trasmissione usata
 - Scheda di rete per mezzi fisici cablati, per mezzi ottici, senza fili (wireless)
 - Prende il nome dai protocolli o dallo standard utilizzato per la trasmissione
 - Ha un codice identificativo unico: **indirizzo di livello MAC** (medium access control)



Reti di calcolatori - Intro

La comunicazione in rete tra calcolatori diversi, attraverso i vari mezzi di trasmissione illustrati, avviene mediante dispositivi interni o periferiche esterne del calcolatore, detti dispositivi o schede di rete. Le schede di rete sono collegate al calcolatore attraverso un'interfaccia di collegamento del calcolatore: su tale interfaccia transitano i dati (bit di informazione) da trasmettere in rete, oppure ricevuti dalla rete. La scheda di rete si occupa inoltre di trasformare i bit di informazione in segnali trasmissibili sul mezzo di trasmissione della rete e viceversa: tali trasformazioni si chiamano codifica e decodifica dei dati. Un connettore di rete pone direttamente in contatto la scheda di rete con il mezzo di trasmissione per l'invio e ricezione dei segnali in rete.

In sintesi, la funzione della scheda di rete è quella di memorizzare temporaneamente, codificare, decodificare, trasmettere e ricevere i dati da e verso il mezzo di trasmissione (cioè la rete) o il calcolatore.

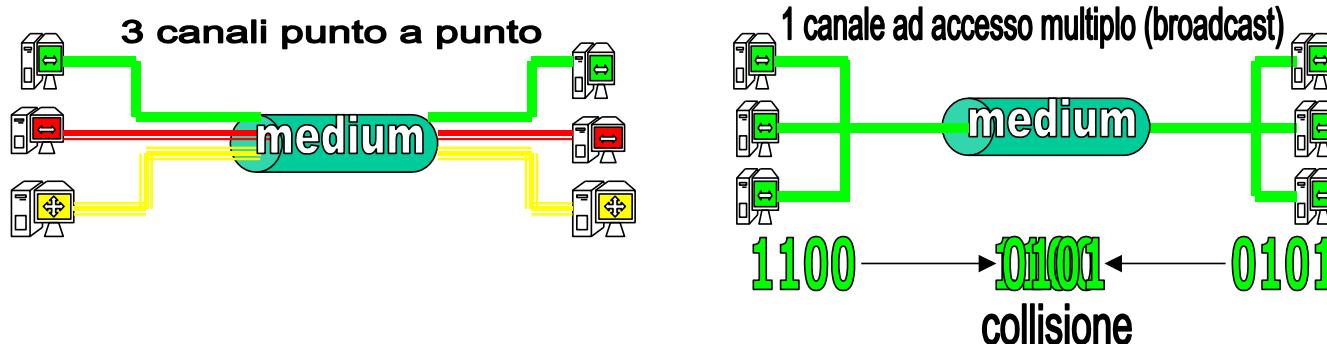
Il tipo della scheda di rete viene identificato a seconda del mezzo trasmisivo, e soprattutto a seconda dei protocolli di comunicazione utilizzati per la codifica, e per la trasmissione dei dati in rete.

Per le reti locali (LAN) basate su mezzo di trasmissione cablato, le tecnologie più diffuse sono chiamate con il nome del protocollo di comunicazione primario: ad esempio Ethernet, nelle varianti a 10, 100 Mbit/sec (Fast Ethernet) e 1000 Mbit/sec (Gigabit Ethernet). Per le reti LAN senza fili (WLAN), le schede di rete più diffuse sono denominate Wi-Fi (da 11 a 54 Mbit/sec), e Bluetooth (da 1 a 2 Mbit/sec).

Ogni scheda di rete, per permettere di essere identificata univocamente nel contesto di una rete locale, dispone dalla sua costruzione di un indirizzo univoco (unico) a livello mondiale, non modificabile, detto indirizzo MAC (Medium Access Control). Tali indirizzi vengono assegnati dai costruttori delle schede, per evitare che si possano originare indirizzi MAC duplicati.

canali di comunicazione della rete

- canale di comunicazione: una visione astratta del mezzo di trasmissione
- canale punto a punto: canale riservato sul mezzo trasmissivo tra due soli dispositivi
- canale ad accesso multiplo (canale broadcast)
 - tutti i dispositivi trasmettono e ricevono sullo stesso canale
 - tali canali richiedono arbitraggio (chi trasmette? quando?)
 - rischio collisione: sovrapposizione di trasmissioni sul canale (distrugge i dati)
 - Richiedono indirizzamento (quale dispositivo è destinatario dei dati trasmessi?)
 - Indirizzi univoci per mittente e destinatario (o destinatari) del dato trasmesso
 - indirizzo MAC della scheda di rete



Due esempi di unico mezzo trasmissivo contenente tre canali punto a punto, oppure un canale broadcast.

Sono mostrate due figure: la prima mostra un unico mezzo trasmissivo entro il quale sono realizzati tre diversi canali che connettono 3 coppie di stazioni, punto a punto. Un canale verde connette le due stazioni verdi, un canale rosso le due stazioni rosse e un canale giallo le due stazioni gialle. Tutti i canali suddetti passano per il mezzo unico di trasmissione condiviso. Nella seconda figura, lo stesso mezzo trasmissivo condiviso consente la creazione di un unico canale ad accesso multiplo (broadcast), di colore verde, che collega tre stazioni verdi sulla sinistra a tre stazioni verdi sulla destra. Due sequenze di bit trasmesse contemporaneamente sullo stesso canale (verde), danno luogo a una collisione che distrugge l'informazione trasmessa.

Introduciamo ora il concetto di canale di comunicazione.

Tralasciando la descrizione dei dettagli tecnici, a volte è possibile considerare il supporto fisico di un mezzo di trasmissione (medium) suddiviso in diversi canali di comunicazione separati. Ogni canale diverso può essere considerato un tubo virtuale sul quale possono essere trasmessi i bit di informazione. Il mezzo di trasmissione può quindi essere considerato come un fascio di canali di comunicazione.

Due definizioni dei canali possono essere realizzate: canali punto a punto e canali ad accesso multiplo.

I canali punto a punto si basano sull'accordo tra un mittente e un destinatario riguardante la definizione del canale da usare (in figura equivale al colore). Solo due dispositivi possono usare il canale di tipo punto a punto a loro riservato.

I canali ad accesso multiplo (broadcast) sono canali sui quali tutti possono trasmettere e dove tutti ricevono le trasmissioni di altri. Un problema per i canali ad accesso multiplo è legato alla possibile collisione di segnali appartenenti allo stesso canale di comunicazione. Se due trasmissioni di segnali si sovrappongono nel tempo sullo stesso canale di comunicazione, l'effetto sui segnali può essere distruttivo e l'esito della comunicazione può essere nullo. Intuitivamente, se due dispositivi trasmettono i loro segnali contemporaneamente, nessuno ricevitore sarà in grado di capire quali bit di informazione siano stati trasmessi.

Il problema delle collisioni è molto critico, e determina l'esigenza di arbitraggio nell'accesso al canale: chi trasmette e quando?

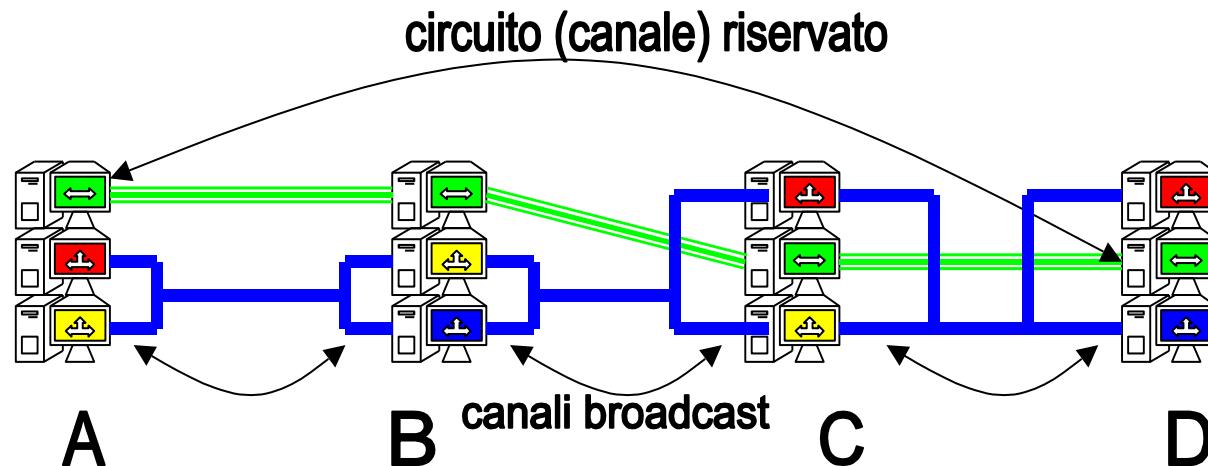
Il problema dell'arbitraggio può essere banale su canali punto a punto, dove mittente e destinatario possono definire semplici leggi (cioè protocolli di gestione della comunicazione) per evitare le collisioni: ad esempio, trasmetto io, poi trasmetti tu.

In canali condivisi ad accesso multiplo (broadcast) il problema risulta invece molto complesso, in quanto occorre definire leggi non ambigue, in grado di regolare gli accessi da parte di molti utenti, evitando le collisioni.

Reti a commutazione di circuito

Una serie di canali in cascata può realizzare un servizio di trasferimento dell'informazione tra due dispositivi anche molto distanti, secondo due modalità principali: commutazione di circuito (o circuito riservato) e commutazione di pacchetto.

- **Commutazione di circuito:** ad esempio, linea telefonica
 - Viene riservato un circuito di canali di comunicazione **punto a punto** su ogni connessione lungo tutto il cammino dal mittente al destinatario
 - Il circuito riservato ha un ritardo di comunicazione basso
 - Si paga il **tempo di connessione**, anche se non si scambiano dati sul canale
 - **Basso utilizzo** delle risorse di rete (canale sprecato quando non si trasmette)



Una sequenza di canali punto a punto riservati danno luogo a una rete a circuito riservato.

La figura mostra una sequenza di quattro reti: A, B, C e D. Ogni rete è connessa alle reti adiacenti attraverso canali punto a punto e canali ad accesso multiplo (broadcast). Una sequenza di dispositivi appartenenti rispettivamente alle reti A, B, C e D è connessa attraverso una sequenza di canali (verde o linea doppia) riservati. Tale sequenza rappresenta una rete a commutazione di circuito, ovvero a canale riservato al traffico tra mittente e destinatario appartenenti alle reti A e D.

Abbiamo visto come una rete sia un insieme di nodi connessi attraverso mezzi fisici di trasmissione sui quali possono esistere diversi canali di comunicazione. Componendo una serie di canali in cascata si può realizzare un servizio di trasferimento dell'informazione tra due utenti appartenenti a una rete più ampia, secondo due modalità principali: commutazione di circuito (o circuito riservato, circuit-switched) e commutazione di pacchetto (packet-switched).

Vediamo ora la definizione di rete a commutazione di circuito. Un esempio può essere fornito dalla rete telefonica. Nelle reti a commutazione di circuito, i dati vengono trasmessi tra un mittente e un destinatario finale agli estremi di un cammino (cicuito) di canali di comunicazione punto a punto. Il circuito viene negoziato e riservato a priori, attraverso opportune procedure (come avviene quando si digita un numero per una chiamata telefonica). Una volta identificati e ottenuti i canali che collegano mittente e destinatario, la comunicazione dati può avvenire anche come un'unica sequenza di bit, senza interruzioni. Non c'è nemmeno bisogno di dichiarare periodicamente chi sia il mittente e il destinatario dei dati, in quanto entrambi sono fissati al momento della creazione del circuito riservato.

Un ulteriore vantaggio è rappresentato dal ridotto ritardo di trasmissione per i dati, in quanto ogni nodo intermedio ha già disponibile il canale libero uscente sul quale inviare immediatamente i dati ricevuti sul canale entrante, dal mittente fino al destinatario finale. Tutto ciò si traduce in una riduzione del ritardo di rete (o latenza di rete) nella comunicazione.

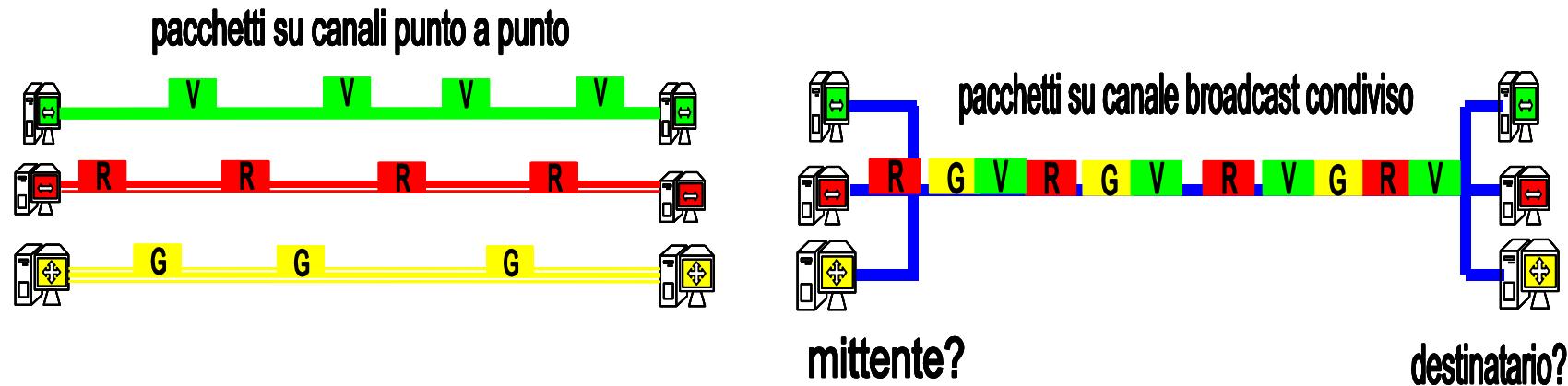
Se la quantità di dati da trasmettere non è molto grande, oppure se i dati da trasmettere arrivano a gruppi, intervallati da tempi di vuoto, allora l'utilizzo dei canali del circuito virtuale potrebbe diventare molto basso. Le risorse del canale verrebbero quindi addebitate per tutto il tempo di permanenza del circuito virtuale, anche se i dati fossero trasmessi solo all'inizio e alla fine del tempo di collegamento (proprio come accade per le telefonate).

Reti a commutazione di pacchetto

Reti a commutazione di pacchetto: alternativa alla commutazione di circuito.

Molto usata in reti basate su canali ad accesso multiplo (broadcast) e su Internet.

- I dati vengono suddivisi in **pacchetti indipendenti**
- ogni pacchetto viene spedito sul **canale**, separatamente
- in particolare, su canali ad accesso multiplo (broadcast):
 - ogni pacchetto deve contenere **indirizzo del destinatario** (e mittente)
 - Si ha condivisione del canale tra **diversi flussi** di pacchetti
 - Richiede minore numero di risorse di rete (canali e connessioni), ma **meglio utilizzate**
 - Maggiore **ritardo della rete** di comunicazione
 - Si paga per **quantità di dati trasmessi** (e non per il tempo necessario)



Due esempi: trasmissioni a commutazione di pacchetto su canali punto a punto e su canale broadcast

La figura mostra tre canali punto a punto sui quali circolano dal mittente al destinatario un certo numero di pacchetti di dati, lasciando molto tempo vuoto tra una trasmissione e la successiva. Nella seconda figura, la stessa informazione viene spedita usando un unico canale ad accesso multiplo (broadcast) e in questo modo i pacchetti occupano tutto il tempo utile sul canale, riducendo lo spreco.

In alternativa alla commutazione di circuito, una rete può trasferire l'informazione su un canale o su una sequenza di canali tra due utenti, secondo la modalità a commutazione di pacchetto. La maggior parte delle reti per trasmissione dati digitali, inclusa la rete di rete globale [Internet](#), sono di questo tipo.

I dati digitali vengono suddivisi in pacchetti separati, e vengono trasmessi su canali ad accesso multiplo ([broadcast](#)).

La trasmissione a pacchetto su canali punto a punto lascerebbe molto tempo vuoto (inutilizzo) sui canali. Un unico canale condiviso ad accesso multiplo potrebbe invece essere molto bene utilizzato per scambiare lo stesso quantitativo di dati, suddivisi a pacchetti, tra i rispettivi mittenti e destinatari (come mostrato in figura). Per consentire la corretta ricezione dei dati, è però necessario includere in ogni pacchetto l'informazione sull'identità del rispettivo mittente e soprattutto del destinatario.

Si attua in questo modo la condivisione di un canale unico per diversi flussi di pacchetti appartenenti a diversi mittenti e destinatari.

Componendo in serie una sequenza di canali [broadcast](#) a commutazione di pacchetto, i nodi ricevitori devono di volta in volta farsi carico di verificare se il pacchetto sia giunto a destinazione o, in caso contrario, possono provvedere all'inoltro del pacchetto ricevuto sul successivo canale [broadcast](#).

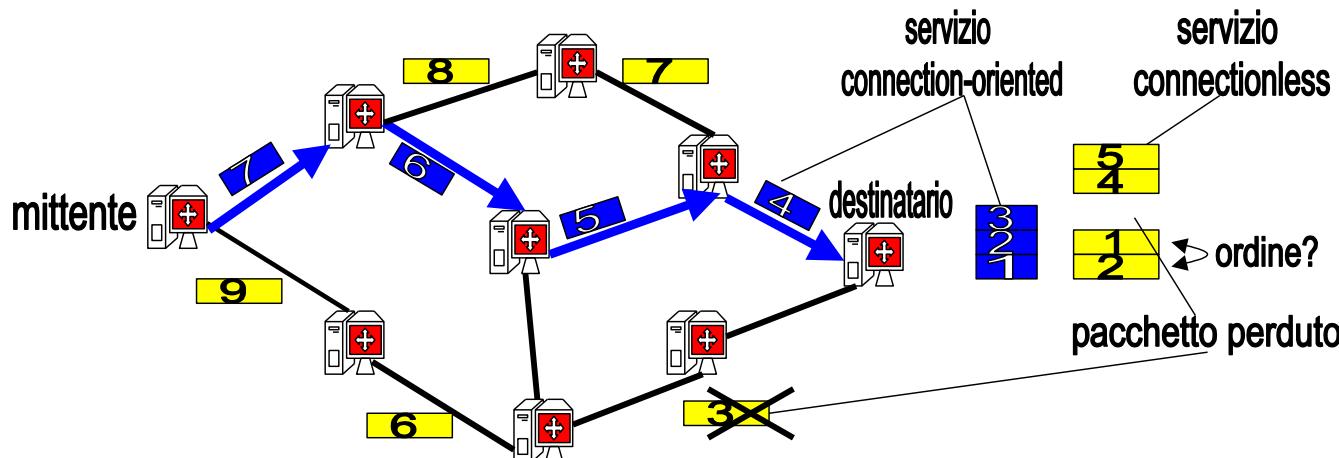
Il prezzo di questa tecnica è legato al maggiore ritardo per la comunicazione dei dati tra mittente e destinatario, dovuto all'esigenza di iterare più volte la ricezione e l'inoltro di pacchetti su canali [broadcast](#) in sequenza.

Il vantaggio è legato al maggiore utilizzo dei canali ad accesso multiplo, e quindi alla possibilità di tariffare la comunicazione in base ai dati trasmessi, e non in base al tempo necessario.

Servizi orientati alla connessione e non

Nelle **reti a commutazione di pacchetto** il servizio di trasmissione dei dati, tra mittente e destinatario ha proprietà determinate dal ruolo dei protocolli di rete utilizzati.

- I dati sono spediti in pacchetti con indicato mittente e destinatario
- ad ogni nodo intermedio i pacchetti vengono immagazzinati e inoltrati verso il destinatario finale
- è possibile la perdita o l'arrivo disordinato di pacchetti della sequenza inviata
- Servizi **orientati alla connessione** (connection-oriented), esempio: telefono, circuito virtuale
 - Garantiscono la consegna ordinata dei pacchetti ricevuti, secondo l'ordine di invio
 - Garantiscono la ri-trasmissione di eventuali pacchetti perduti
- Servizi **non orientati alla connessione** (connectionless), esempio: lettere di posta ordinaria
 - I pacchetti possono seguire strade diverse, e arrivare in ordine diverso o non arrivare mai



Una rete con un servizio di trasmissione orientato alla connessione, e un servizio di trasmissione non orientato alla connessione.

La figura mostra l'astrazione di una rete con un mittente e un destinatario agli estremi. Esistono molti cammini alternativi per inviare informazione suddivisa a pacchetti dal nodo mittente al nodo destinatario. Uno di questi cammini è stato identificato e riservato per l'invio della sequenza di pacchetti numerati da uno a sette. I pacchetti su tale circuito sono inviati in serie, e non possono andare perduti, né subire sorpassi. Essi giungono quindi al nodo destinatario in ordine di invio e senza perdite. Tale servizio è un esempio di

Reti di calcolatori - Intro

servizio di trasmissione orientato alla connessione. I pacchetti di colore giallo, numerati da uno a nove, sono invece inviati tra i nodi della rete scegliendo di volta in volta un cammino qualsiasi tra quelli disponibili. Ogni pacchetto può andare perduto ai nodi intermedi (ad esempio come il pacchetto numero tre), e se giunge a destinazione può essere fuori ordine (ad esempio come i pacchetti uno e due).

Nelle reti a commutazione di pacchetto il servizio di trasmissione dei dati, tra mittente e destinatario ha proprietà determinate dal ruolo dei protocolli utilizzati. I pacchetti di dati sono immagazzinati temporaneamente dai dispositivi intermedi del cammino e quindi inoltrati verso il destinatario finale. In questo modo i pacchetti possono andare perduti, per varie cause, e possono arrivare in ordine diverso rispetto all'ordine di invio.

I servizi di trasmissione dei dati, tra mittente e destinatario remoti, possono essere servizi orientati alla connessione (connection-oriented) e servizi non orientati alla connessione (connectionless).

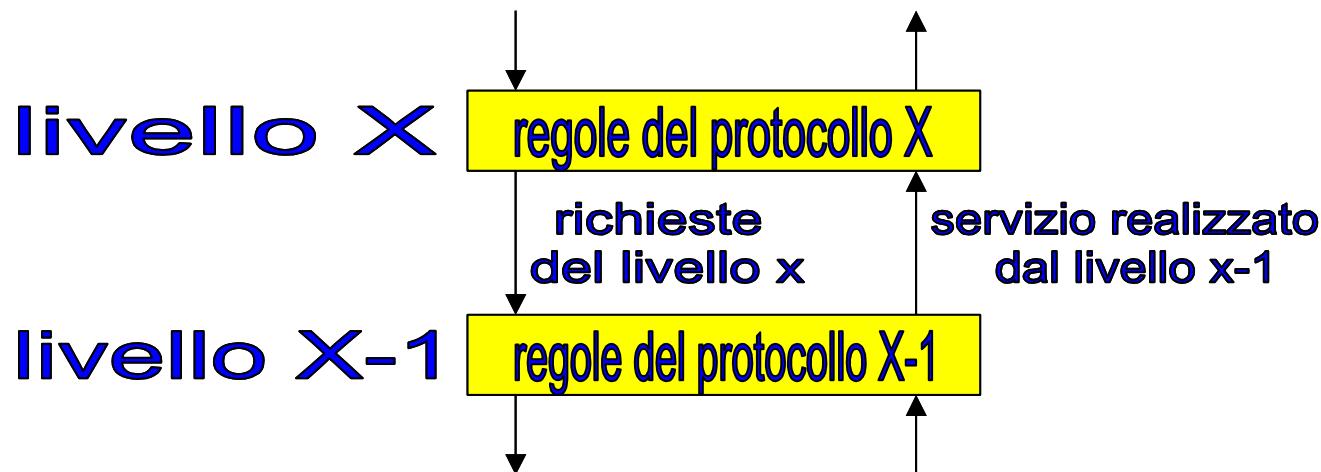
I servizi orientati alla connessione garantiscono che la spedizione di pacchetti di dati tra mittente e destinatario sia equivalente a una trasmissione affidabile e corretta. In altre parole, essi implementano una serie di operazioni attraverso le quali tutti i pacchetti perduti saranno ritrasmessi, e correttamente ordinati, fino a ricostruire esattamente tutta l'informazione trasmessa. L'implementazione di tale tipo di servizi potrebbe essere basata sulla definizione di vari protocolli alternativi. Ad esempio, potrebbe essere definito un cammino riservato unico per i pacchetti. Intuitivamente, ciò equivale a un circuito virtuale per l'invio dei pacchetti. Un altro modo per ottenere tale servizio potrebbe essere basato sulla numerazione dei pacchetti inviati, sul riordino dei pacchetti ricevuti e sulla richiesta di ri-trasmissione dei pacchetti perduti. Questo protocollo, descritto sommariamente, sarà ripreso in seguito, in quanto una definizione simile è alla base della comunicazione a pacchetto di tipo connection-oriented su Internet.

I servizi non orientati alla connessione (connectionless) non si preoccupano di garantire l'ordine corretto dei pacchetti inviati e nemmeno la ricezione di tutti i pacchetti. Tale servizio è simile all'invio dei pacchetti in modo analogo a una sequenza di lettere attraverso la posta ordinaria.

Protocolli di rete organizzati a livelli

Iniziamo a considerare ora gli aspetti di gestione della comunicazione nelle reti di calcolatori

- Definizione di **protocollo**
 - Regole e procedure (semantiche) di gestione dei processi di comunicazione
 - Regole e formati (sintattici) per definire scambio di messaggi non ambigui
 - permettono compatibilità dei dispositivi e dei sistemi conformi allo stesso standard
- **Architettura dei protocolli di rete:** separazione in livelli dei protocolli di rete
 - ogni livello affronta e risolve un problema della comunicazione
 - I livelli superiori effettuano richieste di servizio al loro livello inferiore
 - I livelli inferiori forniscono servizi al loro livello superiore
 - Le richieste e i servizi realizzano l'interfaccia del protocollo verso altri livelli



La necessità di accordarsi su regole e servizi comuni per la comunicazione di rete ha lo scopo di permettere una completa compatibilità e supporto alla comunicazione su sistemi, tecnologie e dispositivi eterogenei.

Le regole che governano i processi di comunicazione in rete, tra dispositivi e sistemi eterogenei, prendono il nome di protocolli di rete. I protocolli definiscono aspetti e regole semantiche sulla sequenza dei messaggi, e regole sintattiche sul formato dei messaggi scambiati durante la comunicazione. La definizione dei protocolli di rete deve prevedere e supportare diverse finalità di comunicazione. Non ha quindi senso definire un protocollo rigido, ma ha senso definire classi di protocolli, deputate a svolgere e gestire determinate funzioni della comunicazione. Tali classi di protocolli, opportunamente organizzate, permettono di semplificare la gestione della rete, ma è necessario definire in modo non ambiguo le relazioni tra le classi di protocolli (ovvero quale protocollo si occupa di gestire un certo problema? Come avviene il dialogo tra protocolli?)

La soluzione che è stata individuata, e ha rappresentato uno dei principali cardini del successo delle reti e della nascita di Internet, è data dalla separazione delle classi di protocolli in livelli. La struttura dei livelli dei protocolli di rete prende il nome di architettura dei protocolli di rete.

Il concetto di architettura dei protocolli, suddivisa in livelli, è semplice ed è basato su alcune condizioni.

Ogni livello svolge determinate funzioni di gestione dei processi di comunicazione, attraverso uno o più protocolli alternativi.

Ogni livello fornisce un livello di astrazione più elevato della rete di comunicazione sottostante, sfruttando i servizi implementati dai livelli sottostanti. In altre parole, i livelli superiori non devono preoccuparsi di risolvere problemi che saranno gestiti e risolti dai livelli inferiori.

Ogni livello ha relazioni dirette solo con i livelli immediatamente superiore e inferiore, attraverso richieste e servizi concordati, detti interfaccia del livello.

Un esempio di architettura dei protocolli

Due innamorati, un italiano e una giapponese, desiderano comunicare per scambiarsi una dichiarazione d'amore, ma devono superare l'eterogeneità e i vincoli del mondo che li separa. Essi possono leggere e scrivere solo nella rispettiva lingua madre. L'unico mezzo fisico di comunicazione è il FAX, l'unico modo per scrivere è una macchina per scrivere testo con alfabeto cirillico e l'unica lingua efficace da esprimere in alfabeto cirillico è la lingua russa.



Un esempio di architettura di protocolli per la comunicazione divisi in livelli, con evidenziati i servizi forniti

La figura mostra due pile, ognuna composta da quattro livelli di protocolli per la comunicazione tra un innamorato italiano e una innamorata giapponese. La dichiarazione d'amore dell'innamorato italiano viene scritta usando il servizio del livello dialogo, poi viene tradotta in russo usando il servizio del livello traduzione, poi viene scritta in caratteri cirillici dal livello dattilografia e infine spedita dal livello FAX. Sull'altro lato, il fax viene ricevuto dal livello FAX, viene convertito in caratteri giapponesi dal livello dattilografia, viene tradotto dal russo al giapponese dal livello traduzione e viene fornito il testo tradotto in giapponese per la lettura dal livello dialogo all'innamorata giapponese.

La figura mostra un esempio intuitivo per chiarire il concetto di architettura a livelli dei protocolli di comunicazione. Sono rappresentati quattro possibili livelli e i relativi servizi implementati da protocolli di comunicazione, definiti per vincere l'eterogeneità e supportare la comunicazione tra due innamorati separati da un mondo eterogeneo e pieno di vincoli. Gli innamorati desiderano scambiarsi una dichiarazione d'amore, ma possono leggere e scrivere solo nella rispettiva lingua madre: italiano e giapponese. A complicare le cose, essi hanno a disposizione solo un FAX per la trasmissione, solo una macchina per scrivere in alfabeto cirillico, attraverso il quale risulta pratico ed efficace esprimersi solo in lingua russa. L'architettura dei livelli e i protocolli di comunicazione definiti per superare i problemi di comunicazione è mostrata in figura. Al livello più alto (livello dialogo) avviene lo scambio della dichiarazione nella rispettiva lingua madre. Al livello dialogo, l'innamorato italiano detta quindi la sua dichiarazione in italiano. Dal livello dialogo la dichiarazione viene passata al livello traduzione, chiedendo il servizio di traduzione in lingua russa. A questo punto il livello traduzione passa la traduzione russa al livello dattilografia, chiedendo il servizio di battitura del testo in alfabeto cirillico. A questo punto il livello dattilografia richiede al livello FAX sottostante il servizio di spedizione del FAX al numero di FAX dell'innamorata giapponese. Avvenuta la trasmissione del FAX, il livello FAX dell'innamorata giapponese riceve il FAX, e quindi lo passa al livello dattilografia dove il testo cirillico viene riletto in lingua russa. Il testo in lingua russa viene passato al livello traduzione, dove avviene la traduzione dalla lingua russa alla lingua giapponese, e finalmente la dichiarazione può essere ricevuta ed ascoltata dall'innamorata giapponese, al livello dialogo, come se l'innamorato italiano le stesse parlando all'orecchio in lingua giapponese. Ovviamente la risposta alla dichiarazione dovrà passare per le stesse fasi e gli stessi protocolli, ma in direzione opposta. Si noti come ad ogni livello, non ci si debba curare dei dettagli e dei problemi risolti ai livelli inferiori. Ogni livello deve saper risolvere un problema e solo quello. Al livello più alto, si realizza l'equivalente del dialogo diretto tra i due innamorati.

Architettura Standard di protocolli di rete

Esiste un riferimento standard per definire l'architettura dei protocolli delle reti di calcolatori

- **Standard ISO/OSI RM** (Open System Interconnection Reference Model)
 - insieme di livelli completo e rigoroso, per supportare la comunicazione in rete
- Definisce un'architettura dei protocolli di rete organizzata in **sette livelli**
 - Ogni livello gestisce una classe di problematiche di rete
 - Ogni livello fornisce ai livelli superiori una visione della rete semplificata
 - Dialogo tra livelli paritari avviene astraendo l'architettura sottostante
 - Dialogo tra livelli sovrapposti attraverso interfaccia comune per tutti i protocolli



Reti di calcolatori - Intro

La pila di livelli dei protocolli di rete dello standard ISO/OSI.

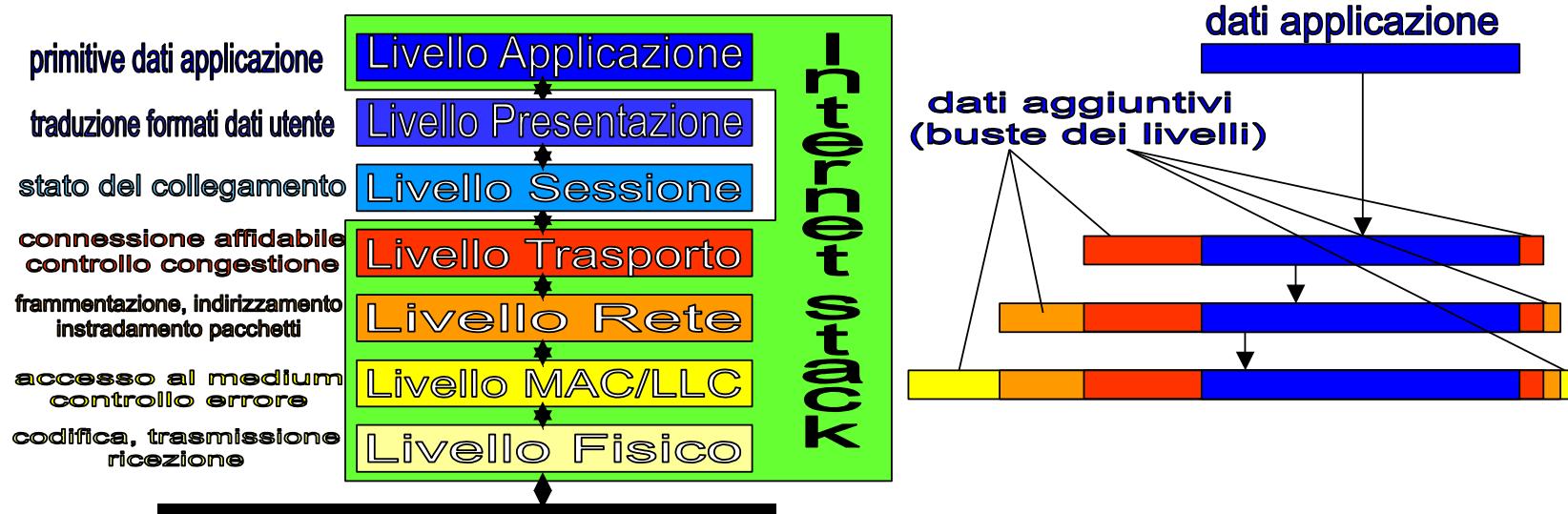
Sono mostrate due istanze, una per il nodo mittente e una per il nodo destinatario, dei sette livelli di protocolli di rete dello Standard ISO/OSI RM. Partendo dall'alto, a scalare dal livello 7 al livello 1, abbiamo il livello 7 Applicazione, 6 Presentazione, 5 Sessione, 4 Trasporto, 3 Rete, 2 Logical Link Control / Medium Access Control (MAC/LLC) e infine il livello 1 Fisico, che si appoggia direttamente sul medium fisico di trasmissione. Il livello applicazione fornisce i servizi di trasmissione dati alle applicazioni in esecuzione, il livello presentazione risolve eventuali eterogeneità dei dati tra i nodi della rete, il livello sessione mantiene lo stato del collegamento, il livello trasporto fornisce un servizio di connessione affidabile e controllo della congestione, il livello rete si occupa di frammentazione dei dati, indirizzamento e instradamento dei pacchetti, il livello LLC/MAC si occupa di garantire una comunicazione affidabile e l'arbitraggio di accesso al mezzo trasmissivo, e il livello fisico si occupa di codificare i dati e trasmetterli sul medium di trasmissione.

Lo Standard ISO/OSI RM (Open System Interconnection Reference Model) definisce un insieme di livelli completo e rigoroso per l'architettura dei protocolli di rete, prevedendo un livello per la gestione di ogni problema di comunicazione in rete. L'architettura dei protocolli di rete definita da ISO/OSI RM prevede sette livelli dei protocolli, numerati da 7 a 1 dall'alto al basso. Vediamo una breve panoramica dei livelli e dei servizi principali implementati. Il livello applicazione (7) fornisce alle applicazioni in esecuzione sul calcolatore i servizi e le primitive di trasmissione e ricezione dei dati. Il livello presentazione (6) risolve eventuali eterogeneità del formato dei dati tra i nodi della rete. Il livello sessione (5) mantiene e gestisce lo stato attuale del collegamento tra due applicazioni remote. Il livello trasporto (4) si occupa di garantire i servizi di trasmissione dei pacchetti (orientati alla connessione e non) e del controllo della congestione della rete. Il livello rete (3) si occupa di frammentare i dati in pacchetti, scrivere gli indirizzi dei destinatari finali e instradare i pacchetti verso i destinatari intermedi del cammino. Il livello LLC/MAC (2) si occupa di garantire l'affidabilità del mezzo di trasmissione e la gestione dell'accesso al mezzo trasmissivo ad accesso multiplo (evitando le collisioni). Infine, il livello fisico (1) si occupa di definire le tecniche di codifica dei dati, la trasmissione e la ricezione dei dati sul mezzo fisico di trasmissione.

Vedremo ora quali livelli dello Standard ISO/OSI, e quali protocolli in ogni livello siano di fatto utilizzati nell'architettura dei protocolli delle reti di calcolatori locali fino alla rete Internet.

Architettura dei protocolli di Internet

- L'architettura dei livelli di protocolli di Internet utilizza solo **5 dei 7 livelli ISO/OSI RM**
- In trasmissione, ogni livello riceve dati dai livelli superiori e li inserisce (incapsula) in “buste” con dati aggiuntivi, utili ad istruire il corrispondente livello del dispositivo ricevente
- In ricezione, ogni livello verifica i dati della busta, agisce di conseguenza, e passa il contenuto della busta ai livelli superiori
 - Il livello trasporto imbusta i dati aggiungendo informazioni utili all'ordinamento e controllo della velocità di invio delle buste
 - Il livello rete frammenta i dati in pacchetti, decide il cammino sul quale inviare il pacchetto a seconda dell'indirizzo del destinatario
 - Il livello MAC/LLC esegue la consegna finale dei dati a dispositivi di una rete locale.



L'architettura dei protocolli di Internet, rispetto allo Standard ISO/OSI, e un esempio di incapsulamento dei dati tra i vari livelli.

La figura mostra i sette livelli dello Standard ISO/OSI, evidenziando come solo i livelli 1 (fisico), 2 (MAC/LLC), 3 (rete), 4 (trasporto) e 7 (applicazione) siano di fatto i soli livelli implementati nell'architettura dei protocolli di rete di Internet. A livello applicazione, i dati che l'applicazione richiede di spedire vengono passati al livello trasporto. Il livello trasporto frammenta e incapsula i dati ricevuti entro una “busta virtuale” realizzata dai dati di intestazione (header) e di coda (trailer) del livello stesso, e quindi richiede l'invio dei dati ottenuti al livello sottostante (rete). Il livello rete incapsula nuovamente i dati ricevuti in una busta

Reti di calcolatori - Intro

contenente le informazioni relative alla gestione del protocollo adottato, e passa il tutto al livello MAC/LLC. Il livello MAC/LLC effettua un nuovo incapsulamento con i dati del livello stesso e finalmente passa tutto al livello fisico per la trasmissione sul mezzo trasmissivo.

L'architettura dei protocolli di Internet, nel senso più comunemente adottato, prevede di fatto l'implementazione di solo cinque livelli dei sette livelli dello Standard ISO/OSI RM. Rimangono spesso esclusi i livelli 5 (sessione) e 6 (presentazione).

Per i livelli rimasti: fisico, MAC/LLC, rete, trasporto e applicazione valgono le considerazioni fatte nella slide relativa allo Standard ISO/OSI RM.

Un aspetto che si pone in evidenza, è il concetto di incapsulamento dei dati tra i livelli implementati. In fase di trasmissione, ogni livello riceve dati dall'alto (dati spediti dall'applicazione) e li inserisce in "buste virtuali" (incapsulamento) ponendo in testa e in coda alcuni dati aggiuntivi, necessari per fornire al livello della controparte ricevente le informazioni utili all'implementazione del protocollo dello stesso livello. In fase di ricezione, ogni livello X riceve dal livello più basso i dati imbustati dallo stesso livello X sul trasmettitore, quindi verifica i dati della busta, agisce in conseguenza alle specifiche fornite nei dati della busta, e passa solo il contenuto della busta ai livelli superiori (decapsulamento).

Il livello trasporto spezza i dati dell'applicazione in frammenti e li imbusta, aggiungendo informazioni utili all'ordinamento e al ri-assemblaggio dei dati ricevuti, oltre che al controllo della congestione della rete. Il livello rete frammenta ulteriormente i dati in pacchetti (se sono troppo lunghi), scrive l'indirizzo del destinatario sulla busta, e decide il cammino sul quale inviare il pacchetto a seconda dell'indirizzo di rete del destinatario. Il livello MAC/LLC esegue la consegna finale dei dati a dispositivi di una rete locale.

Nel seguito procederemo all'analisi livello per livello dell'implementazione dei protocolli che regolano il funzionamento e i servizi delle reti locali, delle internetwork (reti locali connesse tra loro), e di Internet (ovvero la rete di tutte le reti connesse e conformi all'architettura di Internet).

livelli e integrazione delle reti

Vediamo in dettaglio quali siano le problematiche gestite dai protocolli di rete a partire dai livelli più bassi (livello fisico) fino ai livelli più alti (livello applicazione). Ad ogni livello i protocolli gestiscono problemi diversi e la rete assume caratteristiche di integrazione e proprietà diverse.

- livello fisico: la rete è solo un segmento = un mezzo di trasmissione condiviso tra dispositivi
 - regole per codificare e trasmettere dati, con una tecnologia in comune
- livello MAC/LLC: la rete è locale, può integrare mezzi trasmissivi e tecnologie diverse
 - regole per gli indirizzi dei dispositivi, i tempi di accesso al mezzo, e gestione errori
- livello rete: la rete è una collezione di reti, e assume struttura gerarchica (reti di reti, sottoreti)
 - regole per indirizzi di rete che nascondano i dettagli locali
 - si definiscono nuovi dispositivi (router) che smistano i pacchetti di dati tra rete e rete
- livello trasporto: la rete è una collezione di reti organizzate gerarchicamente
 - regole per la spedizione affidabile di pacchetti, e controllo della congestione della rete
- livello applicazione: la rete esiste, funziona, ed è utilizzabile dalle applicazioni dell'utente

Nel seguito della trattazione analizzeremo più in dettaglio quali siano le problematiche gestite dai protocolli di rete a partire dai livelli più bassi (livello fisico) fino ai livelli più alti (livello applicazione). Ad ogni livello affronteremo i protocolli che gestiscono problemi diversi, e analizzeremo il tipo, le caratteristiche di integrazione e le proprietà della rete ottenuta. Al termine saranno evidenziati l'architettura completa dei protocolli di rete e tutti i dispositivi che permettono il funzionamento e l'integrazione delle reti, dalle reti locali fino a Internet.

Il livello uno (fisico) si occupa delle regole per codificare e trasmettere i dati come segnali sul mezzo trasmittivo a disposizione. A questo livello la rete è solo un segmento con la stessa tecnologia in comune a tutti i dispositivi.

Il livello due (MAC/LLC) affronta la definizione delle regole per determinare gli indirizzi dei dispositivi ai quali i dati sono destinati, le regole per accedere al mezzo trasmittivo ad accesso multiplo (condiviso) evitando errori di trasmissione, e le regole per recuperare eventuali errori delle trasmissioni.

A questo livello la rete può integrare diversi mezzi trasmittivi e diverse tecnologie (segmenti diversi), e assume i connotati di una rete locale (LAN).

Il livello tre (rete) affronta il problema dell'integrazione generalizzata di reti, fino a formare reti di reti, organizzate secondo una visione gerarchica (reti di reti e sottoreti). Le regole riguardano la definizione di nuovi indirizzi di rete, che nascondono i dettagli di gestione delle reti locali e delle sottoreti all'esterno. Nuovi dispositivi, detti instradatori (router), sono creati per la gestione dell'instradamento dei pacchetti dati tra una rete e l'altra. I router agiscono da smistatori dei pacchetti tra le reti, riducendo la complessità dell'integrazione delle reti di reti.

A questo livello esiste una rete di reti, ma il servizio di consegna dei pacchetti è di tipo connectionless e i pacchetti di dati possono essere perduti o arrivare disordinati.

Il livello quattro (trasporto) prevede la gestione delle regole che permettano di ripristinare un servizio orientato alla connessione, per il quale i pacchetti spediti siano consegnati tutti e nell'ordine corretto. Si definiscono inoltre le regole per controllare il rischio di congestione della rete.

A questo livello esiste una rete di reti affidabile di tipo orientato alla connessione. Internet è una rete di questo tipo, basata su due protocolli in particolare, che vedremo in seguito (TCP e IP).

Il livello sette (applicazione) mette semplicemente a disposizione il servizio di comunicazione della rete di reti alle applicazioni dell'utente.

Il livello fisico: codifica dei dati digitali

La trasmissione dati sul canale di comunicazione richiede la **codifica e decodifica dei dati**, uno di seguito all'altro sul mezzo trasmisivo, da parte della scheda di rete.

- **dati digitali: bit** (minima unità di informazione dei dati del calcolatore)
 - ogni dato del calcolatore da trasmettere in rete ha solo due valori: 0 oppure 1
- codifica di dati **digitali** usando segnali **analogici**
- velocità di trasmissione dei segnali sul mezzo trasmisivo
 - Tutti i segnali elettromagnetici viaggiano alla velocità fisica della luce
- capacità del canale di trasmissione: il valore massimo di bit/secondo trasmessi
 - determinata dal tempo necessario alla codifica del valore di ogni bit



esempio di canali digitali: canale B giallo ha capacità di trasmissione doppia del canale A rosso
 La figura mostra l'astrazione di due canali digitali con evidenziata la caratteristica di una diversa capacità di trasmissione dei bit. La velocità di propagazione fisica dei bit è la stessa, ma il canale B giallo impiega metà tempo per codificare ogni bit, e quindi ha il doppio di capacità di trasmissione rispetto al canale A rosso.

I valori possibili per i dati digitali (bit) del calcolatore sono solo due: i valori 0 e 1. Tali valori devono essere trasmessi o ricevuti sui mezzi di trasmissione delle reti, sotto forma di variazioni di segnali analogici (elettrici, ottici o radio), uno di seguito all'altro. Per questo scopo, i dati digitali devono essere opportunamente codificati o decodificati, in sequenza, da parte della scheda di rete.

L'attività di codifica, effettuata in fase di trasmissione, equivale a tradurre i valori dei bit in segnali analogici.

L'attività di decodifica, effettuata in fase di ricezione, equivale a tradurre i segnali analogici ricevuti nei valori dei bit.

Le tecniche di codifica digitali permettono di ridurre, ma non di escludere completamente, la possibilità di errori di trasmissione sulla rete.

Una nota importante riguarda l'ambiguità di fondo sul concetto di velocità della trasmissione dei segnali in rete.

Dal punto di vista fisico, tutti i segnali analogici, elettrici, ottici o radio, si propagano praticamente alla stessa velocità, cioè alla velocità della luce, pari a circa 300.000 Km/sec. Non ha quindi senso parlare di bit, oppure di segnali, più veloci di altri.

Tuttavia, nell'esempio in figura, un canale A di comunicazione sul quale siano codificati dieci bit al secondo ha una densità di trasmissione dei bit (detta anche capacità del canale) pari alla metà della capacità ottenuta da un canale B, sul quale possano essere codificati venti bit al secondo. I canali a capacità più elevata, ovvero in grado di trasmettere più bit al secondo, devono tali prestazioni al fatto di usare meno tempo per codificare, ovvero rappresentare il valore del bit sul mezzo trasmittivo, rispetto a canali più "lenti".

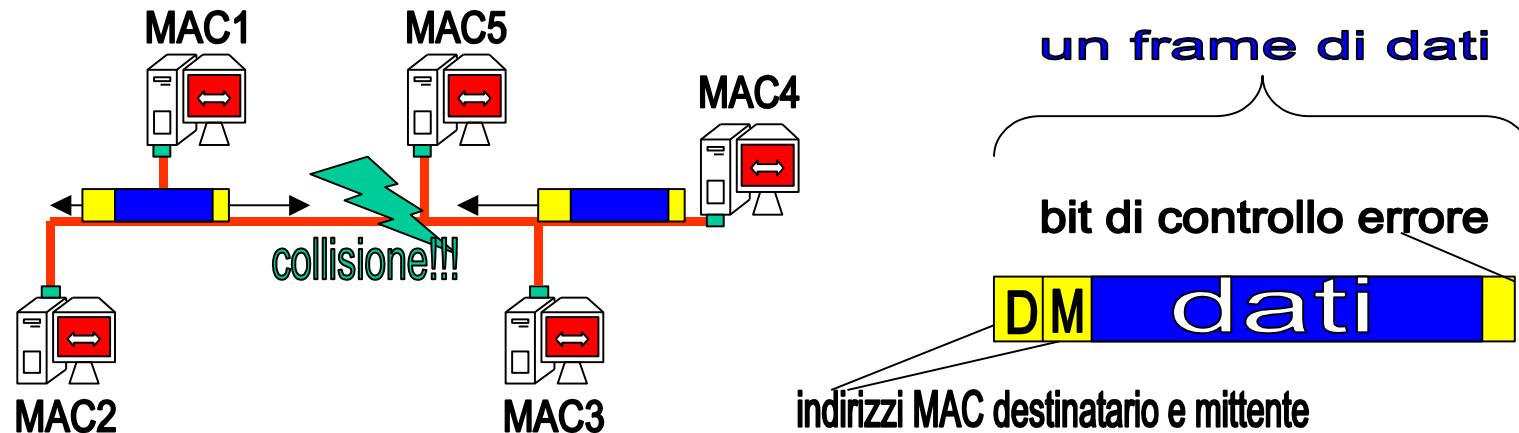
Le migliori tecnologie di rete sono quelle che permettono di codificare i bit nel minor tempo possibile, ottenendo quindi alte capacità dei canali (ad esempio, miliardi di bit trasmessi al secondo).

Un segmento di rete locale

Il livello più semplice di una rete a commutazione di pacchetto: un segmento di rete locale

- segmento di rete locale: un mezzo di trasmissione condiviso con canale ad accesso multiplo
 - tutte le schede di rete ricevono le trasmissioni effettuate sul mezzo trasmittivo
- ogni scheda di rete ha assegnato un indirizzo fisico diverso: **indirizzo MAC**
- compiti e servizi offerti dai protocolli di livello 2 (MAC/LLC): **accesso al mezzo trasmittivo**
 - trasmissione: arbitraggio degli accessi al canale: chi trasmette e quando?
 - indirizzamento di livello MAC: quale scheda di rete è destinataria del frame trasmesso?
 - Ricezione: il frame di dati è destinato al mio indirizzo MAC?
 - in caso affermativo, ricevo il frame e lo passo ai protocolli dei livelli superiori

segmento di rete LAN: canale condiviso



Un esempio di rete locale a canale condiviso e 5 dispositivi con indirizzo MAC diverso.

La figura mostra cinque calcolatori connessi a un canale unico condiviso (broadcast). Ogni dispositivo ha una scheda di rete connessa al mezzo di trasmissione, dotata del suo indirizzo MAC unico al mondo. La figura mostra che la trasmissione contemporanea di due pacchetti (frame) da parte di due stazioni diverse origina una collisione distruttiva dei segnali. Viene mostrato inoltre come la struttura del frame trasmesso

Reti di calcolatori - Intro

abbia in testa l'indirizzo MAC del destinatario e del mittente del frame, al centro i dati (incapsulati dai livelli superiori), e in coda un campo di bit utile a scoprire la presenza di eventuali bit errati.

Analizziamo ora l'esempio più semplice per la definizione di una rete di calcolatori a commutazione di pacchetto: un segmento di rete locale. Un segmento di rete locale è definito come un mezzo di trasmissione condiviso sul quale sia definito un canale ad accesso multiplo.

Ogni calcolatore si assume dotato della scheda di rete opportuna per il mezzo trasmittivo e i protocolli di codifica utilizzati al livello uno (fisico). Ogni scheda di rete è dotata di un identificativo (indirizzo) di livello MAC unico al mondo (assegnato dal costruttore). Ogni trasmissione di un pacchetto di dati (detto frame a questo livello) sul canale ad accesso multiplo è ricevuta da tutti i calcolatori la cui scheda di rete sia connessa al canale stesso.

Immaginiamo che il dispositivo con indirizzo MAC1 voglia spedire un frame di dati al dispositivo con indirizzo MAC5, e allo stesso tempo il dispositivo di indirizzo MAC4 voglia spedire un frame di dati al dispositivo di indirizzo MAC2.

Nel contesto del canale condiviso, la conoscenza degli indirizzi MAC dei dispositivi mittente e destinatario basta ad effettuare la trasmissione, sul canale comune. Semplificando molto il problema, per ragioni di presentazione, è sufficiente aggiungere le informazioni sull'indirizzo MAC del destinatario e del mittente sulla busta di ogni frame, prima di trasmetterlo.

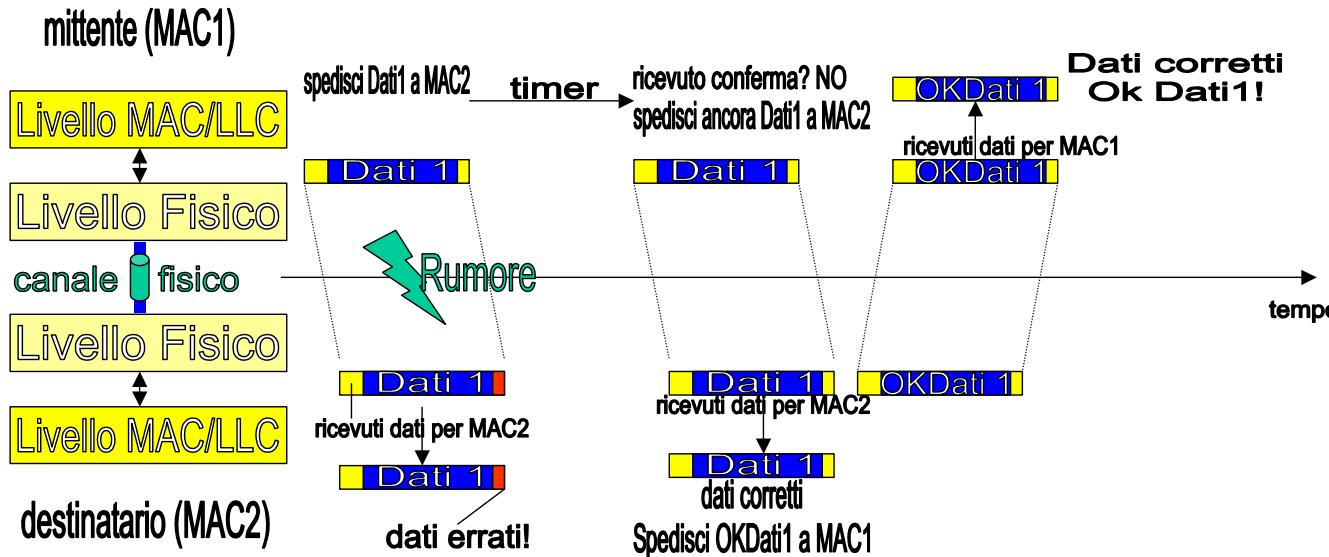
Ogni frame trasmesso sul canale da parte di ogni dispositivo risulta quindi rilevato da tutti gli altri dispositivi, ma viene ricevuto (cioè copiato e passato ai livelli superiori) solo se l'indirizzo MAC del destinatario specificato nel frame coincide con l'indirizzo MAC del dispositivo ricevente.

Il compito principale dei protocolli di livello 2 (MAC/LLC), oltre all'indirizzamento dei frame trasmessi sul canale condiviso del segmento di rete locale, è dato dall'arbitraggio degli accessi al canale. Ciò equivale a determinare quali nodi possano trasmettere, quando, e in che ordine, in modo da evitare le collisioni tra due o più trasmissioni contemporanee. Le collisioni equivalgono a uno spreco del tentativo di trasmissione. Nell'esempio in figura, se l'arbitraggio funzionasse male, le trasmissioni dei dispositivi MAC1 e MAC4 si scontrerebbero originando una collisione distruttiva dei segnali.

Come rendere il segmento affidabile?

Scopo: rendere il canale condiviso ad accesso multiplo un **canale affidabile**, senza errori di trasmissione dovuti a collisioni o interferenza
 Altri compiti del livello dei protocolli di livello 2, ([MAC/LLC](#))

- il pacchetto (frame) trasmesso sul mezzo trasmissivo è stato ricevuto correttamente?
 - Sì! Il destinatario invia un **frame di conferma** della corretta ricezione al mittente
 - No! il destinatario non fa nulla
 - se il mittente non riceve conferma entro un tempo fissato, trasmette di nuovo il frame
- i tentativi di trasmissione vengono ripetuti finché non si ottiene la conferma del successo



Un esempio di gestione della trasmissione a livello LLC.

La figura mostra la sequenza temporale di eventi gestiti dal livello LLC per la trasmissione affidabile di un frame di dati tra due dispositivi sullo stesso segmento di rete locale. Il frame di dati viene spedito dal dispositivo con indirizzo MAC1 al dispositivo con indirizzo MAC2 sul canale. Il mittente fa partire un timer dopo la trasmissione. Il ricevente MAC2 si accorge che il frame è destinato a lui, ma rileva errori sui bit ricevuti, per cui non fa nulla (non passa il frame ai livelli superiori) e non invia la conferma a MAC1. Allo scadere del timer, MAC1 verifica che non ha ricevuto conferma, per cui ripete da capo la trasmissione, e fa

ripartire il timer. Questa volta MAC2 riceve correttamente il frame e spedisce a MAC1 un frame di conferma. MAC1 riceve il frame di conferma e solo ora considera terminata con successo la trasmissione del frame di dati.

Scopo dei protocolli del livello 2 (MAC/LLC) è nascondere ai livelli superiori i dettagli del mezzo fisico, e mostrare il canale condiviso sul segmento di rete locale come se si trattasse di un canale affidabile, senza alcun errore di trasmissione. A tal fine il frame di dati viene delimitato mediante particolari etichette di bit, poste all'inizio e alla fine del frame, e viene arricchito con altri campi di dati utili al protocollo.

La trasmissione di un frame procede per tentativi, fino alla ricezione di una conferma (un frame di conferma) da parte del destinatario. Quello qui illustrato è solo un meccanismo semplice per realizzare trasmissione affidabile, tra quelli possibili.

La figura mostra la sequenza temporale di eventi gestiti dal livello 2 (MAC/LLC) per la trasmissione affidabile di un frame di dati tra due dispositivi sulla stessa rete locale. Il frame di dati viene spedito dal dispositivo con indirizzo MAC1 al dispositivo con indirizzo MAC2 sul mezzo trasmissivo. Il mittente fa partire un timer dopo la trasmissione. Il ricevente MAC2 si accorge che il frame è destinato a lui, ma rileva errori sui bit ricevuti, per cui non fa nulla (non passa il frame ai livelli superiori) e non invia la conferma a MAC1. Allo scadere del timer, MAC1 verifica che non ha ricevuto conferma, per cui ripete da capo la trasmissione, e fa ripartire il timer. Questa volta MAC2 riceve correttamente il frame e spedisce a MAC1 un frame di conferma. MAC1 riceve il frame di conferma e solo ora considera terminata con successo la trasmissione del frame.

Ai livelli dei protocolli superiori al livello due tutto ciò viene nascosto, e appare solamente la trasmissione corretta del frame sul segmento di rete.

Tecnologie per schede di rete: 3 esempi

Tre esempi di protocolli di livello due, che possono essere usati in alternativa tra loro per le schede di rete di segmenti di rete locale, sono:

- **Ethernet**
 - Molto usato in reti locali cablate
 - La scheda di rete ascolta il canale e trasmette solo se nessuno sta già trasmettendo
 - Se viene rilevata una collisione la trasmissione è interrotta per riprovare più tardi
- **IEEE 802.11 (Wi-Fi)**
 - È alla base delle reti locali senza fili
 - La scheda di rete ascolta il canale e trasmette solo se nessuno sta già trasmettendo
 - si cerca di prevenire le collisioni dilazionando le trasmissioni nel tempo
- **Token ring**
 - usata se i dispositivi sono collocati su topologia ad anello cablato
 - Esiste un frame detto token che viene passato come un “testimone” tra i dispositivi
 - Solo chi detiene il token ha diritto di trasmettere, poi deve passare il token

o

Vengono brevemente forniti tre esempi molto famosi di protocolli Standard di livello due (MAC), che possono essere adottati in alternativa, a seconda del tipo di mezzo fisico, del tipo di topologia, nella realizzazione di segmenti di rete locale.

La scelta di una tecnologia indica il tipo di schede di rete dovranno essere utilizzate per la connessione al segmento di rete.

Il più famoso è il protocollo Ethernet che da il nome a una vasta serie di schede di rete che implementano la sua definizione, per l'utilizzo in reti locali basate su mezzo fisico cablato.

L'idea di base di Ethernet, per ridurre le collisioni dei segnali, è di adottare il principio dell'ascolto del canale, prima di ogni trasmissione. Se nessuno sta già trasmettendo, allora la trasmissione può essere iniziata senza collisione con le trasmissioni in atto. Siccome può accadere che due schede di rete possano iniziare allo stesso istante le rispettive trasmissioni, occorre trovare una soluzione all'insorgere di possibili collisioni. La scheda di rete trasmittente è in grado di rilevare le collisioni in atto durante la trasmissione, e in tal caso interrompe immediatamente il tentativo di trasmissione.

Il tentativo verrà tentato da capo, dopo un attesa di tempo casuale, variabile da scheda a scheda.

Una tecnica simile a Ethernet viene adottata nelle reti senza fili (wireless), ad esempio in reti Wi-Fi conformi allo Standard IEEE 802.11. Il problema principale in reti senza fili è dato dall'impossibilità pratica di realizzare la rilevazione di collisioni in atto durante la fase di trasmissione. La tecnica si basa sulla prevenzione delle collisioni, dilazionando nel tempo i tentativi di accesso.

Il protocollo Token Ring è un protocollo MAC concepito per reti locali con topologia ad anello.

L'accesso è regolato per mezzo di un frame speciale, detto Token, che viene passato (trasmesso), come se fosse un "testimone", ciclicamente tra tutti i dispositivi in rete. Solo chi detiene il token ha diritto di trasmettere sul canale, evitando il rischio di collisione, dopodiché deve trasmettere il token alla stazione successiva nell'anello.

Comporre segmenti in reti locali

Dispositivi utili a **comporre reti locali** di calcolatori, agendo al **livello fisico** (1) e **MAC/LLC** (2):

- **Repeater** (ripetitore): livello fisico (1)
 - I segnali trasmessi sul mezzo fisico degradano con la distanza
 - Limite massimo alla lunghezza di un segmento di rete Ethernet: 100-200 metri
 - Un repeater è un dispositivo che amplifica e rigenera il segnale ricevuto
 - collega due o più segmenti di rete (con stessa tecnologia MAC)
 - estende la lunghezza dei segmenti di rete locale
- **Hub** (perno di una ruota a raggi): livello fisico (1) (detto anche repeater multiporta)
 - dispositivo concentratore e punto di contatto delle connessioni (a stella)
- **Bridge** (ponte): livello MAC/LLC (2)
 - connette segmenti di una rete locale con tecnologie MAC diverse
 - Es. connette un segmento Ethernet a un segmento Token Ring
 - Traduce i frame ricevuti da un segmento nel formato frame dell'altro segmento
 - Ri-trasmette il frame tradotto usando il protocollo MAC opportuno
- **Switch** (commutatore): livello MAC/LLC (2)
 - Analogico al bridge, ma permette di connettere molti segmenti (fino a 10-12)
 - Ha capacità di filtrare e inviare i frame sul segmento giusto, leggendo l'indirizzo MAC del destinatario del frame

o

A questo punto esistono i presupposti per introdurre alcuni dispositivi che possono essere usati per comporre ed estendere una rete locale di calcolatori, unendo segmenti di rete altrimenti separati.

Il primo dispositivo è il ripetitore (repeater). Siccome i segnali emessi su qualsiasi mezzo fisico si degradano al crescere della distanza percorsa, esiste un limite massimo per la lunghezza di un segmento di rete. Ad esempio, un segmento Ethernet, può variare dai 100 ai 200 metri. Un repeater è un dispositivo che agendo solo a livello fisico, amplifica e rigenera il segnale ricevuto verso un prolungamento del segmento di rete. Mediante un repeater è possibile collegare due segmenti di rete aventi la stessa tecnologia a livello MAC, ed estendere la lunghezza dei segmenti di rete locale.

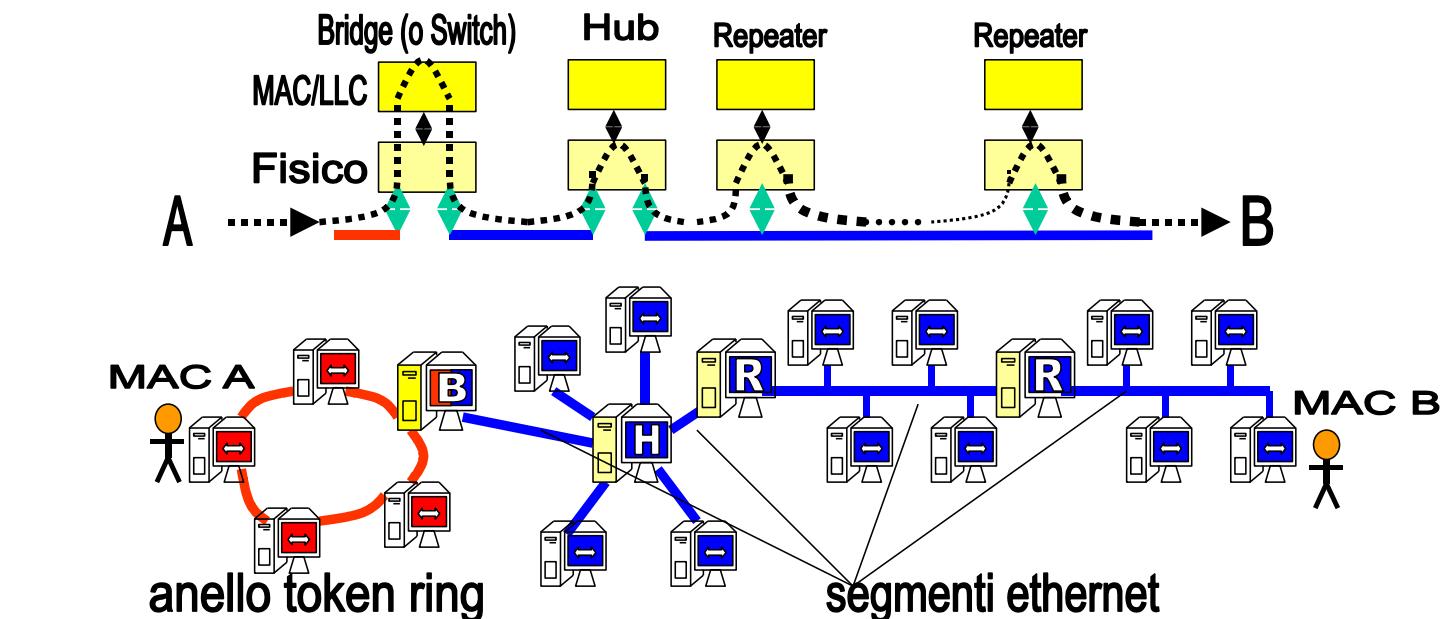
Un Hub (che significa perno di una ruota a raggi) è un altro dispositivo che agisce solo a livello fisico. Esso realizza il punto centrale di connessione, detto concentratore, dei segmenti di una rete locale con topologia a stella. In pratica si tratta di un ripetitore (repeater) con tante connessioni entranti e uscenti.

Un Bridge (ponte) è invece un dispositivo che agisce anche da traduttore a livello due (MAC/LLC). Un bridge permette di connettere segmenti di una stessa rete locale ma con tecnologie e MAC diversi tra loro (ad esempio un segmento Ethernet con uno Token Ring). I bridge fanno quindi da traduttori dei frame nei formati richiesti dal livello MAC di ogni segmento connesso al bridge, e provvedono alla trasmissione su segmenti diversi adottando il protocollo MAC opportuno.

Uno Switch (commutatore) è un dispositivo di livello due (MAC/LLC) analogo al bridge. Al contrario del bridge, esso permette di connettere un numero maggiore di segmenti diversi (fino a 10 o 12). I Bridge sono dotati della capacità di filtrare e instradare opportunamente i frame di dati sul segmento opportuno, osservando sui frame le informazioni di indirizzo MAC del dispositivo destinatario.

Un esempio completo di rete locale

Diversi segmenti di rete locale sono collegati attraverso bridge B (o switch), hub H e repeater R.



Un esempio di rete locale composta da segmenti ethernet e token ring

La figura mostra un esempio di rete locale composta da un segmento token ring colorato in rosso (l'anello a sinistra) e da segmenti ethernet colorati in blu (tutti gli altri). A sinistra, un calcolatore dotato di dispositivo di rete token ring con indirizzo MAC A è connesso a un anello di rete token ring insieme ad altri 3 calcolatori e un bridge (oppure uno switch) identificato dal colore giallo (per indicare che agisce a livello MAC/LLC) e dalla lettera B sullo schermo. Lo schermo è bicolore (rosso-blu) per denotare che il bridge agisce da collegamento e traduttore dei frame tra segmenti token ring (anello rosso) ed ethernet (blu). Il bridge B è connesso a un hub (H) dal quale partono sei segmenti ethernet, di tipo punto a punto, verso altrettanti calcolatori. Uno di questi calcolatori, identificato dalla lettera R è un repeater, che propaga e amplifica i segnali verso un successivo segmento ethernet con topologia a bus, sul quale esistono quattro calcolatori. Al termine del bus esiste un nuovo repeater R, che propaga e amplifica i segnali verso un

Reti di calcolatori - Intro

ultimo segmento ethernet con topologia a bus, al quale è collegato un calcolatore dotato di scheda ethernet con indirizzo MAC B. Al di sopra dei dispositivi citati viene rappresentato il cammino logico di un frame trasmesso dal calcolatore con MAC A al calcolatore con MAC B, passando per i segmenti, le interfacce e i livelli dei protocolli opportuni. In particolare, il bridge B è l'unico elemento nel quale i frame trasmessi passano fino al livello 2. Nei rimanenti dispositivi hub e repeater, i frame sono semplicemente ricevuti e ri-trasmessi su segmenti diversi, al livello fisico.

La figura mostra un esempio di rete locale composta da diversi segmenti: un segmento con topologia ad anello e protocollo MAC di tipo token ring colorato in rosso e da vari segmenti ethernet, con topologia a stella e a bus, colorati in blu. A sinistra, un calcolatore dotato di dispositivo di rete token ring con indirizzo MAC A è connesso a un anello di rete token ring insieme ad altri 3 calcolatori e insieme a un bridge (oppure uno switch) identificato dal colore giallo (per indicare che agisce a livello MAC/LLC) e dalla lettera B sullo schermo. Il bridge B agisce da collegamento e traduttore dei frame tra il segmento token ring (rosso) ed il successivo segmento ethernet (blu). Il bridge B ha quindi due connettori di rete: uno token ring e uno ethernet. Il segmento ethernet del bridge B è connesso a un Hub (H) dal quale partono sei segmenti ethernet, di tipo punto a punto, verso altrettanti calcolatori. Uno di questi calcolatori, identificato dalla lettera R è un repeater, che propaga e amplifica i segnali verso un successivo segmento ethernet con topologia a bus, sul quale esistono quattro calcolatori. Al termine del bus esiste un nuovo repeater R, che propaga e amplifica i segnali verso un ultimo segmento ethernet con topologia a bus, al quale è collegato un calcolatore dotato di scheda ethernet con indirizzo MAC B.

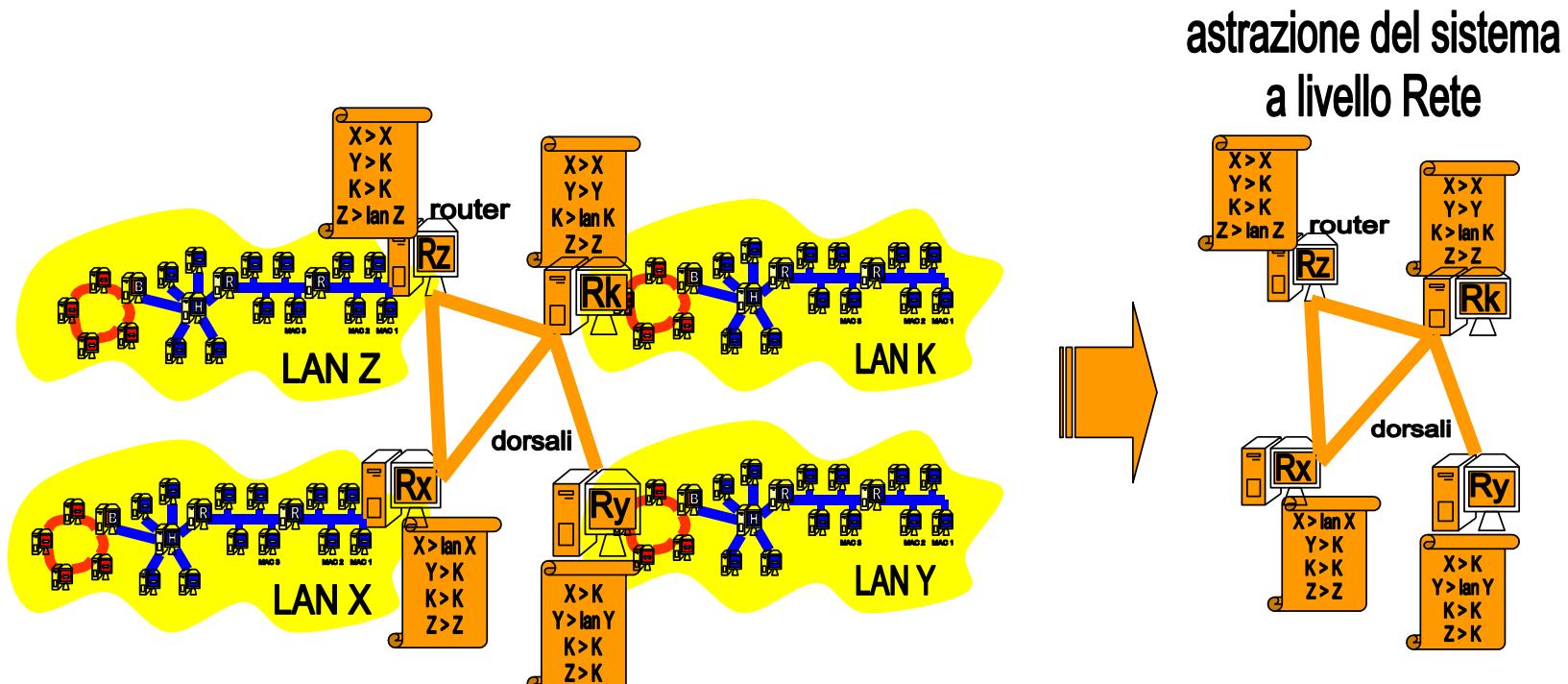
Al di sopra dei dispositivi citati viene rappresentato il cammino logico di un frame trasmesso dal calcolatore con MAC A al calcolatore con MAC B, passando per i segmenti, i connettori di rete, e i livelli dei protocolli opportuni. In particolare, il bridge B è l'unico elemento nel quale il frame trasmesso sul segmento token ring sale fino al livello 2, per essere tradotto e ritrasmesso sul segmento uscente adottando il nuovo protocollo MAC ethernet. Nei rimanenti dispositivi hub e repeater, i frame sono semplicemente ricevuti e ri-trasmessi sui segmenti uscenti. Se il bridge avesse dovuto connettere più di due segmenti diversi, allora si sarebbe utilizzato uno switch, che svolge l'attività del bridge gestendo più interfacce di rete e protocolli.

Dovrebbe essere chiaro a questo punto come sia possibile connettere diversi segmenti di rete locale, e gestire la trasmissione di frame di dati tra due dispositivi qualsiasi di una rete locale, semplicemente identificando i dispositivi attraverso il loro indirizzo MAC.

Reti di calcolatori - Intro

Reti di reti e Internetworking

Le reti locali sono connesse attraverso collegamenti organizzati in modo gerarchico, basati su calcolatori rappresentanti della rete locale (router) a loro volta collegati da linee dati veloci o dorsali (backbone).



Un esempio di rete di reti locali e router collegati da dorsali, e l'astrazione dello stesso sistema al terzo livello (Rete) visto dai router.

La figura mostra a sinistra quattro reti locali, X,Y,Z e K dotate ognuna di un dispositivo router (intradatatore) Rx, Ry, Rx e Rk, rispettivamente. Ogni router, oltre ad essere connesso alla propria rete locale, è connesso attraverso linee di comunicazione, dette dorsali, ad altri router. I router e le dorsali sono di colore scuro (arancio), per evidenziare che si tratta di componenti che agiscono al livello 3 (rete) della gerarchia

Reti di Calcolatori - introduzione

ISO/OSI. Una dorsale connette i router Rx e Rz, un'altra i router Rz e Rk, un'altra i router Rx e Rk e infine un'altra i router Rk e Ry. Ogni router è inoltre istruito da una tabella su quale sia il primo router intermedio per comunicare con ogni altro router. Ad esempio, il router Rz può comunicare indirettamente con Ry per mezzo del router intermedio Rk. A Destra viene mostrata l'astrazione del sistema che appare ad ogni router, ovvero al terzo livello (Rete).

Se i milioni di calcolatori oggi connessi a Internet fossero tutti organizzati secondo i protocolli e gli schemi visti finora per le reti locali, la comunicazione tra due calcolatori su Internet richiederebbe di passare per migliaia di calcolatori intermedi, switch, bridge, segmenti di rete, ognuno dei quali aggiungerebbe ritardi di gestione, complessità, rischi di errore. Il problema dell'instradamento dei frame (routing), ovvero il decidere da che parte o su che segmento deve essere inoltrato un frame per raggiungere il destinatario finale, richiederebbe in ogni dispositivo una lista completa (tabella di instradamento) di tutti gli indirizzi MAC dei dispositivi nel mondo, con a fianco l'indicazione della direzione di inoltro. Ovviamente questo limiterebbe in modo critico la scalabilità e la crescita di Internet.

Una soluzione semplice consiste nell'elezione di un rappresentante per ogni rete locale X (il router di X), incaricato di ricevere tutti i pacchetti dati destinati a uno dei calcolatori della rete locale (es. mac1 di X, mac2 di X, ecc.). Ricevuti i pacchetti destinati alla rete locale, il router potrebbe occuparsi di recapitare alla rete locale i pacchetti, come se si trattasse di un frame a livello MAC/LLC destinato all'indirizzo MAC del destinatario. Allo stesso modo, ogni router dovrebbe farsi carico di inoltrare tutti i pacchetti uscenti dalla propria rete locale, verso i router delle reti di destinazione. Per rispettare le direttive dettate dallo standard ISO/OSI, il livello di indirizzamento e la gestione dell'instradamento dei pacchetti tra i router vengono gestiti al terzo livello (rete) della gerarchia dei protocolli di Internet.

Per ciò che riguarda i router, tuttavia, lo scambio diretto tra router di pacchetti destinati alle rispettive reti locali potrebbe ridurre molto la complessità dell'instradamento. I router comunicano quindi attraverso collegamenti dati molto veloci, dette dorsali (backbone). Ogni router deve ricordare in una tabella di instradamento (forwarding table) solo quale sia il primo router intermedio per raggiungere ogni altro router. La visione del sistema al terzo livello (Rete) da parte dei router è quindi simile alla visione che appare a destra nella figura. Si nota come tutti i dettagli delle reti locali siano di fatto nascosti dai router a questo livello.

Livello rete: Internet protocol (IP)

Inizia a questo punto la trattazione degli aspetti di gestione del **livello rete**. Il concetto di rete non si limita ora alla sola rete locale (LAN) ma si estende alla rete di reti globale (**Internet**).

- Livello Rete (Network) per Internet:
 - protocollo Internet (**Internet Protocol, IP**)
 - nuovo tipo di indirizzamento globale e gerarchico (**indirizzamento IP**)
 - fornisce indirizzi alla rete locale e ai suoi nodi
 - instradamento dei pacchetti dal mittente al destinatario finale (**forwarding**)
 - Servizio di comunicazione di tipo **connectionless**
 - nuovi dispositivi amministratori del livello tre: **router**
 - tabelle di instradamento che illustrano la topologia della rete (livello Rete)
 - nasconde dettagli interni delle LAN al livello Rete
 - protocolli di aggiornamento delle tabelle di instradamento (**routing**)
 - **frammentazione** dei dati da spedire in pacchetti
 - **busta del pacchetto** di livello rete con gli indirizzi di mittente e destinatario



La collocazione del livello rete, i servizi in esso implementati e il protocollo IP.

La figura mostra l'architettura a livelli dei protocolli di Internet visti finora. Al secondo livello troviamo i protocolli MAC, tra i quali Ethernet, Token ring, 802.11, e protocolli LLC, tra i quali HDLC e PPP. I servizi indicati al livello MAC/LLC sono essenzialmente l'accesso al mezzo trasmittivo e il controllo degli errori di trasmissione. Al terzo livello (rete) troviamo il protocollo IP. Sono indicati i servizi supportati dal protocollo IP a livello rete: frammentazione, indirizzamento e instradamento dei pacchetti dati.

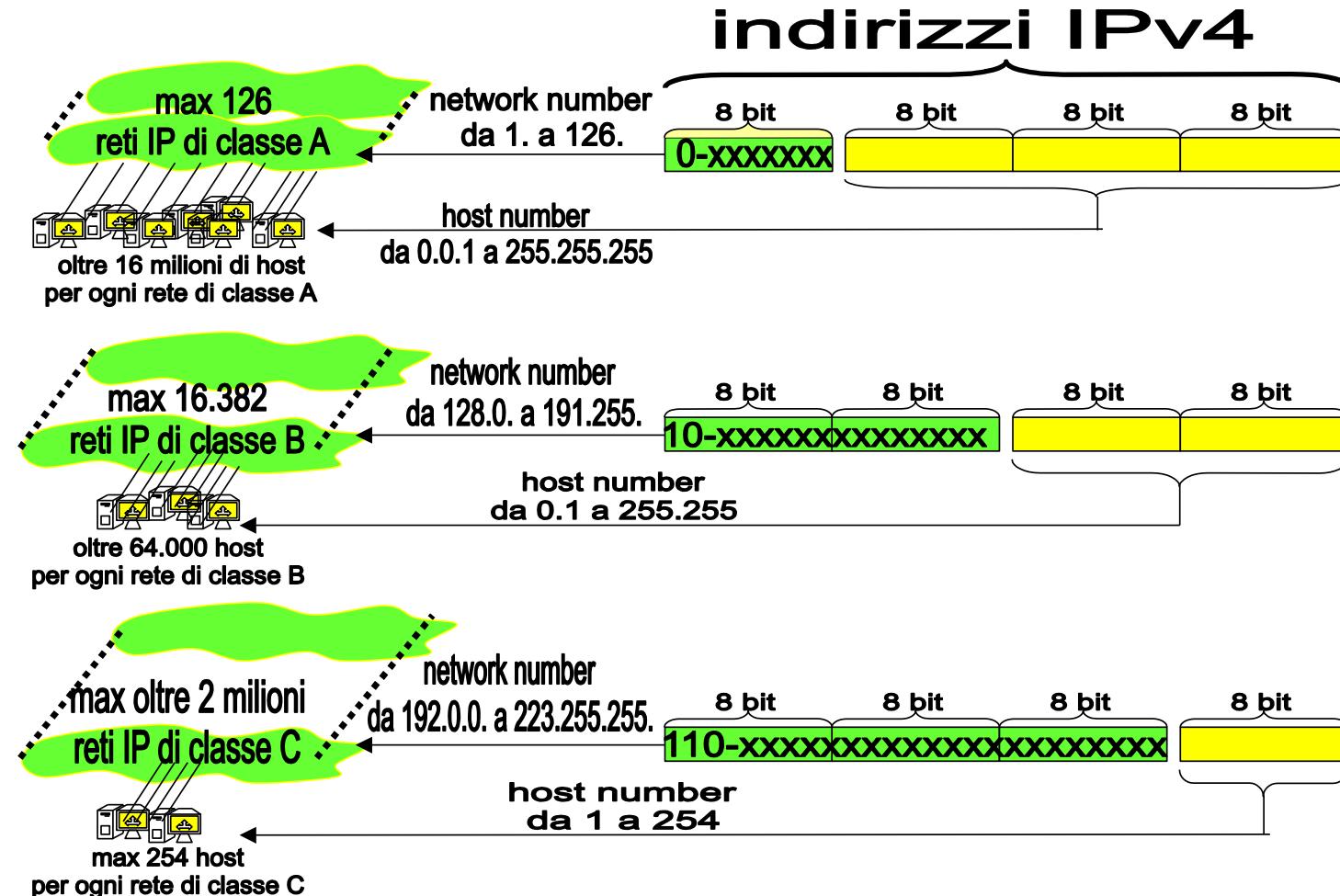
Reti di Calcolatori - introduzione

Inizia a questo punto la trattazione degli aspetti di gestione del terzo livello dell'architettura dei protocolli di Internet: il livello Rete (Network). Il concetto di rete non si limita ora alla sola rete locale (LAN) ma si estende alla rete di reti globale (Internet). Il livello rete di Internet si basa sul protocollo IP (Internet Protocol). Il protocollo IP definisce un nuovo schema di indirizzamento globale e gerarchico, che permette di identificare univocamente tutti i dispositivi di rete e allo stesso tempo la loro rete locale di appartenenza. Gli indirizzi usati permettono di identificare intere reti locali come un riferimento singolo nella gestione dell'instradamento dei pacchetti. Questo fatto semplifica molto la visione della rete che appare al livello Rete (si veda la figura della diapositiva precedente). Al protocollo IP, si possono associare protocolli di instradamento dei pacchetti dal mittente al destinatario finale (forwarding), originando servizi di trasmissione a pacchetto di tipo connectionless. Il protocollo IP richiede l'adozione di nuovi dispositivi amministratori dell'inoltro dei pacchetti a livello Rete, detti router. I router sono forniti di tabelle di instradamento che illustrano la topologia della rete vista al livello dei router stessi (quindi non è necessario conoscere gli indirizzi dei calcolatori di una LAN al di fuori della LAN stessa. I router devono implementare protocolli di aggiornamento delle tabelle di instradamento (detti protocolli di routing). Ulteriore compito dei router è la gestione della frammentazione dei dati da spedire nei pacchetti, e la creazione della busta di livello rete con gli indirizzi del router mittente e destinatario di ogni pacchetto inoltrato.

Vediamo ora come è definito l'indirizzamento caratteristico del protocollo IP di Internet: IPv4.

Indirizzamento IPv4

- Il protocollo IP definisce una nuova specie di indirizzi: gli indirizzi IP
- un indirizzo IP viene associato a una e una sola interfaccia di rete (scheda di rete)
 - associazione univoca tra indirizzo MAC e indirizzo IP
 - IP statico (sempre lo stesso) e IP dinamico (può cambiare l'associazione MAC-IP)
- Gli indirizzi IP attualmente usati si riferiscono al protocollo IP versione 4, (**IPv4**)
 - Un indirizzo IPv4: 32 bit (4 Byte) = sequenza di 4 valori decimali separati da punto
 - Ogni valore decimale può essere compreso tra i valori 0 e 255
 - Esempio di indirizzo IP valido: 130.136.25.1
 - Indirizzo IP è sempre composto da due parti:
 - numero della rete IP alla quale appartiene la scheda (**network number**)
 - numero dell'interfaccia di rete (**host number**) all'interno della rete
 - il valore dell'indirizzo IP determina la **classe della rete**: A,B,C



Una classificazione delle tre classi di reti, A, B e C, associate agli indirizzi IPv4.

La figura illustra uno schema di associazione degli indirizzi IPv4 alla classe di rete relativa. Le reti di classe A sono al massimo 126 e ognuna può contenere oltre 16 milioni di host. Per le reti di classe A, il byte di indirizzo più significativo (a sinistra) ha sempre il primo bit uguale a zero, e può assumere i valori da 1 a 126 (network number). I tre byte rimanenti possono assumere oltre 16 milioni di combinazioni, ognuna associabile a un host della rete. Le reti di classe B sono al massimo 16.382 e ognuna può contenere fino a oltre 64.000 host. Per le reti di classe B, i due byte di indirizzo più significativi (a sinistra) hanno sempre i primi due bit uguali alla coppia (uno,zero), e possono assumere i valori da 128.0. a 191.255. (network number).

Reti di Calcolatori - introduzione

number). I due byte rimanenti possono assumere oltre 64.000 combinazioni, ognuna associabile a un host della rete. Le reti di classe C sono oltre 2 milioni, e ognuna può contenere fino a 254 host. Per le reti di classe C, i tre byte di indirizzo più significativi (a sinistra) hanno sempre i primi tre bit uguali alla terna (uno,uno,zero), e possono assumere i valori da 192.0.0 a 223.255.255 (network number). Il byte rimanente può assumere 254 combinazioni utili, ognuna associabile a un host della rete.

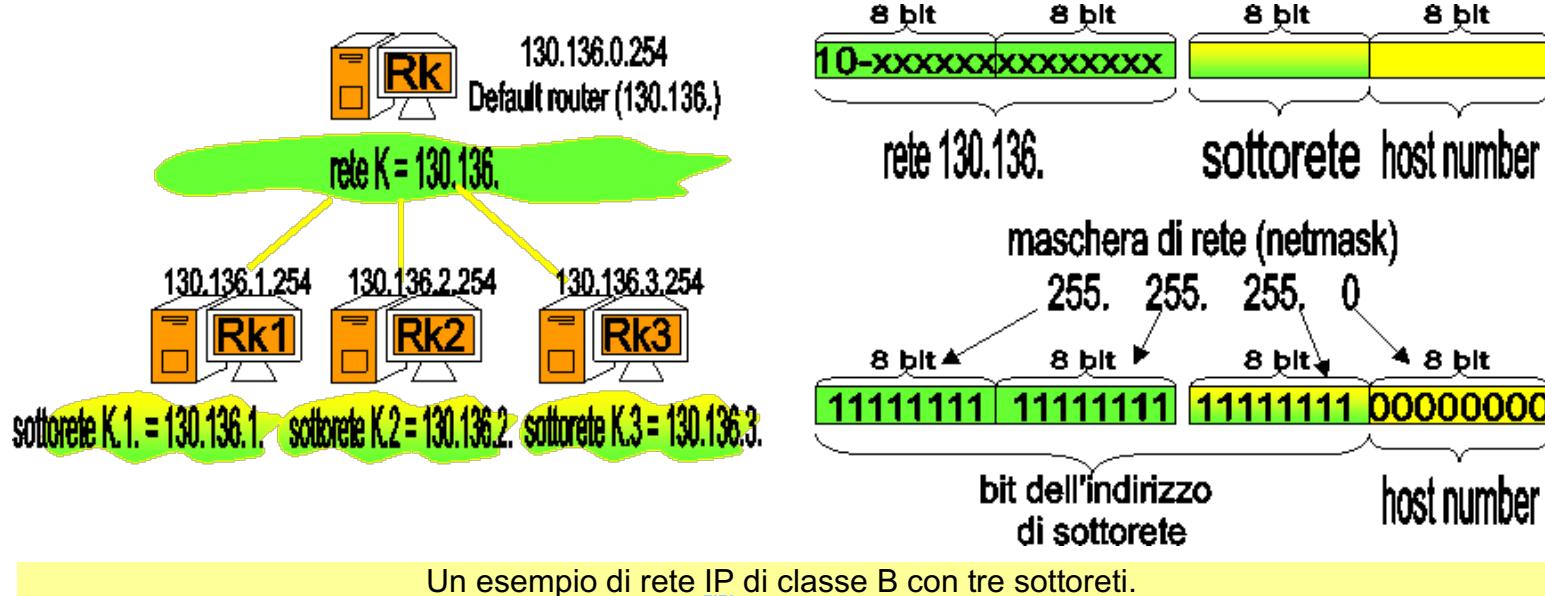
Il protocollo IP definisce una nuova specie di indirizzi: gli indirizzi IP. Gli indirizzi IP attualmente usati si riferiscono al protocollo IP versione 4, (IPv4). Un indirizzo IP viene associato a una e una sola interfaccia di rete (scheda di rete), e più precisamente esiste un rapporto diretto tra un indirizzo MAC dell'interfaccia di rete e un indirizzo IP, dal momento che l'interfaccia è collegata alla rete Internet. Se un calcolatore dispone di più di una scheda di rete, ed è contemporaneamente collegato a Internet mediante entrambe le interfacce, allora il calcolatore potrebbe essere contemporaneamente identificato da più di un indirizzo IP. Se l'associazione univoca tra indirizzo MAC dell'interfaccia di rete e l'indirizzo IP rimane sempre lo stesso, allora si parla di IP statico. In caso contrario, se può cambiare l'associazione MAC-IP a seconda di vari fattori, si parla di IP dinamico. Un indirizzo IPv4 è un valore espresso su 32 bit (4 Byte), e può essere espresso anche come sequenza di 4 valori decimali, separati da punto. Ogni valore decimale può essere compreso tra i valori 0 e 255. Un esempio di indirizzo IP valido è (130.136.250.1). Ogni indirizzo IP è sempre composto da due parti: numero della rete IP alla quale appartiene la scheda (network number), e numero dell'interfaccia di rete (host number) all'interno della rete IP. Sono definite tre classi di reti IP, che si differenziano sulla base del numero massimo di host supportabili. Il valore dell'indirizzo IP determina la classe della rete: A,B,C (vedere figura).

Le reti di classe A sono al massimo 126 e ognuna può contenere fino a oltre 16 milioni di host. Per le reti di classe A, il byte di indirizzo più significativo (a sinistra) ha sempre il primo bit uguale a zero, e può assumere i valori da 1 a 126 (network number) rispetto ai 128 valori possibili. I tre byte rimanenti possono assumere oltre 16 milioni di combinazioni, ognuna associabile a un host della rete. Le reti di classe B sono al massimo 16.382 e ognuna può contenere fino a oltre 64.000 host. Per le reti di classe B, il network number è dato dai due byte di indirizzo più significativi (a sinistra), che hanno sempre i primi due bit uguali alla coppia (uno,zero). I network number di classe B possono assumere i valori da 128.0. a 191.255. I due byte rimanenti (host number) possono assumere oltre 64.000 combinazioni, ognuna associabile a un host della rete. Le reti di classe C sono oltre 2 milioni, e ognuna può contenere fino a 254 host. Per le reti di classe C, i tre byte di indirizzo più significativi (a sinistra) rappresentano il network number, e hanno sempre i primi tre bit uguali alla terna (uno,uno,zero). I network number di classe C possono assumere i valori da 192.0.0 a 223.255.255. Il byte rimanente (host number) può assumere 254 combinazioni utili, su 256 possibili, ognuna associabile a un host della rete.

Sottoreti (subnetwork)

Reti IP e router sono organizzati secondo una gerarchia, basata sul concetto di rete e sottorete

- **indirizzo IP** spezzato in due componenti logiche mediante **maschera di rete (netmask)**
 - indirizzo di sottorete (subnetwork): bit uguali a uno nella netmask
 - host number dell'host appartenente alla sottorete: bit uguali a zero nella netmask
- gerarchia di sottoreti, ognuna delle quali è amministrata da un **router** (il **default router**).
- Esempio: rete di classe B 130.136. con 256 sottoreti: **netmask** 255.255.255.0
 - es. 130.136.1. è la sottorete 1, 130.136.2 è la sottorete 2...
 - es. 130.136.1.22 è l'**host** 22 della sottorete 1, 130.136.3.48 è l'**host** 48 della sottorete 3...
- Indirizzo IP, netmask e default router sono **informazioni di configurazione del livello rete**



Reti di Calcolatori - introduzione

La figura mostra a sinistra uno schema gerarchico di strutturazione di una rete di classe B. A partire dall'alto troviamo il router principale (default router) della rete 130.136, il cui IP è 130.136.0.254. Alla rete 130.136 appartengono anche tre router subordinati, con IP 130.136.1.254, 130.136.2.254, 130.136.3.254 rispettivamente amministratori delle sottoreti (130.136.1.), (130.136.2.) e (130.136.3.). Per istruire ogni router subordinato sulla dimensione e sull'interpretazione degli indirizzi IP da amministrare, ogni router subordinato deve essere fornito di una maschera di rete (netmask), evidenziata a destra. La maschera di rete serve solo a definire quali bit degli indirizzi IP vadano interpretati come numero della sottorete (indirizzo della sottorete), cioè i bit uguali a uno, partendo da sinistra. I bit della maschera uguali a zero (a destra) servono invece a descrivere quali bit degli indirizzi IP vadano considerati come il numero dell'host (host number) appartenente alla sottorete. Nell'esempio i 3 Byte a sinistra della maschera sono tutti a uno, quindi i primi 24 bit dell'indirizzo IP rappresentano il numero di sottorete, come risulta dal disegno di sinistra.

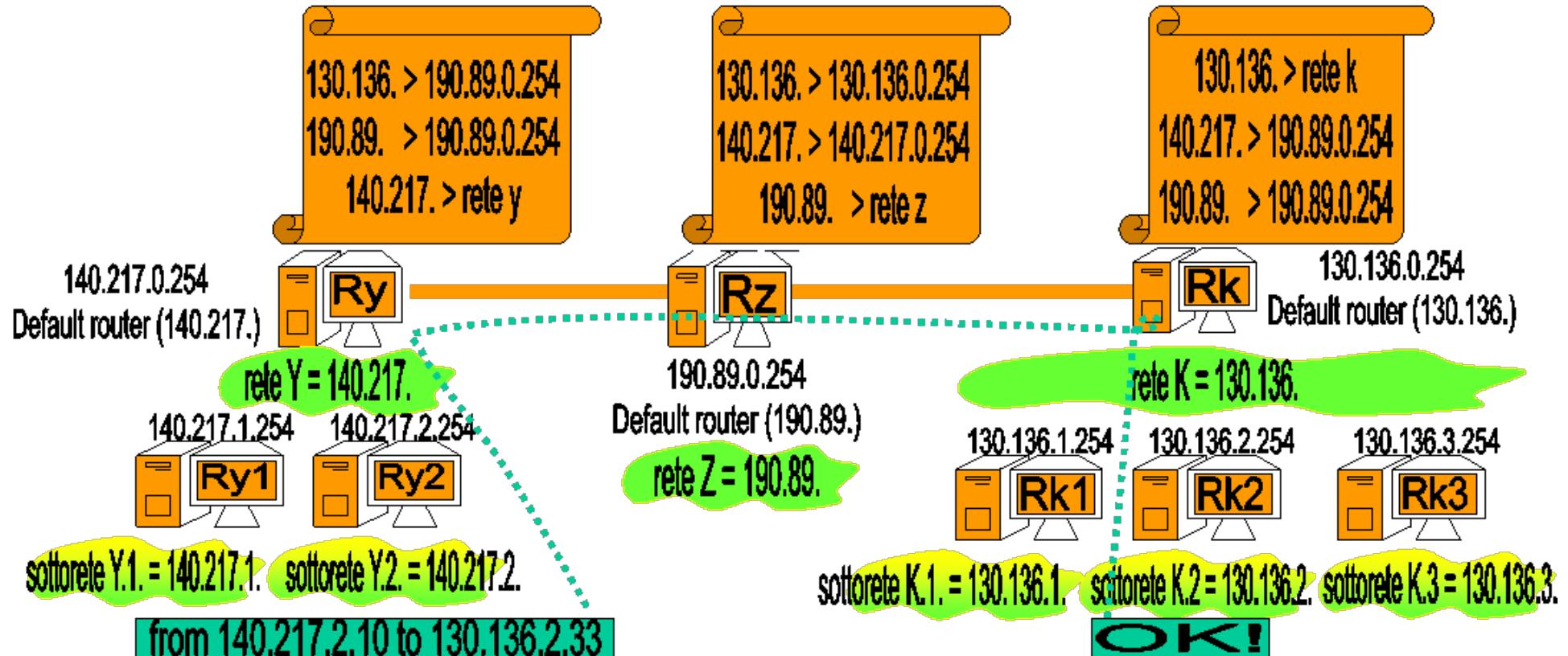
I router e le reti IP sono organizzate secondo una gerarchia, basata sul concetto di rete e sottorete. Ogni indirizzo IP può essere spezzato in due componenti logiche attraverso la maschera di rete (netmask). La prima componente logica è un indirizzo di sottorete (subnetwork), che corrisponde ai bit con la stessa posizione dei bit uguali a uno nella netmask. La seconda componente logica è il campo host number che identifica gli host appartenenti alla sottorete, e corrisponde ai bit corrispondenti ai bit uguali a zero nella netmask. In questo modo è possibile creare una gerarchia di sottoreti, ognuna delle quali è amministrata da un router (il default router). Esempio: data la rete di classe B 130.136, per semplicità decidiamo di considerare possibili 256 sottoreti: netmask 255.255.255.0. Il numero della sottorete è quindi fornito dai primi tre byte dell'indirizzo IP, es. 130.136.1. è la sottorete 1, 130.136.2. è la sottorete 2, mentre ad esempio 130.136.1.22 è l'host 22 della sottorete 1, 130.136.3.48 è l'host 48 della sottorete 3, e così via.

La figura mostra a sinistra uno schema gerarchico di strutturazione di una rete di classe B. A partire dall'alto troviamo il router principale (default router) della rete 130.136, il cui indirizzo IP è nell'esempio 130.136.0.254. Alla rete 130.136 appartengono anche tre router subordinati, con IP 130.136.1.254, 130.136.2.254, 130.136.3.254 rispettivamente amministratori delle sottoreti (130.136.1.), (130.136.2.) e (130.136.3.). Per istruire ogni router subordinato sulla dimensione e sull'interpretazione degli indirizzi IP da amministrare, ogni router subordinato deve essere fornito di una maschera di rete (netmask), evidenziata a destra. Per ogni dispositivo di rete o calcolatore connesso a una rete IP (es. Internet), la maschera di rete, l'indirizzo del default router e l'indirizzo IP, costituiscono i tre parametri fondamentali di configurazione del protocollo IP, e devono essere forniti, al momento della connessione, al livello IP.

Instradamento (forwarding) dei pacchetti

- esempio: host 140.217.2.10 spedisce pacchetto IP a host 130.136.2.33
 - pacchetto inviato da host 140.217.2.10 a host 130.136.2.33
 - raccolto dal default router della sottorete Ry2: 140.217.2.254
 - destinatario non appartiene alla sottorete: inoltrato verso il default router di livello superiore: 140.217.0.254
 - raccolto dal default router 140.217.0.254
 - destinatario 130.136.--: tabella di forwarding indica di inviare a 190.89.0.254: inoltrato
 - raccolto da 190.89.0.254
 - dest. 130.136.--: tabella di forwarding indica di inviare a 130.136.0.254: inoltrato
 - raccolto da 130.136.0.254
 - dest. 130.136.--: la tabella di forwarding indica che appartiene a questa rete (k)
 - inoltrato alla rete 130.136.--
 - raccolto dal router 130.136.2.254
 - la tabella indica che dest. appartiene alla sottorete Rk2: 130.136.2. : inoltrato
 - raccolto dall'host destinatario finale 130.136.2.33: OK!

Reti di Calcolatori - introduzione



Un esempio di instradamento su rete IP

La figura mostra un esempio di instradamento su rete IP. Esistono tre router Ry, Rz e Rk rispettivamente amministratori delle reti (140.217.), (190.89.), e (130.136.). Il router Ry è connesso a Rz, e Rz è connesso a Rk. Tale informazione risulta dalle tabelle di instradamento di Ry, Rz, Rk. La rete del router Ry include due sottoreti (140.217.1.) e (140.217.2), amministrate dai rispettivi default router 140.217.1.254 e 140.217.2.254 . La rete del router Rk include tre sottoreti (130.136.1.), (130.136.2.) e (130.136.3.), amministrate dai rispettivi default router 130.136.1.254, 130.136.2.254 e 130.136.3.254 . Un pacchetto IP spedito dall'host 140.217.2.10 all'host 130.136.2.33 deve compiere il seguente tragitto: passa per il default router di sottorete 140.217.2.254 che lo inoltra al default router di rete 140.217.0.254. Esso controlla la tabella di forwarding e scopre che per raggiungere la destinazione il pacchetto deve essere inoltrato al router intermedio 190.89.0.254. Il router intermedio riceve il pacchetto, verifica la propria tabella di forwarding, scopre che il prossimo destinatario intermedio è il router 130.136.0.254 e vi inoltra il pacchetto.

Reti di Calcolatori - introduzione

Il router 130.136.2.254 riceve il pacchetto e verifica che appartiene alla propria rete, inoltrando quindi il pacchetto internamente. Il router di sottorete 130.136.2.254 riceve il pacchetto e scopre che appartiene alla propria sottorete, inoltrandovi il pacchetto. Finalmente, l'host 130.136.2.33 riceve il pacchetto a lui destinato.

La figura mostra un esempio di instradamento su rete IP. Esistono tre router Ry, Rz e Rk rispettivamente amministratori delle reti (140.217.), (190.89.), e (130.136.). Il router Ry è connesso a Rz, e Rz è connesso a Rk. Tale informazione risulta dalle tabelle di instradamento di Ry, Rz, Rk. La rete del router Ry include due sottoreti (140.217.1.) e (140.217.2), amministrate dai rispettivi default router 140.217.1.254 e 140.217.2.254 . La rete del router Rk include tre sottoreti (130.136.1.), (130.136.2.) e (130.136.3.), amministrate dai rispettivi default router 130.136.1.254, 130.136.2.254 e 130.136.3.254 . Un pacchetto IP spedito dall'host 140.217.2.10 all'host 130.136.2.33 deve compiere il seguente tragitto: passa per il default router di sottorete 140.217.2.254 che, notando che il destinatario non appartiene alla sottorete, lo inoltra al default router del livello superiore di rete: 140.217.0.254. Il default router di rete controlla la propria tabella di forwarding e scopre che per raggiungere la destinazione il pacchetto deve essere inoltrato al router intermedio 190.89.0.254. Il router intermedio riceve il pacchetto, verifica la propria tabella di forwarding, scopre che il prossimo destinatario intermedio è il router 130.136.0.254, al quale inoltra il pacchetto. Il router 130.136.2.254 riceve il pacchetto e verifica che appartiene alla propria rete, inoltrando quindi il pacchetto internamente. Il router di sottorete 130.136.2.254 riceve il pacchetto e scopre che appartiene alla propria sottorete, inoltrando il pacchetto internamente. Finalmente, l'host 130.136.2.33 riceve il pacchetto a lui destinato.

Si possono notare alcuni aspetti importanti: malgrado il numero elevato di host che potrebbero essere parte del sistema considerato, il processo di instradamento permane molto semplice, composto da piccole e semplici operazioni di base. Le tabelle di instradamento sono limitate agli elementi che agiscono allo stesso livello, e quindi l'instradamento è gerarchico.

Routing

Il problema del **routing**: aggiornamento delle tabelle di **forwarding** dei **router**

- modifiche dei cammini per i dati nella rete
 - possibili soprattutto in reti senza fili a causa della mobilità degli **host**
 - causate da modifiche agli accordi di servizio tra gestori di sistemi autonomi (AS)
- le modifiche dei cammini rendono sbagliate le tabelle di **forwarding** dei **router**
 - i pacchetti possono andare perduti, o seguire strade diverse e arrivare disordinati
- occorrono reazioni da parte dei **router** per scoprire nuovi cammini
 - **protocolli (algoritmi) di routing:**
 - invio richieste: qualcuno conosce il modo per arrivare al destinatario?
 - Aggiornamento della tabella di **forwarding** con il cammino migliore
- Esempio di algoritmi di **routing** su Internet: Routing Information Protocol (RIP), Open Shortest Path First (OSPF), Border Gateway protocol (BGP)

Reti di Calcolatori - introduzione

•

Il problema del routing può essere definito come il problema di mantenere l'aggiornamento delle tabelle di forwarding in tutti i router della rete.

Questo problema può essere a volte molto complesso, a causa di frequenti modifiche forzate dei cammini per i pacchetti in rete. Le cause di tali modifiche possono essere dovute a molti fattori, ad esempio: la mobilità degli host in reti senza fili, guasti di mezzi trasmissivi, interruzione delle linee, guasti di router, nuove politiche e accordi per lo scambio dei dati tra gestori di dorsali e sistemi autonomi. Un sistema autonomo (AS) è sinonimo di una grossa rete, o una collezione di reti, soggetta a una comune politica di amministrazione. Gli accordi commerciali tra gestori di AS possono modificare i cammini consentiti per lo scambio dei pacchetti di dati. Per realizzare un parallelo intuitivo, gli AS si comportano come nazioni che permettano o meno il passaggio di pacchetti, analoghi a voli aerei, sul loro suolo nazionale.

Ogni volta che si verifica uno dei problemi citati, esistono router che hanno indicazioni errate nelle loro tabelle di forwarding. Tutto ciò può causare la perdita di pacchetti, oppure può determinare un disordine nell'arrivo di pacchetti che hanno seguito strade diverse. Ecco quindi una causa del servizio connectionless ottenuto dal livello rete basato solo sul protocollo IP.

I router hanno bisogno di aggiornare al più presto le loro tabelle, per evitare malfunzionamenti del servizio. I protocolli di routing hanno la funzione di richiedere e scambiare informazioni per trovare cammini alternativi (idealmente il cammino migliore tra le possibili alternative), tra mittenti e destinatari dei pacchetti, e consentire quindi l'aggiornamento delle tabelle di forwarding. A puro titolo informativo, si citano alcune sigle di protocolli di routing adottati in Internet: Routing Information Protocol (RIP), Open Shortest Path First (OSPF), Border Gateway protocol (BGP).

Protocollo ICMP

Internet Control Message Protocol (ICMP); protocollo dei **messaggi di controllo** su Internet

- Uno standard per definire la comunicazione di informazioni utili alla gestione di Internet
- ICMP è usato da host, router e gateway per scambiare informazioni di livello rete, usando pacchetti definiti con il protocollo IP
 - Notifica di errori di configurazione e gestione dei cammini e collegamenti
 - Rete di destinazione non raggiungibile (possibile interruzione di rete?)
 - Rete di destinazione sconosciuta (indirizzo di rete male specificato?)
 - Host destinazione non raggiungibile (host spento o scollegato?)
 - Host destinazione sconosciuto (indirizzo di host male specificato?)
 - Protocollo richiesto non disponibile (servizi non previsti)
 - Ricerca di un cammino alternativo per la destinazione (se esiste)

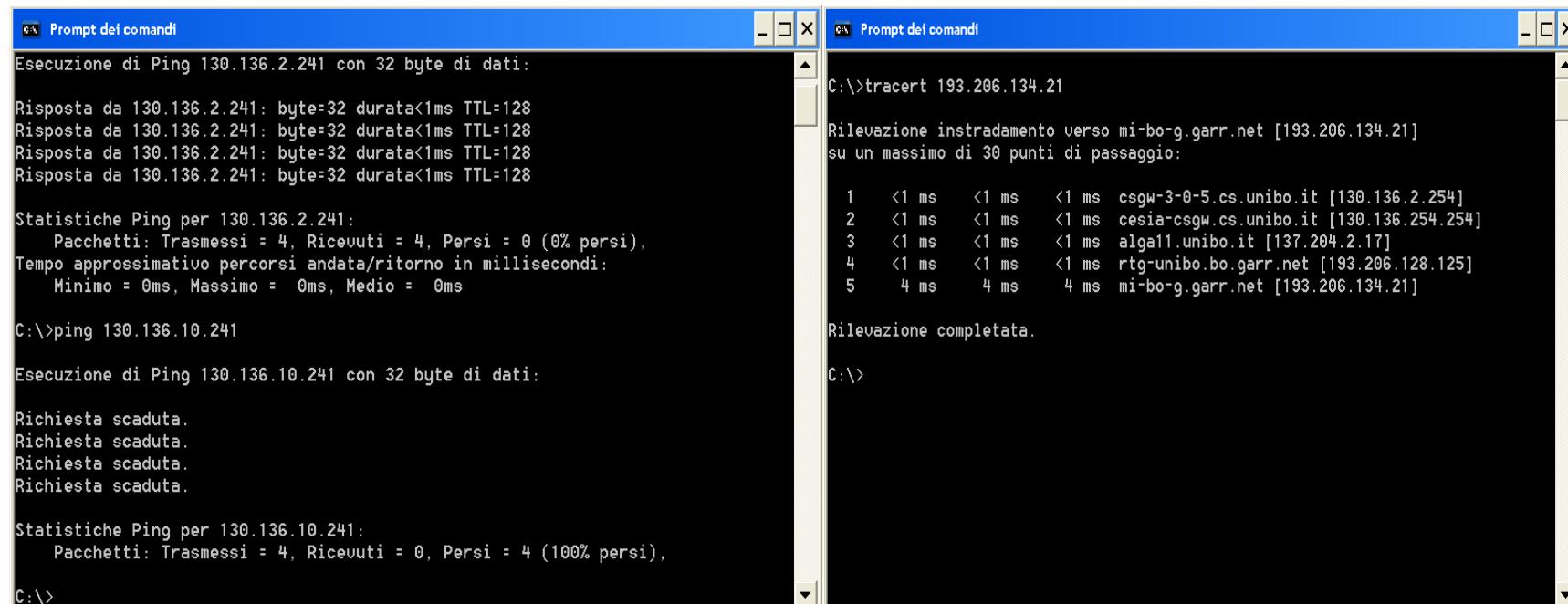
Reti di Calcolatori - introduzione

Vediamo ora un esempio di protocollo, basato sul protocollo IP, definito per supportare i messaggi di controllo per la gestione della rete al terzo livello dello stack ISO/OSI (Rete). Il protocollo in questione si chiama Internet Control Message Protocol (ICMP), ovvero protocollo dei messaggi di controllo su Internet. ICMP è un protocollo standard, e quindi può essere adottato da parte di tutti i dispositivi per fornire uno strumento generalizzato e compatibile. ICMP viene usato da semplici host, da router e persino da gateway (router speciali con ulteriori funzioni) per scambiare informazioni utili alla gestione del livello Rete. Le informazioni scambiate sono trasferite sotto forma di pacchetti IP.

Alcuni esempi di messaggi ICMP che possono essere scambiati indicano, ad esempio: situazioni di rete di destinazione irraggiungibile (possibile sintomo di problemi di routing, o di rottura di un router), rete di destinazione sconosciuta (possibile sintomo di indirizzo IP di rete male specificato), host di destinazione non raggiungibile (possibile sintomo che l'host sia spento o il cavo di connessione sia male collegato), host di destinazione sconosciuto (possibile sintomo di host number dell'indirizzo IP male specificato, malgrado la rete indicata esista e sia raggiungibile), protocollo richiesto non disponibile (sintomo di un tentativo di dialogo tra dispositivi male configurati, che non forniscono i servizi richiesti), ricerca di cammino alternativo (può essere usato per risolvere i problemi di routing).

Applicazioni basate su ICMP

- applicazione **PING**: test di verifica di connessione tra due host
 - esempio: eseguire “ping <indirizzo IP host2>” sull’host1
 - host1 invia richieste ICMP e host2 risponde (eco), se esiste ed è raggiungibile
 - viene calcolato il tempo di andata e ritorno delle richieste (Round Trip Time, RTT)
- applicazione **Traceroute**: mostra la sequenza di router attraversati da un pacchetto per arrivare all’host destinazione
 - esempio: eseguire “traceroute <indirizzo IP host2>” sull’host1
 - È realizzato con messaggi ICMP spediti in sequenza, verso distanze via via maggiori



The image shows two separate windows titled "Prompt dei comandi".

Left Window (Ping Results):

```

Esecuzione di Ping 130.136.2.241 con 32 byte di dati:
Risposta da 130.136.2.241: byte=32 durata<1ms TTL=128

Statistiche Ping per 130.136.2.241:
  Pacchetti: Trasmessi = 4, Ricevuti = 4, Persi = 0 (0% persi),
  Tempo approssimativo percorsi andata/ritorno in millisecondi:
    Minimo = 0ms, Massimo = 0ms, Medio = 0ms

C:\>ping 130.136.10.241

Esecuzione di Ping 130.136.10.241 con 32 byte di dati:
Richiesta scaduta.
Richiesta scaduta.
Richiesta scaduta.
Richiesta scaduta.

Statistiche Ping per 130.136.10.241:
  Pacchetti: Trasmessi = 4, Ricevuti = 0, Persi = 4 (100% persi),

```

Right Window (Traceroute Results):

```

C:\>tracert 193.206.134.21
Rilevazione instradamento verso mi-bo-g.garr.net [193.206.134.21]
su un massimo di 30 punti di passaggio:

  1  <1 ms   <1 ms   <1 ms  csgw-3-0-5.cs.unibo.it [130.136.2.254]
  2  <1 ms   <1 ms   <1 ms  cesia-csgw.cs.unibo.it [130.136.254.254]
  3  <1 ms   <1 ms   <1 ms  alga11.unibo.it [137.204.2.17]
  4  <1 ms   <1 ms   <1 ms  rtg-unibo.bo.garr.net [193.206.128.125]
  5  4 ms    4 ms    4 ms  mi-bo-g.garr.net [193.206.134.21]

Rilevazione completata.

C:\>

```

Reti di Calcolatori - introduzione

Alcuni esempi di esecuzione: applicazione PING e traceroute (tracert).

Sono mostrate due finestre di esecuzione delle applicazioni PING e traceroute (tracert). La figura di sinistra mostra l'applicazione PING in esecuzione verso un host di indirizzo IP 130.136.2.241. Il testo mostrato è il seguente: "Esecuzione di Ping 130.136.2.241 con 32 Byte di dati:", seguito da 4 invii di richieste ping ognuna delle quali raggiunge con successo l'host destinazione. I messaggi mostrati infatti dicono: "Risposta da 130.136.2.241: byte=32 durata < 1ms TTL=128". Al termine delle 4 richieste viene mostrato il riepilogo delle statistiche: "Numero pacchetti trasmessi = 4, pacchetti ricevuti = 4, persi = 0 (0%persi). Tempo approssimativo percorsi andata/ritorno in millisecondi: minimo 0ms, massimo 0ms medio 0ms". di seguito, sulla stessa finestra, viene mostrato un esempio di ping alla macchina 130.136.10.241 (non collegata alla rete). Il testo questa volta dice: : "Esecuzione di Ping 130.136.10.241 con 32 Byte di dati:", seguito da 4 invii di richieste ping ognuna delle quali non raggiunge con successo l'host destinazione. I messaggi mostrati infatti dicono: "Richiesta scaduta". Al termine delle 4 richieste scadute viene mostrato il riepilogo delle statistiche: "Numero pacchetti trasmessi = 4, pacchetti ricevuti = 0, persi = 4 (100%persi). La figura di destra mostra il tracciato di indirizzi IP dei router attraversati nel cammino dall'host 130.136.2.241 all'host 193.206.134.21. Il testo dice: "C:\>tracert 193.206.134.21" che è il comando inviato, seguito dalla risposta dell'applicazione tracert: "Rilevazione instradamento verso mi-bo-q.garr.net [193.206.134.21] su un massimo di 30 punti di passaggio:", seguito da 5 righe simili a: "1 <1ms <1ms <1ms csgw-3-0-5.cs.unibo.it [130.136.2.254]" ognuna relativa a un router diverso, fino ad arrivare all'host finale: "1 <1ms <1ms mi-bo-q.garr.net [193.206.134.21]", seguito da "Rilevazione completata".

Reti di Calcolatori - introduzione

Esistono due applicazioni di servizio che sono basate sui messaggi ICMP. Tali applicazioni sono molto utili per la verifica delle cause o del semplice sospetto di problemi di rete. Le applicazioni sono l'applicazione PING e l'applicazione Traceroute.

L'applicazione PING permette di testare la connessione tra due host: eseguendo il comando “ping <indirizzo IP di host2>” da un host1 qualsiasi (mittente) connesso in rete, l'applicazione invia una richiesta ICMP di eco, alla quale l'host2 indicato risponde con una risposta ICMP (eco della richiesta). Dopo l'invio della richiesta, host1 fa partire un timer. In caso di successo, viene calcolato il tempo di andata e ritorno dei pacchetti (durata o Round Trip Time, RTT), mentre in caso di insuccesso viene indicato che il timer per la richiesta inviata è scaduto senza ottenere risposta (secondo esempio della prima figura). Al termine dei tentativi, viene mostrato un elenco di statistiche sul numero di richieste andate a buon fine e i tempi medi stimati di andata e ritorno dei pacchetti.

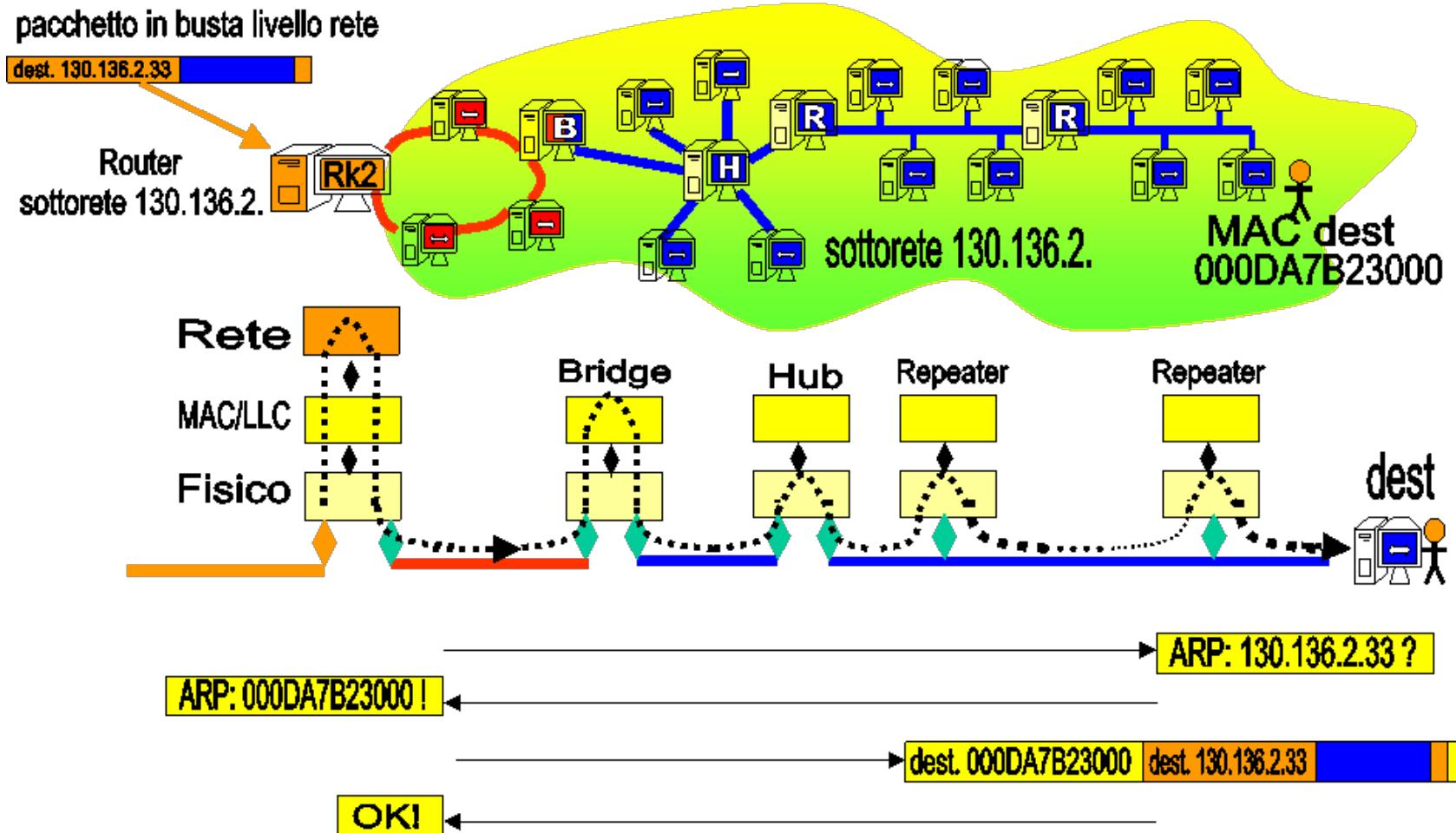
L'applicazione Traceroute (tracert) permette di verificare la lista di tutti i router attraversati da una richiesta ICMP inviata da host1 a host2. Il comando “tracert <indirizzo IP di host2>” eseguito da host1, causa l'invio di una sequenza di richieste ICMP verso host2, per le quali viene fissato il numero massimo di router da attraversare (numero di passaggi o tempo di vita, TTL), a valori crescenti da 1 in poi. Ogni router a distanza TTL risponde con un messaggio ICMP di errore (tempo di vita scaduto) attraverso il quale è possibile risalire al suo indirizzo IP (ognuno mostrato su righe successive). Un esempio è mostrato nella seconda figura.

protocollo ARP e RARP

Quando un router riceve un pacchetto destinato a un indirizzo IP della propria sottorete, esso deve produrre un frame di livello MAC/LLC che riporti specificato l'indirizzo MAC del destinatario (e non l'indirizzo IP), per poterlo passare al livello MAC/LLC per la trasmissione sulla rete locale.

- Protocollo Address Resolution Protocol (ARP)
 - Usato se il router non conosce l'indirizzo MAC corrispondente all'indirizzo IP
 - Il router genera un frame spedito a tutti i dispositivi della rete locale dove si chiede: “quale è l'indirizzo MAC del dispositivo che ha questo indirizzo IP”?
 - Se tale dispositivo esiste, esso risponde con un frame di livello MAC indirizzato al router, nel quale viene evidenziato l'indirizzo MAC richiesto
- Protocollo Reverse-ARP (RARP)
 - È la versione opposta del protocollo ARP, ma funziona allo stesso modo
 - La domanda è “quale indirizzo IP corrisponde al dispositivo con questo indirizzo MAC”?

Reti di Calcolatori - introduzione



Un'illustrazione della consegna di un pacchetto IP al destinatario finale, usando il protocollo ARP.

La figura mostra la fase di consegna di un pacchetto IP da parte del router all'host destinatario appartenente alla sottorete. Si tratta dell'illustrazione del passo finale descritto nella diapositiva che illustra il forwarding a livello IP. Il router che riceve un pacchetto a livello IP destinato alla sua sottorete (130.136.2.33) spedisce sui segmenti della rete locale un frame in broadcast (cioè ricevuto da tutti i dispositivi) contenente il codice di richiesta ARP, e l'indirizzo IP del destinatario del pacchetto. Il destinatario in questione, se esiste, risponde con un frame indirizzato all'indirizzo MAC del router, con allegato l'indirizzo MAC richiesto. A questo punto il router può quindi preparare la busta di livello MAC/LLC,

Reti di Calcolatori - introduzione

indirizzata al MAC del dispositivo destinatario del pacchetto IP, e contenente il pacchetto IP incapsulato all'interno. Il destinatario riceve il frame e risponde con il frame di conferma per il sottolivello LLC.

Si è visto come i router si occupino di gestire l'inoltro dei pacchetti a livello rete, e si è visto come la gestione basata su indirizzamento IP permetta di nascondere i dettagli interni delle reti locali, al di fuori della rete locale stessa. Si è visto anche che all'interno delle reti locali, la trasmissione su un segmento di rete locale avviene mediante frame di livello MAC/LLC, indirizzati mediante gli indirizzi MAC dei dispositivi, e non attraverso gli indirizzi IP. Un router deve quindi saper gestire l'associazione tra indirizzo IP di un dispositivo a livello rete e il suo indirizzo MAC a livello MAC/LLC. Il protocollo Address Resolution Protocol (ARP) risponde in modo standard a questa esigenza. In altre parole il protocollo ARP è il protocollo che lega l'indirizzamento a livello MAC con l'indirizzamento a livello IP.

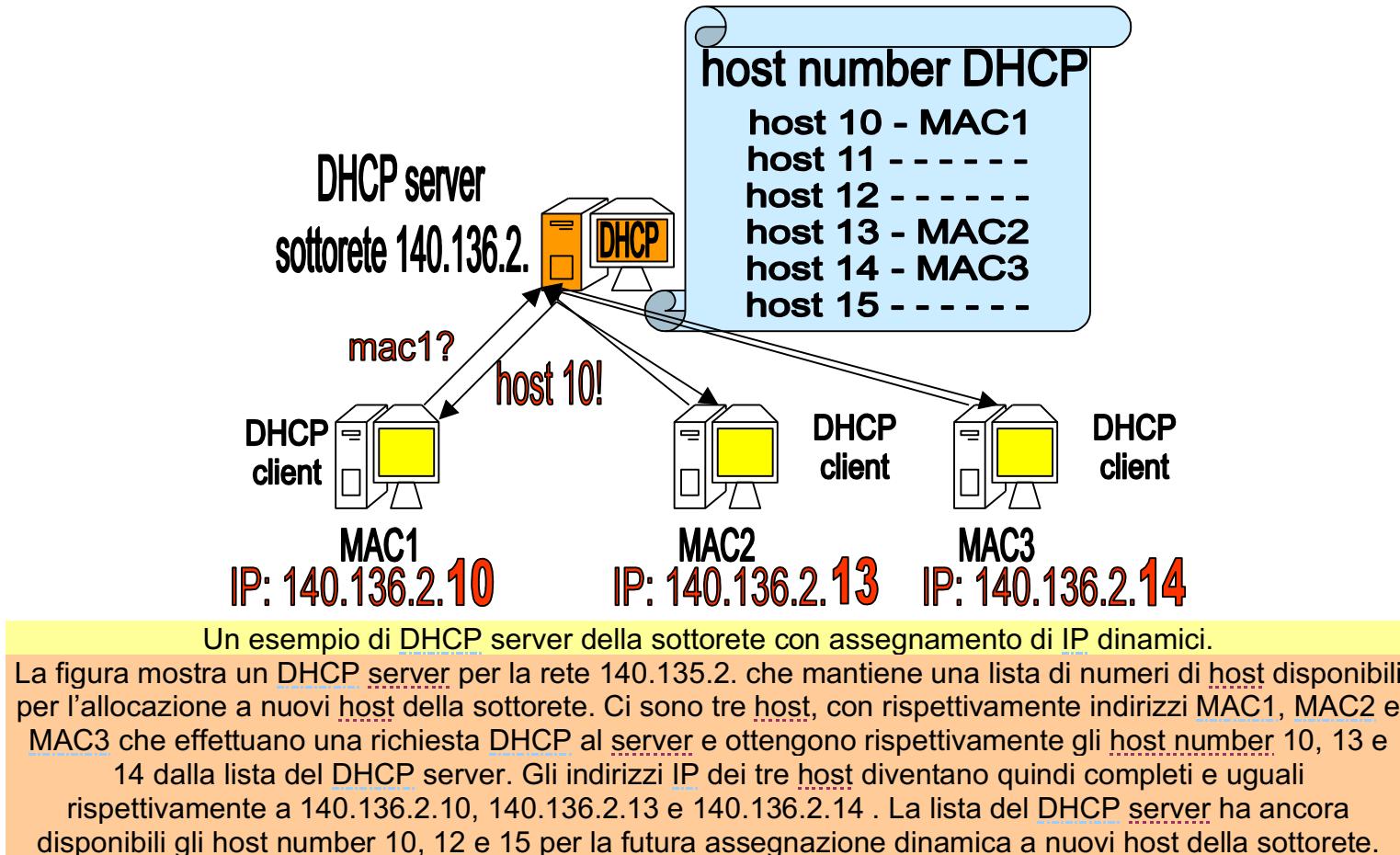
Quando un router riceve un pacchetto a livello IP destinato alla sua sottorete (es. 130.136.2.33) esso verifica se a tale IP risulti o meno associato un indirizzo MAC. In caso contrario, il router spedisce sui segmenti della rete locale un frame in broadcast (cioè ricevuto da tutti i dispositivi) contenente il codice di richiesta ARP, e l'indirizzo IP del destinatario del pacchetto. Tale frame equivale quindi al rivolgere a tutti i dispositivi la domanda: "quale indirizzo MAC ha il dispositivo corrispondente al seguente indirizzo IP"? Il dispositivo in questione, se esiste, risponde con un frame indirizzato all'indirizzo MAC del router, contenente il codice di risposta ARP, e con allegato l'indirizzo MAC richiesto.

A questo punto il router può quindi preparare e spedire la busta di livello MAC/LLC, indirizzata al MAC del dispositivo destinatario del pacchetto IP, contenente il pacchetto IP incapsulato all'interno. Il destinatario riceve il frame e risponde con il frame di conferma per il sottolivello LLC.

Esiste anche una versione analoga del protocollo ARP, detta Reverse-ARP, che risponde alla domanda: "quale indirizzo IP corrisponde al dispositivo con questo indirizzo MAC"?

Assegnazione indirizzi IP: DHCP

- Assegnazione dei **numeri di rete** di classe A, B, C
 - vengono assegnati a enti, aziende ed organizzazioni che ne fanno richiesta, da parte di enti internazionali (RIPE, ICANN, ARIN, APNIC).
- Assegnazione dei **numeri di host** ai dispositivi di una rete
 - Assegnazione manuale da parte dell'amministratore di rete
 - associa manualmente indirizzi IP e indirizzi MAC (IP statico)
 - Assegnazione automatica da parte di un **server Dynamic Host Configuration Protocol (DHCP)**: indirizzi IP dinamici
 - Metodo automatico usato in reti **wireless**, reti locali e connessioni domestiche
 - **Server DHCP**: è un **host** che implementa il servizio di assegnazione dell'indirizzo IP agli **host** che ne fanno richiesta
 - Un **DHCP** server dispone di un blocco di **host number** liberi per la sua rete
 - **DHCP** server associa indirizzi IP a indirizzi MAC dei dispositivi che lo richiedono
 - I dispositivi devono scegliere di affidarsi al **server DHCP** (“ottieni automaticamente indirizzo IP”)



Reti di Calcolatori - introduzione

I numeri di rete delle classi A, B e C, ovvero la parte sinistra degli indirizzi IP vengono assegnati da enti internazionali quali RIPE, ICANN, ARIN, APNIC a enti, aziende, consorzi e imprese che ne fanno richiesta motivata.

Un problema molto più pratico riguarda il modo in cui un nuovo dispositivo che venga connesso a una rete esistente, veda associare al proprio indirizzo MAC un indirizzo IP della rete stessa. Il numero di rete o di sottorete viene automaticamente determinato dall'appartenenza alla rete, ovvero alla presenza al di sotto del dominio di gestione di un router.

La prima, ovvia alternativa (molto usata) è quella di avere un amministratore di rete che assegna manualmente uno dei numeri di host disponibili al nuovo indirizzo MAC. In questo modo l'associazione indirizzo MAC e indirizzo IP può essere mantenuta per un tempo indeterminato, e quindi si considera l'indirizzo IP come statico.

La seconda alternativa, molto usata in reti senza fili, in reti locali e nei collegamenti domestici a Internet Service Provider (ISP) via Modem o ADSL consiste nell'utilizzare un server per Dynamic Host Configuration Protocol (DHCP).

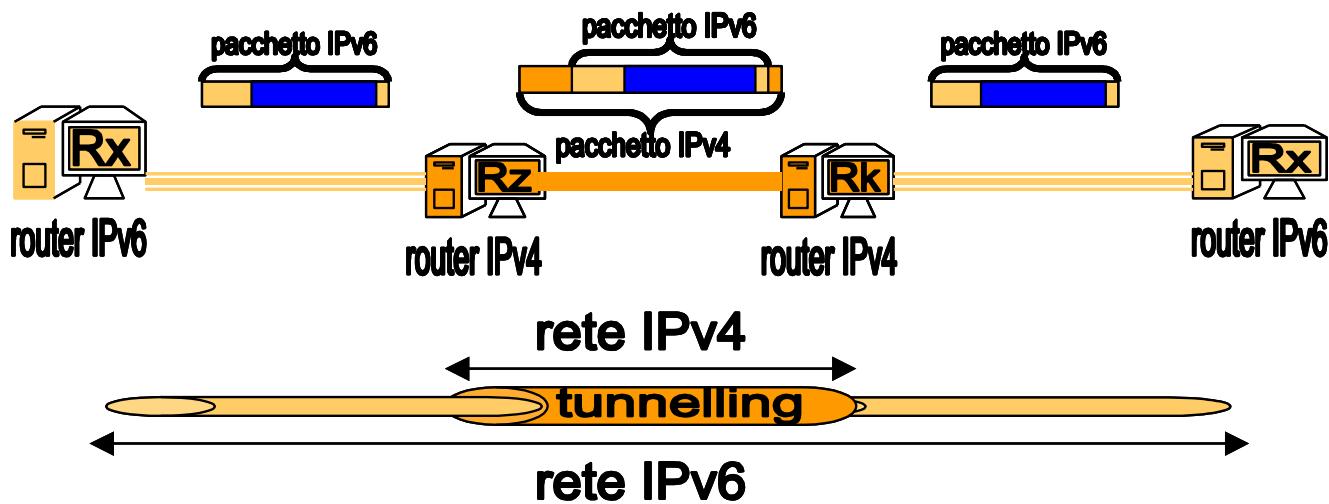
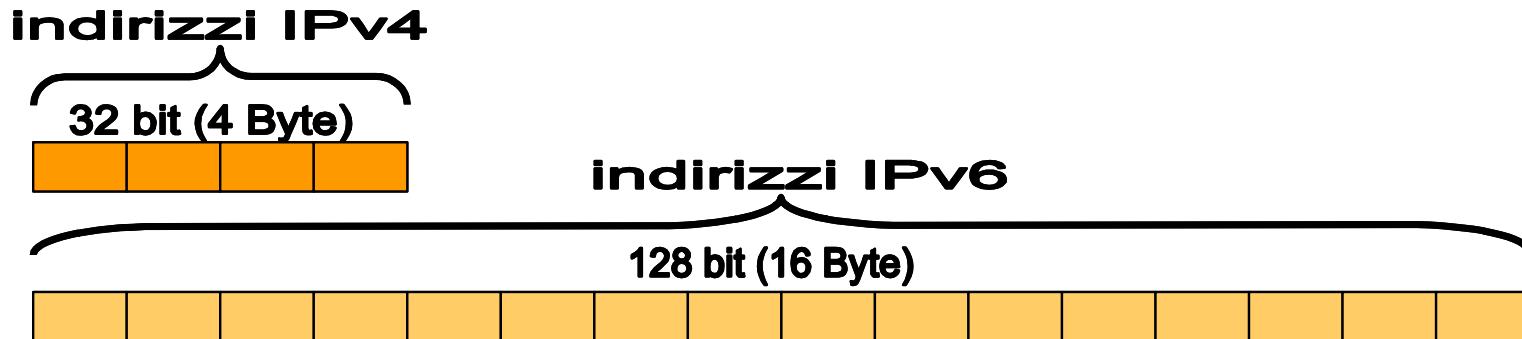
Il server DHCP è dotato di una lista di numeri di host liberi per la sottorete amministrata, che provvede ad associare su richiesta agli indirizzi MAC dei dispositivi che lo richiedono. Tale associazione dipende spesso dalla disponibilità degli indirizzi già assegnati in precedenza, quindi allo stesso indirizzo MAC possono essere associati di volta in volta indirizzi IP diversi, e si parla in questo caso di indirizzi IP dinamici.

E' possibile configurare attraverso DHCP anche altri parametri di rete, come la maschera di rete, il default router e il server DNS (che vedremo dopo).

Il servizio DHCP equivale spesso al concetto di rete "plug and play", ovvero rete in cui basta connettere il dispositivo al medium e non c'è bisogno di nessuna configurazione manuale aggiuntiva.

IPv6 e tunnelling IPv4

- Dal 1990 è attiva la definizione e l'implementazione di una nuova versione del protocollo di indirizzamento IP: **IPv6**
- Motivazioni per la definizione di IPv6: Indirizzi IPv4 finiranno negli anni 2008-2018.
- Caratteristiche salienti di IPv6
 - **Indirizzi IPv6:** estesi a 128 bit (16 Byte) anziché i 32 bit (4 Byte) di IPv4
 - circa 15000 indirizzi IPv6 per ogni metro quadrato di tutta la superficie terrestre!
 - Nuova struttura dei campi della busta dei pacchetti di livello rete (IP)
 - Identificazione di parametri per differenziare flussi di dati a priorità diverse
- Integrazione di IPv4 o sostituzione di IPv4?
 - Per ora la sperimentazione IPv6 avviene su reti separate (nuovi **router IPv6**)
 - Ci sono casi di integrazione tra IPv4 e IPv6 usando tecniche di **tunnelling**
 - **Tunnelling:** spedire pacchetti IPv6 in buste IPv4.



Un confronto in scala tra indirizzi IPv4 e IPv6, e un esempio di tunnelling IPv6 in IPv4.
 La figura mostra in alto una scala dimensionale degli indirizzi IPv4 di 32 bit rispetto agli indirizzi IPv6 di 128 bit (4 volte più grandi). In basso viene mostrato come sia possibile spedire pacchetti IPv6 tra due router IPv6, passando per cammini che includono router IPv4, attraverso il tunnelling IPv6 in IPv4. Il pacchetto IPv6 viene incapsulato dai router IPv4 in pacchetti IPv4, in modo da poter essere instradato lungo la rete di router IPv4. Uscito dal tunnel IPv4 il pacchetto IPv6 prosegue il suo inoltro fino alla destinazione IPv6 finale.

Reti di Calcolatori - introduzione

Dal 1990 è stato avviato un progetto di definizione e sviluppo di una nuova versione del protocollo IPv4, denominato versione IPv6. In seguito all'esplosione del collegamento di calcolatori in rete, e quindi dell'utilizzo di indirizzi e reti IP, le proiezioni mostrano che al ritmo attuale gli indirizzi IPv4 saranno esauriti nel decennio 2008-2018.

Brevemente, le caratteristiche salienti di IPv6 vanno nella direzione di ovviare a questo problema, oltre a migliorare alcuni aspetti di IPv4.

La caratteristica fondamentale di IPv6 è la definizione di nuovi indirizzi IPv6 composti da 128 bit (16 byte), cioè ben quattro volte la dimensione degli indirizzi IPv4. Questo incredibile numero di indirizzi potrebbe consentire di avere circa 15000 indirizzi IPv6 per dispositivi diversi su ogni metro quadrato di superficie dell'intero pianeta, oceani inclusi.

Sono inoltre stati ridefiniti i campi che costituiscono la busta dei pacchetti di livello IPv4, aggiungendo ad esempio parametri per la gestione di flussi di pacchetti IP con diversi livelli di priorità.

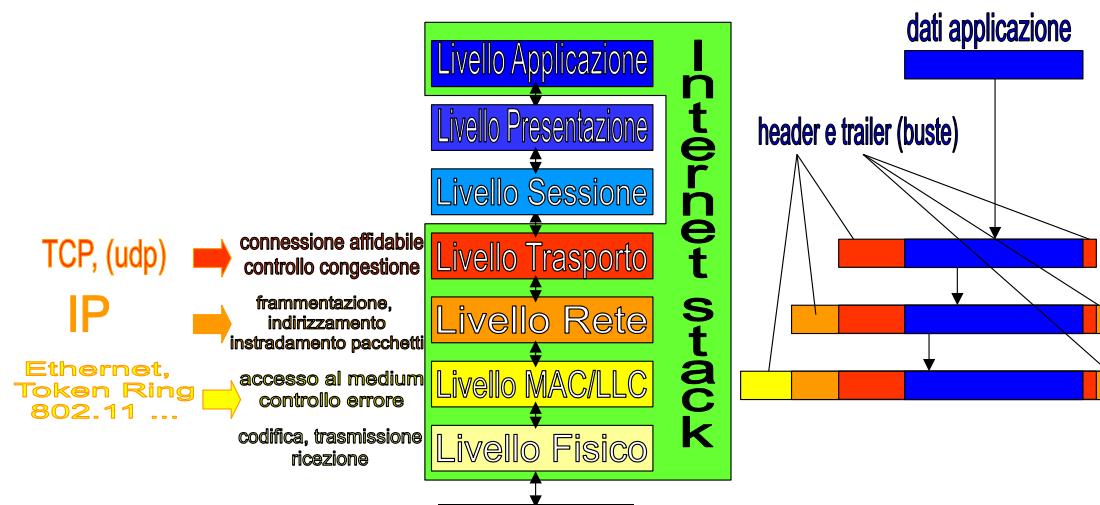
Purtroppo la definizione di IPv6 nella maggioranza dei casi non permette di continuare a usare i vecchi router IPv4, e quindi non è compatibile con l'attuale struttura di Internet. La sperimentazione e lo sviluppo di IPv6 sta procedendo su reti IPv6 separate, che possono in certi casi integrarsi alle reti IPv4 usando la tecnica del tunnelling dei pacchetti IPv6 in IPv4.

Nella figura viene mostrato come sia possibile spedire pacchetti IPv6 tra due router IPv6 passando per cammini che includono router IPv4, attraverso il tunnelling IPv6 in IPv4. Il pacchetto IPv6 viene incapsulato dai router IPv4 in pacchetti IPv4, in modo da poter essere instradato lungo la rete di router IPv4. Uscito dal tunnel IPv4 il pacchetto IPv6 prosegue il suo inoltro fino alla destinazione IPv6 finale.

Livello trasporto

I protocolli di Internet di quarto livello (trasporto) sono essenzialmente il **Transmission Control Protocol (TCP)** e lo **User Data Protocol (UDP)**.

- Livello Trasporto: protocolli TCP e UDP
 - Servizio **trasporto affidabile** (protocollo TCP): servizio di tipo connection-oriented
 - configurazione del **numero di porta** (port number) del **socket TCP**
 - numerazione sequenziale dei pacchetti, riordino, eliminazione duplicati
 - Pacchetti di conferma della ricezione (acknowledgment di livello trasporto)
 - Pacchetti non ricevuti (non confermati entro timeout) spediti di nuovo
 - gestione della congestione e controllo di flusso dei pacchetti.
 - Meccanismi a finestra scorrevole (sliding window)
 - Servizio **trasporto non affidabile** (protocollo UDP): servizio connectionless



Reti di Calcolatori - introduzione

La collocazione del livello Trasporto, i servizi in esso realizzati e i protocolli TCP e UDP.

La figura mostra l'architettura a livelli dei protocolli di Internet visti finora. Al secondo livello troviamo i protocolli MAC, tra i quali Ethernet, Token ring, 802.11, e protocolli LLC, tra i quali HDLC e PPP. I servizi indicati al livello MAC/LLC sono essenzialmente l'accesso al mezzo trasmissivo e il controllo degli errori di trasmissione. Al terzo livello (rete) troviamo il protocollo IP. Sono indicati i servizi supportati dal protocollo IP a livello rete: frammentazione, indirizzamento e instradamento dei pacchetti dati. Al quarto livello troviamo i protocolli TCP e UDP. TCP in particolare realizza i servizi di connessione orientata alla connessione e controllo della congestione.

Reti di Calcolatori - introduzione

Il quarto livello dei protocolli dell'architettura di Internet è il livello trasporto (*transport*), ed è basato su due protocolli in particolare: il **Transmission Control Protocol (TCP)** e lo **User Data Protocol (UDP)**, che possono essere usati in alternativa tra loro.

Il servizio di trasporto dei pacchetti ottenuto mediante il protocollo TCP è quello di tipo **connection-oriented**: malgrado la rete a livello IP sia non affidabile (cambia l'ordine, duplica, o perde i pacchetti dati spediti), il protocollo TCP si occupa di garantire il ripristino dell'ordinamento dei pacchetti e la ri-trasmissione dei pacchetti perduti. TCP numera sequenzialmente tutti i dati spediti, e in questo modo permette alla sua controparte di procedere al riordino dei pacchetti che dovessero giungere disordinati, ad esempio (pacchetto 10), (pacchetto 9). Inoltre, attraverso il numero d'ordine, la controparte può inviare a sua volta pacchetti di conferma della ricezione (**acknowledgment**, simili, ma da non confondere con gli **acknowledgment** di livello LLC), es. (conferma 10, conferma 9). Grazie al meccanismo di conferma della ricezione dei pacchetti numerati, il mittente può accorgersi di pacchetti non ricevuti e inviarli nuovamente, anche più volte, finché non riceve la conferma attesa.

TCP si basa sul principio di connessione punto a punto tra punti virtuali detti **socket**, che permettono di smistare i pacchetti verso le rispettive applicazioni di livello superiore. Questo punto sarà evidenziato nella diapositiva successiva.

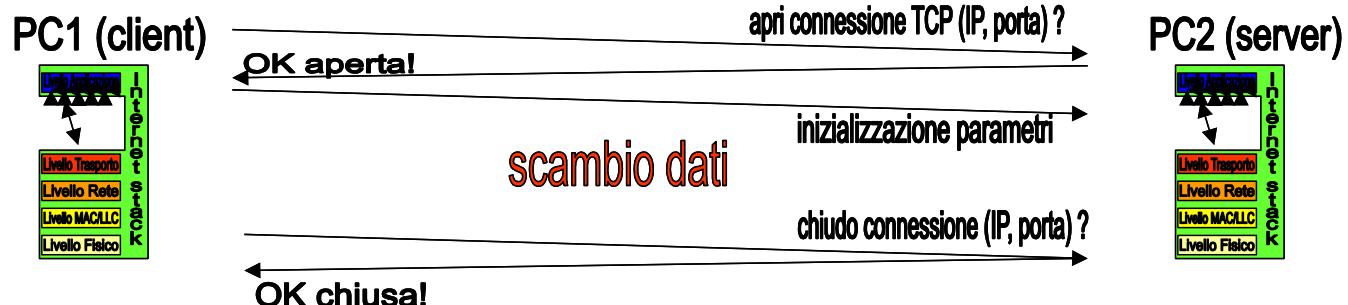
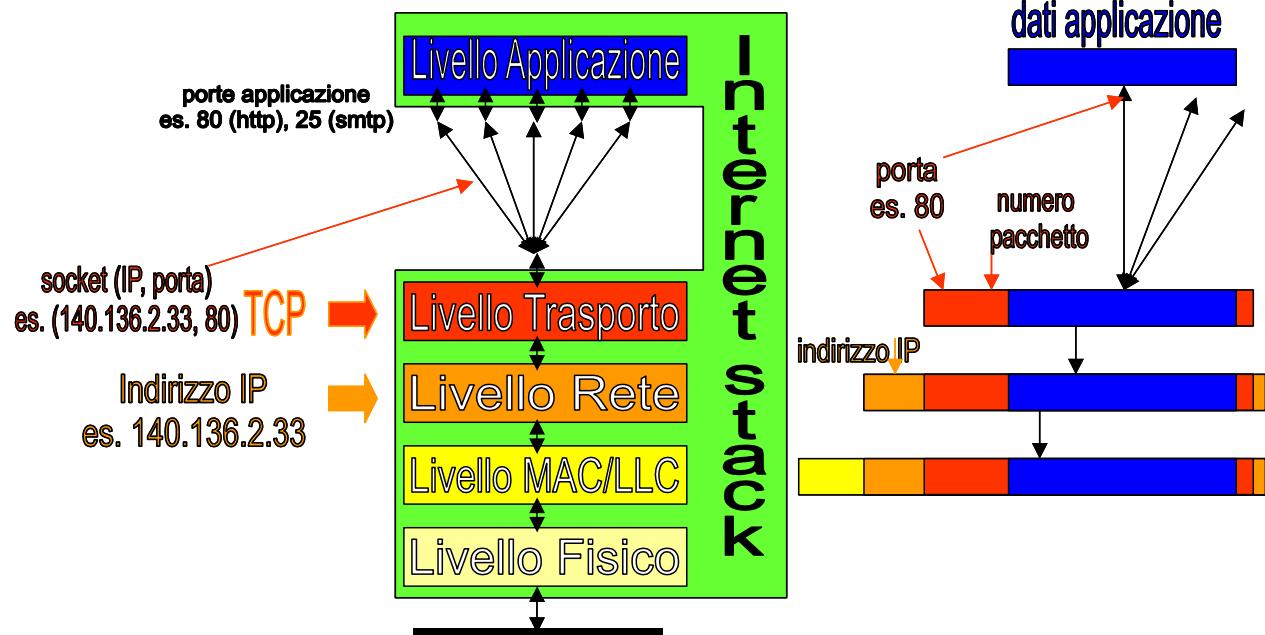
TCP permette infine di controllare la velocità di invio dei flussi dei pacchetti, mediante il meccanismo a finestra scorrevole, in modo da evitare la congestione della rete. Anche questo punto sarà ripreso in una diapositiva successiva.

In alternativa, il protocollo UDP non fa nulla di tutto questo, a parte lo smistamento verso le applicazioni di livello superiore. L'unico aspetto positivo di UDP è la sua estrema semplicità e il fatto che non aggiunge pesanti gestioni all'invio di pacchetti. Il servizio ottenuto, tuttavia, rimane un servizio di tipo **connectionless**, simile a quello fornito dal livello rete con IP.

Livello trasporto su Internet: TCP

- TCP è il protocollo di livello trasporto usato di fatto su Internet
- lo standard architetturale “de facto” usato su Internet prevede il **connubio TCP/IP**
- TCP consente lo smistamento dei pacchetti verso le rispettive applicazioni in ascolto su “porte”
- TCP richiede **attivazione della connessione** punto a punto tra due socket.
 - **Socket**: (indirizzo IP + numero di porta dell’applicazione di livello superiore)
 - In questo modo mette in comunicazione le applicazioni in attesa sui socket.
 - Es. PC1 (client) invia richiesta TCP di connessione sul socket del PC2 (server)
 - Se il socket esiste e non è occupato, TCP di PC2 risponde ok!
 - Se riceve l’ok da PC2, TCP di PC1 può inviare i dati di configurazione
 - Ora avviene lo scambio dati veri e propri a livello TCP
- **Rilascio della connessione** TCP: si liberano le porte usate

Reti di Calcolatori - introduzione



Il concetto di socket TCP, l'incapsulamento TCP, e le fasi di apertura connessione, scambio dati e chiusura connessione.

La figura mostra i livelli ISO/OSI di Internet tra i quali spicca al livello quarto, il livello trasporto e il suo protocollo TCP. Il livello trasporto mostra al livello applicazione un servizio di spedizione affidabile, mentre il livello applicazione mostra al livello trasporto una porta per ognuna delle applicazioni attive. Il socket viene rappresentato come una linea dal livello trasporto (associato all'indirizzo IP sottostante) a una delle porte delle applicazioni del livello applicazione. A destra dei livelli descritti viene schematizzato l'incapsulamento

Reti di Calcolatori - introduzione

dei dati dell'applicazione. Al livello trasporto, i dati sono incapsulati ponendo in testa al pacchetto il numero di porta e il valore del contatore di pacchetti di TCP. Il pacchetto TCP viene passato al livello inferiore IP che aggiunge l'indirizzo IP del destinatario. Sotto a queste figure viene schematizzato lo scambio di messaggi per l'apertura e la chiusura della connessione TCP. Il client invia la richiesta di apertura del socket, il server risponde con la conferma dell'apertura e il client spedisce i dati di configurazione. In seguito avviene lo scambio dati a livello TCP. Al termine il client invia la richiesta di chiusura della connessione e il server risponde confermando la chiusura.

Il protocollo TCP richiede a due dispositivi che intendano comunicare di effettuare preventivamente la configurazione dei parametri del socket TCP, originando in questo modo un canale virtuale di tipo punto a punto tra due socket, ovvero tra due applicazioni di livello superiore alle quali vengono smistati i pacchetti da TCP. Un socket è un punto di arrivo o partenza (virtuale) dei dati a livello trasporto, dal quale è in atto l'invio e la ricezione di pacchetti destinati a un'applicazione, ed equivale a una coppia: (indirizzo IP, numero di porta dell'applicazione). Una volta instaurata la configurazione punto a punto tra due socket, attraverso lo scambio di pacchetti di configurazione, può iniziare lo scambio dei dati a livello trasporto.

La connessione viene instaurata con una richiesta di uno dei due host (il client) nei confronti dell'host server. Deve essere specificato l'indirizzo IP del server e il numero di porta sul quale è in attesa l'applicazione (o il servizio) con la quale si intende dialogare. Il server verifica che il socket esista (verifica il numero di porta) e che non sia già occupato, e in caso affermativo risponde con un pacchetto di conferma. Se il pacchetto di conferma è ricevuto, il client invia un ulteriore pacchetto di conferma contenente i dati di configurazione, dopodiché la comunicazione procede attraverso lo scambio di pacchetti dati tra i livelli TCP, che verranno smistati verso le porte indicate.

Lo scambio dati avviene attraverso pacchetti che saranno ordinati e ritrasmessi in caso di perdita a cura del protocollo TCP.

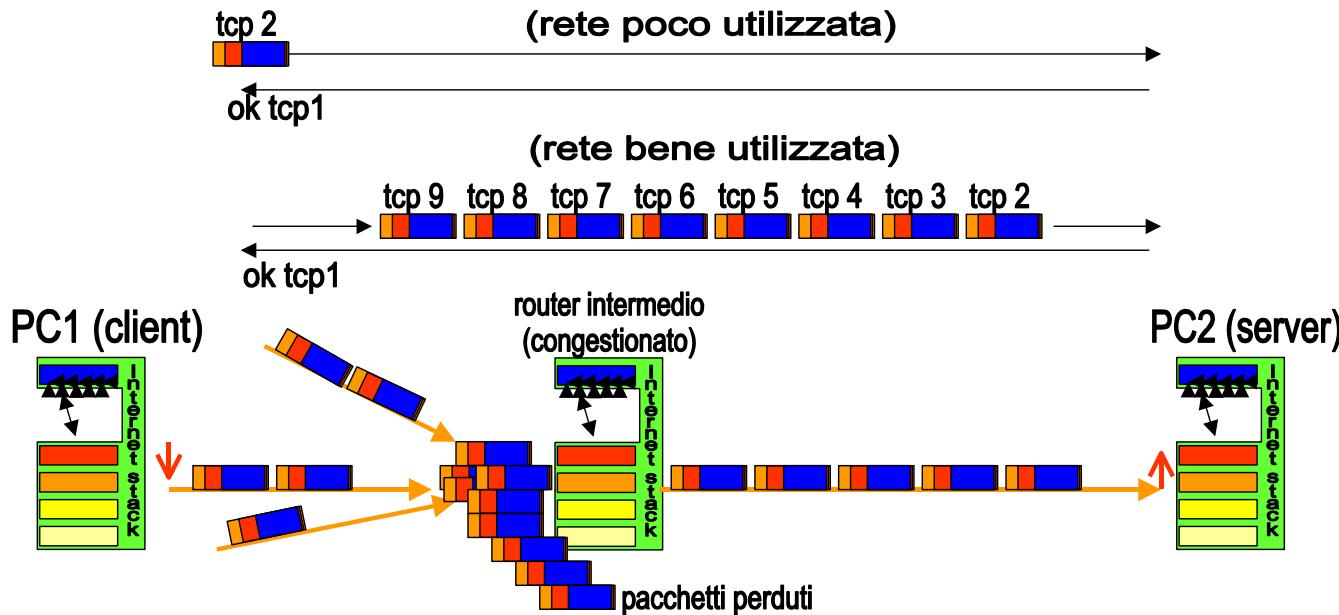
Al termine del dialogo, la connessione TCP tra due applicazioni può essere rilasciata, liberando la porta occupata.

Controllo di flusso e congestione di rete

TCP funziona tra due dispositivi di una rete, anche molto distanti tra loro, ed implementa tecniche di controllo di flusso e controllo della congestione di rete.

- Problema: la latenza di rete (tempo di andata e ritorno dei pacchetti) è alta
 - Invio pacchetto e attendo conferma prima di inviare il successivo?
 - Solo un pacchetto in tutto il cammino: troppo lento!
 - Invio tutti i pacchetti uno di seguito all'altro in un colpo solo?
 - Possibile congestione nei router intermedi, che perdono pacchetti: troppo veloce!
 - Possibile saturazione del destinatario, che riceve pacchetti troppo velocemente
 - Scopo del **controllo di flusso**: inviare pacchetti al massimo ritmo sostenibile dal destinatario finale
 - Scopo del **controllo di congestione**: inviare pacchetti al massimo ritmo sostenibile dal router più lento della rete dal mittente al destinatario finale
- Esistono vari metodi: meccanismo a **finestra scorrevole (sliding window, SW)**.

Reti di Calcolatori - introduzione



Il problema del controllo di flusso e della congestione a livello trasporto.

La figura mostra due dispositivi, client e server, che scambiano un flusso di pacchetti a livello trasporto con il protocollo TCP. Tra il client e il server esiste un router impegnato dai pacchetti IP di molte connessioni TCP (da parte di altri client e server). Il router non è in grado di sostenere un ritmo troppo veloce per inoltrare i pacchetti, che si accatastano e possono finire perduti. Questo esempio rappresenta un caso di congestione di un router intermedio.

Reti di Calcolatori - introduzione

Come si è detto in precedenza, TCP richiede una conferma per ogni pacchetto inviato. La distanza tra due dispositivi che scambiano pacchetti a livello trasporto può essere molto significativa. Il tempo per inviare un pacchetto e ottenere la conferma dell'avvenuta ricezione può quindi diventare dell'ordine dei secondi.

Il problema del controllo di flusso dei pacchetti nel protocollo TCP si basa su due scopi apparentemente in contraddizione tra loro. Il primo scopo è quello di saturare il più possibile la rete di pacchetti, inviandoli a un ritmo elevato. Questo favorisce l'utilizzo delle risorse e le prestazioni della rete (si spediscono e si ricevono tanti bit al secondo). Se si decidesse di inviare un pacchetto e aspettare l'arrivo della conferma, la rete sarebbe usata solo in minima percentuale, e si riuscirebbero a spedire solo pochi bit al secondo. Quindi la rete, pur essendo veloce nell'invio dei bit, verrebbe sfruttata al minimo delle potenzialità. E' quindi evidente quanto sia opportuno spedire i pacchetti a un ritmo il più veloce possibile.

D'altra parte, occorre evitare che un ritmo di invio troppo elevato possa causare il sorgere della congestione nei router intermedi del cammino dei pacchetti, dal mittente TCP (client) al destinatario TCP (server). Se un router si trova a dover inoltrare troppi pacchetti, provenienti da flussi TCP diversi, i pacchetti si accumulano fino ad andare perduti e la rete va in crisi. In tal caso si deve ricorrere a una tecnica di controllo della congestione.

Una forma di congestione può comparire anche sul destinatario finale, nel caso in cui esso non sia in grado di ricevere i pacchetti inviati troppo velocemente. In tal caso si deve ricorrere a una tecnica di controllo di flusso.

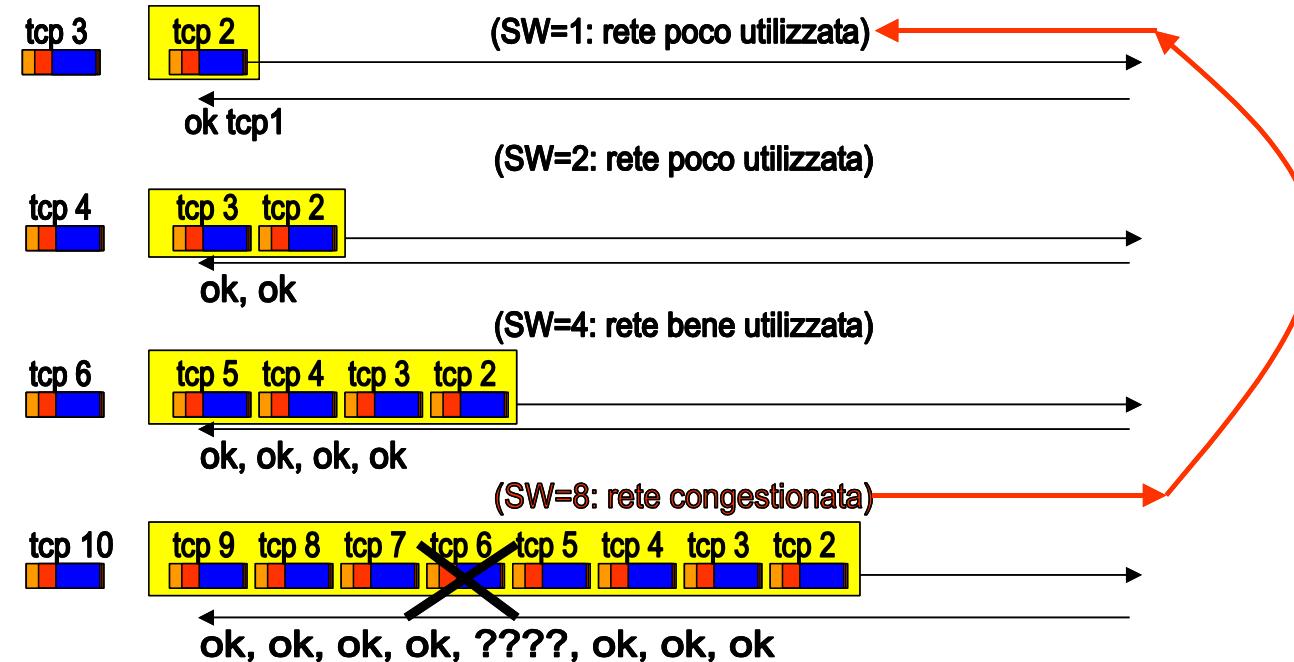
TCP usa un meccanismo per il controllo di flusso, detto a finestra scorrevole (*sliding window*), e un meccanismo per il controllo della congestione, basato sul dimensionamento della finestra scorrevole. Tutto ciò per cercare il massimo ritmo di spedizione che possa garantire l'inoltro dei pacchetti da parte del router più lento del cammino, e prevenire la saturazione del destinatario finale.

finestra scorrevole di TCP

Una **finestra scorrevole (sliding window, SW)** è un valore intero, es. 1 e rappresenta il numero massimo di pacchetti che un mittente può spedire di seguito, in attesa di ricevere la loro conferma.

- Ogni mittente TCP spedisce al massimo ritmo possibile un gruppo di SW pacchetti.
- **Controllo di flusso:** non spedisce più di SW pacchetti oltre l'ultimo non confermato
 - Se i primi SW pacchetti vengono confermati, spedisce i successivi SW pacchetti
 - Se un pacchetto non viene confermato lo rispedisce, prima di spedire i successivi SW
- **Controllo della congestione:** spedisce un numero SW variabile di pacchetti
 - Se SW pacchetti spediti sono tutti confermati, aumenta SW
 - Es. spedisce al massimo ritmo prima SW=2, poi 4, poi 8... pacchetti.
 - Appena un pacchetto degli SW inviati non viene confermato (scade il timeout)
 - Assume che ciò sia dovuto a congestione su un router
 - Riparte da SW minimo
 - Cerca di accelerare il ritmo gradualmente e, appena scopre la congestione, rallenta.

Reti di Calcolatori - introduzione



il meccanismo di controllo di flusso di TCP: finestra scorrevole (Sliding Window, SW)

La figura mostra una sequenza di pacchetti spediti su una rete a livello TCP, da sinistra a destra, usando il meccanismo di controllo di flusso della finestra scorrevole composto con un meccanismo di controllo della congestione basato su finestra di dimensione variabile. Il meccanismo parte con finestra uguale a uno, il che significa che solo un pacchetto può essere spedito prima di ricevere la conferma della ricezione. In questo caso la rete è poco utilizzata. Se la conferma è ricevuta, la finestra viene raddoppiata, spedendo due pacchetti al massimo ritmo di invio. Se entrambi i pacchetti vengono confermati, si passa alla finestra di dimensione quattro, inviando quattro pacchetti al massimo ritmo di invio. Se i pacchetti sono confermati si passa a finestra di otto pacchetti. A questo punto, almeno uno degli otto pacchetti non viene confermato.

Si suppone che questo fatto sia dovuto a un router congestionato e quindi si rallenta il ritmo di invio ripartendo dalla finestra minima (pari a uno). Il massimo grado sostenibile di invio per la rete in esame è stato quindi ottenuto con finestra pari a quattro.

Reti di Calcolatori - introduzione

La finestra scorrevole è un valore intero, che parte da un valore minimo (ad esempio il valore uno). L'idea alla base del controllo di flusso a finestra scorrevole è quello di spedire non più di SW pacchetti consecutivi, a partire dall'ultimo pacchetto non confermato, e quindi attendere la ricezione di una conferma. Un valore di SW uguale a 1 significa che solo un pacchetto può essere spedito, poi occorre aspettare di ricevere la conferma della ricezione. In questo caso la rete è poco utilizzata. Ogni volta che alcuni pacchetti spediti sono confermati, allora è possibile spedire i pacchetti successivi mantenendosi entro il limite massimo di SW pacchetti dall'ultimo pacchetto non ancora confermato. Eventuali pacchetti non confermati sono rispediti fino al ricevimento della conferma. Il senso di questo meccanismo è quello di lasciare in sospeso non più di SW pacchetti, per evitare di saturare il mittente. Questo meccanismo, molto semplificato, realizza il controllo di flusso di TCP.

Se i pacchetti vengono confermati, si può adottare un meccanismo dinamico per accelerare gradualmente il ritmo di invio dei pacchetti, ovvero la dimensione della finestra SW, fino a che non si nota la perdita di almeno un pacchetto tra quelli inviati.

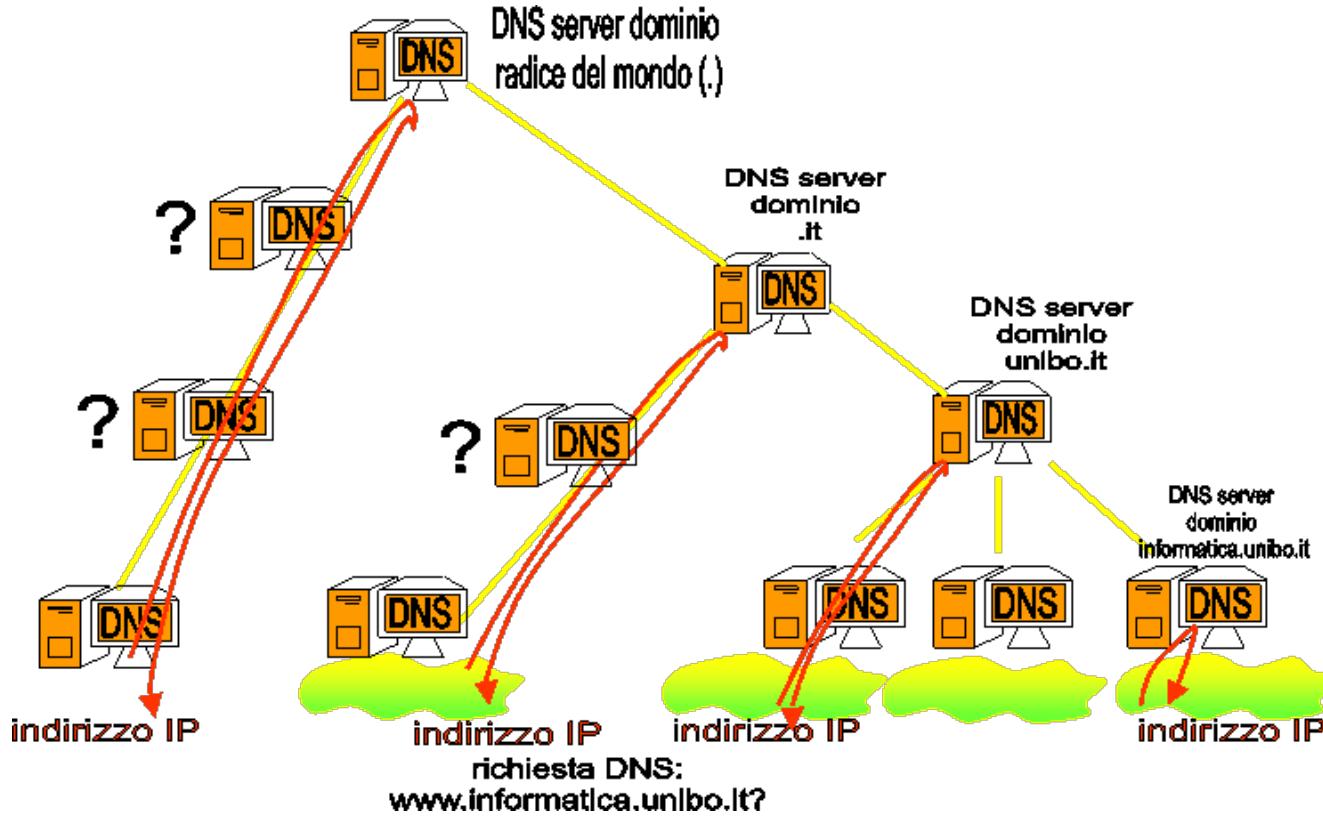
Se i pacchetti vanno perduti, TCP assume anche che la causa di ciò sia la presenza di un router intermedio congestionato, e quindi rallenta il ritmo di invio dei pacchetti per dare modo al router congestionato di smaltire i pacchetti accumulati. Tale meccanismo, sommariamente descritto, è il meccanismo di controllo della congestione di rete di TCP.

Osservando l'esempio, partendo con SW uguale a 1, se la conferma è ricevuta, la finestra viene raddoppiata, spedendo due pacchetti al massimo ritmo di invio. Se entrambi i pacchetti vengono confermati, si passa alla finestra di dimensione quattro, inviando quattro pacchetti al massimo ritmo di invio. Se i pacchetti sono confermati si passa a finestra di otto pacchetti. A questo punto, nell'esempio, almeno uno degli otto pacchetti non viene confermato. Si suppone che questo fatto sia dovuto a un router congestionato e quindi si rallenta il ritmo di invio ripartendo dalla finestra minima (pari a uno). Il massimo grado sostenibile di invio per la rete in esame nell'esempio è stato quindi ottenuto con finestra pari a quattro.

Nomi di Dominio e servizio DNS

- Gli utenti preferiscono usare **nomi per le risorse in rete**, anziché indirizzi IP
- Nomi di dominio per le reti: sono mnemonici e hanno una struttura gerarchica
- Anche i nomi di dominio sono assegnati in modo univoco come i numeri di rete
 - I nomi delle risorse sono arbitrari ma non duplicabili entro il dominio stesso.
- Problema: I protocolli di rete e i router pretendono di usare indirizzi IP
- Soluzione: Il servizio **Domain Name System (DNS)**
 - È basato su una catena di server DNS organizzati gerarchicamente
 - Ogni host in rete deve conoscere almeno un DNS server
 - Ogni server DNS conosce almeno un DNS server superiore
 - I server ricevono richieste (protocollo DNS) e forniscono indirizzi IP
 - Se un server non conosce la risposta inoltra la richiesta DNS a un server superiore
 - I server DNS radice (DNS root server) conoscono tutti i domini e i loro IP! (ma sono pochi e costosi)

Reti di Calcolatori - introduzione



un esempio di richieste DNS per associare indirizzi IP (sconosciuti) a risorse e nomi di dominio.

La figura mostra una possibile gerarchia di server DNS e un esempio di richieste DNS che risalgono la gerarchia dei server fino ad essere soddisfatte. Una richiesta DNS consiste nella richiesta dell'indirizzo IP associato a una risorsa, tipicamente a un host di un dominio. La richiesta della risorsa www.informatica.unibo.it, che equivale all'indirizzo IP del server web del dominio informatica.unibo.it può essere richiesta dal dominio stesso, e in tal caso viene risolta dal DNS locale. Da un altro dominio in Italia viene risolta da un server DNS del dominio .it, mentre dal resto del mondo viene risolta da un server di dominio radice del mondo.

Reti di Calcolatori - introduzione

Gli utenti di Internet preferiscono usare nomi mnemonici per identificare le risorse in rete, ad esempio nomi di host appartenenti a una certa rete, oppure indirizzi di e-mail di utenti di una certa rete. Anche le reti, risultano spesso facilmente identificabili attraverso i nomi di dominio della rete. I nomi di dominio hanno quindi lo stesso senso degli indirizzi IP, e infatti vengono assegnati da enti internazionali, come gli indirizzi IP, per evitare confusione e nomi duplicati. Le risorse appartenenti a un dominio possono avere nomi scelti arbitrariamente (ad esempio nomi di host, indirizzi di e-mail) purchè non siano duplicati all'interno del dominio stesso. Nomi di risorse duplicati sono ammessi in domini diversi, (ad esempio, pippo@topolinia.it e pippo@paperopoli.com). I nomi di dominio hanno una struttura gerarchica del tipo:

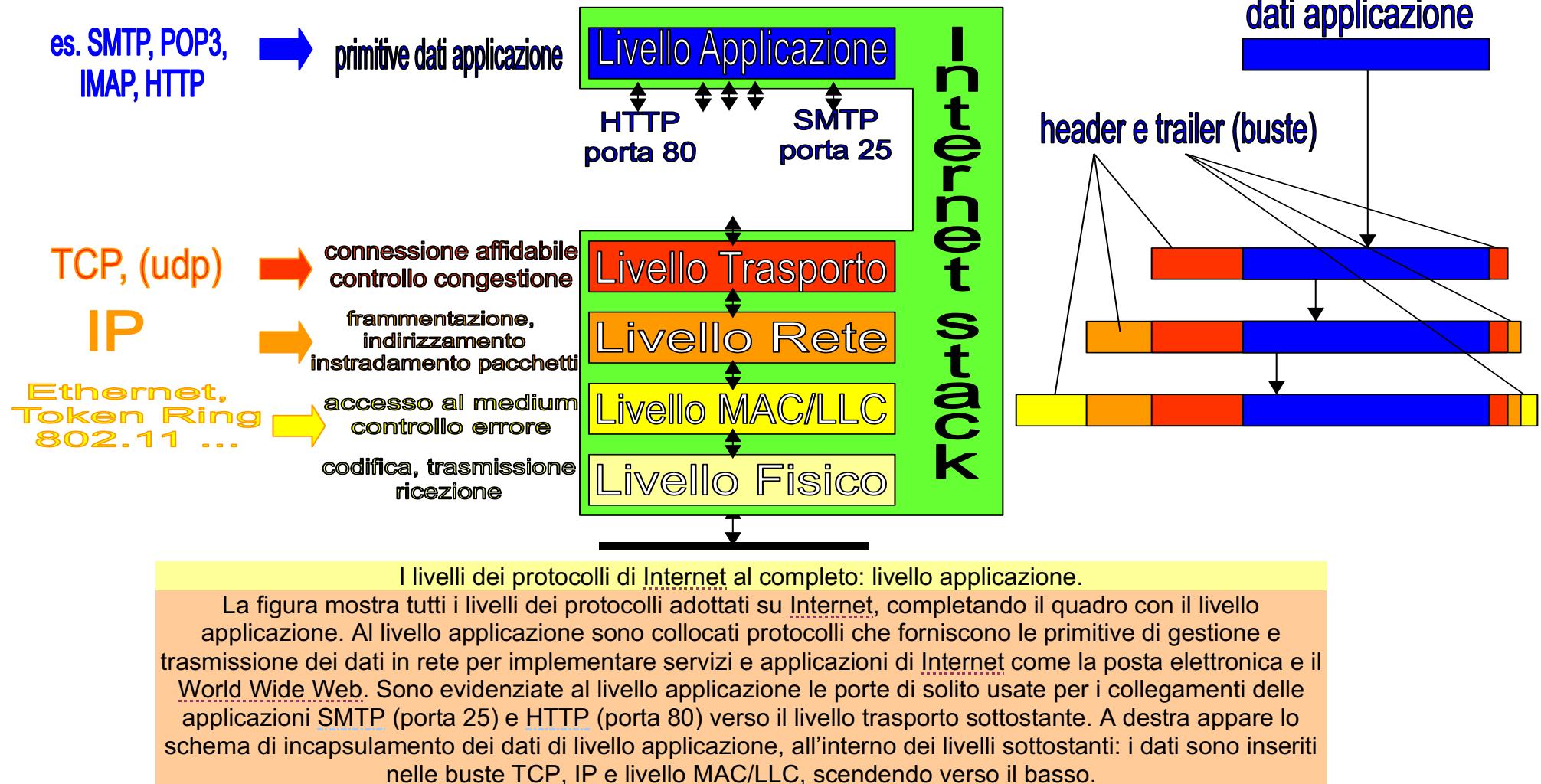
(nomerisorsa.sottodominio.sottodominio.dominioradice). Ad esempio www.informatica.unibo.it è il nome dell'host che agisce da web server per il sottodominio informatica, del sottodominio Università di Bologna, del sottodominio di livello massimo .it (Italia). In realtà il dominio radice del mondo, che esiste implicitamente, non si scrive mai.

Tutto ciò è comodo ma viola le esigenze del livello rete e dei router che pretendono solo indirizzi IP.

Per risolvere il problema, è nato il servizio Domain Name System (DNS) che attraverso una gerarchia di server e un protocollo standard per le richieste permette di risolvere l'associazione tra nome della risorsa e indirizzo IP. Ogni host in rete deve conoscere un server DNS al quale inviare le richieste e ogni server DNS deve conoscere almeno un server DNS di livello superiore. I server di livello superiore conoscono un numero sempre maggiore di nomi e relativi indirizzi IP, ma sono sempre meno per motivi di costo. L'esempio mostra come viene soddisfatta una richiesta DNS a seconda del punto della rete di server DNS dalla quale parte. Se un server DNS non conosce la risposta passa la richiesta al livello superiore, finché qualcuno non conosce l'indirizzo IP.

Livello applicazione

- Il **livello applicazione**: primitive e protocolli per spedire e ricevere dati delle applicazioni
- I livelli Sessione e Presentazione sono raramente implementati sugli host di Internet
- Il livello applicazione si appoggia sul livello trasporto (in particolare sul protocollo TCP)
- Esempi di famose applicazioni di rete e servizi del livello applicazione
 - **Posta elettronica (E-mail)**: basati su protocolli di livello applicazione SMTP, POP3, IMAP
 - Simple Mail Transfer Protocol (SMTP): per la spedizione e trasporto dei messaggi
 - Post Office Protocol 3 (POP3): per la consegna dei messaggi all'utente
 - Internet Mail Access Protocol (IMAP): alternativa a POP3
 - **World Wide Web (WWW)**: basato su applicazione e protocollo HTTP
 - Hyper Text Transfer Protocol (HTTP): protocollo per trasferire pagine web
 - **DNS**: Domain Name Service (protocollo DNS)



Reti di Calcolatori - introduzione

Il livello applicazione dei protocolli di Internet contiene l'implementazione delle funzioni e dei servizi che permettono alle applicazioni di rete in esecuzione sull'host di spedire e ricevere i dati. I protocolli sottostanti di Presentazione e Sessione, previsti dallo Standard ISO/OSI, non sono quasi mai considerati nell'architettura dei protocolli di Internet.

Il livello Applicazione si appoggia direttamente sul livello trasporto e, in particolare, molte applicazioni che richiedono servizi connection-oriented si basano sul protocollo TCP, attraverso numeri di porta che nel tempo sono diventati standard “de facto”.

Ad esempio, la spedizione e il trasferimento dei messaggi di posta elettronica, basati sul protocollo di livello applicazione Simple Mail Transfer Protocol (SMTP) è comunemente associata alla porta di livello applicazione 25. La porta 80 è destinata al protocollo di trasferimento di ipertesti HyperText Transfer Protocol (HTTP) alla base del trasferimento delle pagine di siti del World Wide Web.

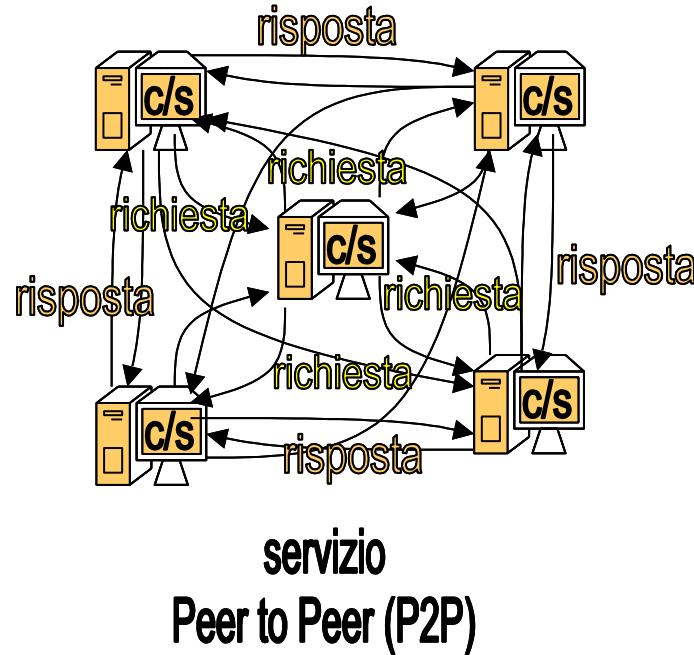
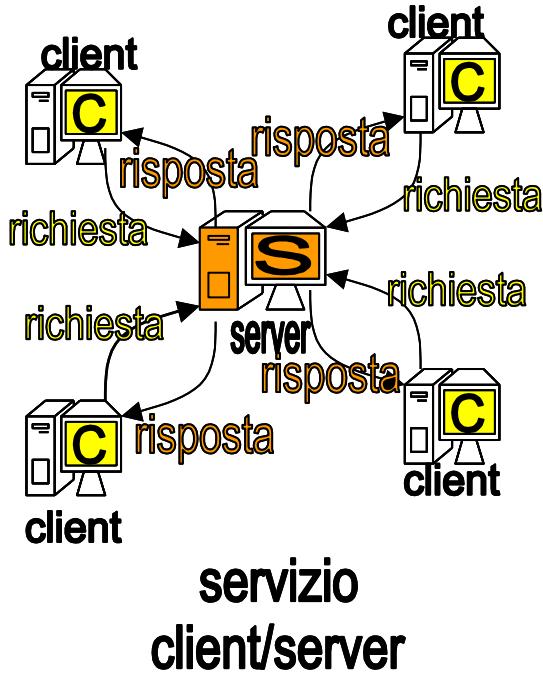
Altri esempi di protocolli e servizi che si collocano al livello applicazione sono il protocollo e servizio di Domain Name Service (DNS) e i protocolli IMAP e POP3 per la consegna della posta elettronica.

Una dettagliata illustrazione sul mondo dei servizi e protocolli applicativi di Internet sarà oggetto di un modulo apposito.

Servizi Client/Server e Peer to Peer

Le applicazioni e i servizi su Internet possono essere realizzati secondo due **modalità architetturali**

- **Architettura Client/Server**
 - I Client sono host che spediscono richieste di servizio
 - I Server sono host sui quali sono in esecuzione i servizi che soddisfano le richieste
 - Esempio: servizio DNS, servizio World Wide Web, servizio posta elettronica
- **Architettura Peer to Peer (P2P)**
 - Tutti gli host sono contemporaneamente sia client che server
 - Ogni host agisce da Server cercando di soddisfare le richieste ricevute, se possibile
 - Ogni host agisce da Client quando spedisce ad altri host le richieste, o per se stesso, o per soddisfare richieste di terzi
 - Esempio: servizi di condivisione dati (file-sharing): Freenet, Gnutella, Kazaa
- Servizi ibridi:
 - Esistono server che aiutano solo a trovare in fretta i giusti host Peer to Peer
 - Esempio: Napster (file-sharing)



Due esempi di applicazione o servizio basati su architettura Client/Server e Peer to Peer (P2P).

La figura a sinistra mostra un'applicazione o un servizio basato sull'architettura Client/Server. Nell'esempio esiste un host server S al quale quattro host Client C inviano le loro richieste. Il server S è l'unico destinatario possibile per le richieste da parte dei client. Ad ogni richiesta, il server S restituisce una risposta del servizio direttamente al client corrispondente. Nel paradigma architettonicale Peer to Peer, invece, esistono cinque host che agiscono sia da Client che da Server. Un host può sia generare richieste verso altri host (i suoi peer host), sia fornire risposte, se è in grado di completare il servizio richiesto.

Reti di Calcolatori - introduzione

Le applicazioni e i servizi su Internet possono essere realizzati secondo almeno due modalità architettoniche distinte: Architettura Client/Server e architettura Peer to Peer (P2P).

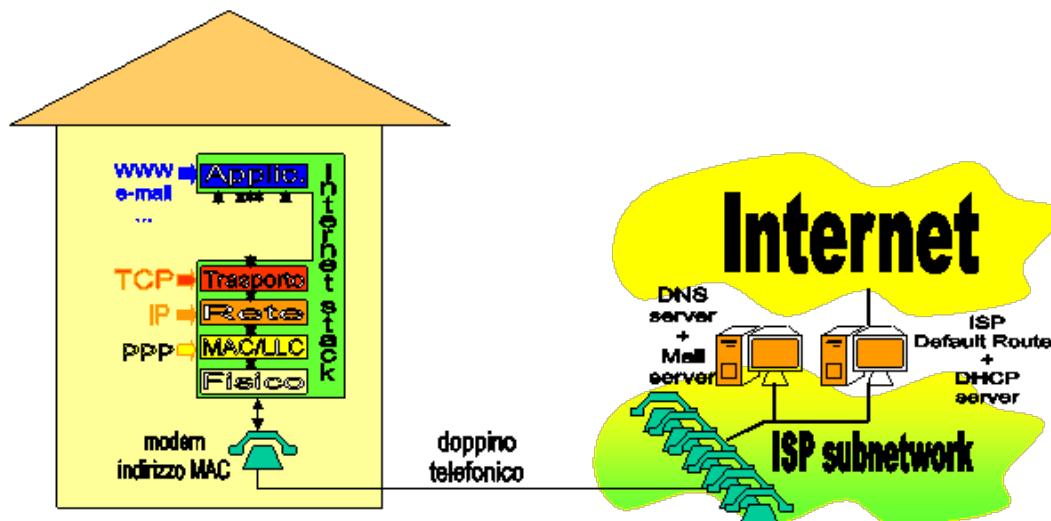
Nell'architettura Client/Server, i Client sono host che spediscono richieste di servizio ai Server. I Server sono i soli host sui quali sono in esecuzione i servizi che permettono di soddisfare le richieste. Un esempio di servizi di tipo Client/Server sono: il servizio DNS, dove ogni host può agire da client spedendo richieste degli indirizzi IP ai DNS server, oppure il servizio World Wide Web, e il servizio di posta elettronica, entrambi basati su client che chiedono pagine web o spediscono e-mail, e server che mantengono le informazioni o memorizzano le e-mail spedite.

Nell'architettura Peer to Peer (P2P), invece, tutti gli host sono contemporaneamente sia client che server. Ogni host agisce da Server cercando di soddisfare, se possibile, le richieste ricevute da altri host. Ogni host agisce da Client quando spedisce ad altri host le sue richieste, o per conto personale, o per cercare di soddisfare richieste di terzi. Un esempio di servizi P2P sono: i servizi di condivisione dati (file-sharing) basati su protocolli Freenet, Gnutella, Kazaa.

Esistono anche servizi ibridi, nei quali esistono server che aiutano solo a trovare più rapidamente gli host P2P migliori per comunicare e implementare servizi P2P (esempio: file-sharing con Napster).

Configurazione TCP/IP

- Riassunto: cosa serve per **configurare un host per la connessione a Internet?**
 - Esempio: computer domestico e Internet Service Provider (ISP) via modem.
- Installare il dispositivo o scheda di rete
 - Creare istanza dello stack TCP/IP e PPP per il dispositivo (modem)
 - Installare **firewall** per impedire accessi esterni all'host?
- Informazioni da fornire (manualmente, o attraverso DHCP server dell'ISP)
 - Indirizzo IP
 - Default Router e maschera di rete (netmask)
 - Indirizzo IP del Server DNS
 - Indirizzo IP dei Server DHCP e SMTP, POP3, IMAP?



Reti di Calcolatori - introduzione

Schema di collegamento del computer domestico a Internet.

La figura mostra un computer domestico, rappresentato dalla pila di protocolli adottati, per il collegamento domestico via modem a Internet. I protocolli utilizzati sono: al livello applicazione tutti i protocolli utili, ad esempio SMTP per la posta elettronica e HTTP il WWW. A livello trasporto e rete viene configurata la coppia TCP/IP. Per il livello LLC/MAC, assumendo di usare un modem, viene adottato il protocollo PPP. Sul lato dell'Internet Service Provider (ISP) si trova: un insieme di modem ai quali collegarsi (telefonando), e quindi una rete locale sulla quale si trovano tutti i server necessari: server DNS, server DHCP, server Mail, e soprattutto il Default Router che inoltra il traffico da e verso il resto di Internet.

Si fornisce ora un breve esempio riassuntivo che illustra quali informazioni e quale configurazione sia necessaria per il collegamento di un host a Internet. Si considera il caso di un host domestico, che viene connesso attraverso il servizio di connessione fornito da un Internet Service Provider (ISP), via modem telefonico.

Un modem è un dispositivo di rete che trasmette i bit a un altro modem attraverso la linea telefonica.

Per il collegamento dell'host occorre quindi: installare il modem e applicare al modem i protocolli PPP per il livello LLC, e TCP/IP per i livelli trasporto e rete. Tali protocolli sono di solito forniti con il sistema operativo in uso.

A questo punto, all'atto della connessione telefonica, occorre configurare (manualmente o meglio automaticamente, attraverso DHCP) le seguenti informazioni: indirizzo IP dell'host (relativo alla sottorete dell'ISP), maschera di rete, Indirizzo IP del default router, e indirizzo IP del DNS server. Possono poi essere inseriti gli indirizzi IP di servizi applicativi come posta elettronica (server SMTP, POP3 o IMAP).

Al rilascio della comunicazione, automaticamente l'host viene cancellato dalla sottorete dell'ISP e l'indirizzo IP eventualmente riciclato per altri dispositivi.

Cenni sulla sicurezza in rete

- Prevenire e contrastare la diffusione di programmi dannosi (**computer virus**)
 - Possono essere ricevuti via e-mail, via web e se eseguiti causano perdita dei dati
- Prevenire e contrastare **l'accesso a sistemi di rete** privati connessi a Internet
 - Bloccare l'accesso ai dati e alle applicazioni degli intrusi
 - Filtrare i pacchetti a livello rete
 - Firewall: è il primo router visto dall'esterno della rete, che filtra i pacchetti
 - Consentire l'accesso ai dati richiesti da applicazioni e dal personale autorizzato
 - Lista del personale autorizzato (registration) e autenticazione (login e password)
 - Application gateway: è un server che verifica che chi usa certe applicazioni rischiose sia autorizzato a farlo
- **Segretezza (Privacy) dei dati trasmessi**
 - Uso di tecniche di crittografia e cifratura dei dati

Reti di Calcolatori - introduzione

○

Introduciamo brevemente alcuni aspetti legati alla sicurezza dei sistemi e dei dati della comunicazione in rete.

Un primo aspetto della sicurezza consiste nel dotarsi degli strumenti per prevenire e contrastare la diffusione di programmi dannosi (computer virus), che attraverso le reti possono diffondersi all'interno di posta elettronica, documenti scaricati, e se eseguiti possono causare la perdita di dati importanti per l'utente e per il funzionamento del sistema stesso.

Un secondo aspetto riguarda la prevenzione e il contrasto dell'accesso indiscriminato ai sistemi di rete privati.

In particolare, occorre bloccare l'accesso ai dati e alle applicazioni di intrusi: per fare questo è possibile filtrare i pacchetti di dati a livello rete, attraverso dei router speciali detti firewall. I firewall sono posti di solito come il primo router che i pacchetti incontrano dall'esterno entrando nella rete privata.

D'altra parte, potrebbe essere opportuno consentire l'accesso dall'esterno della rete ai dati e alle applicazioni di utenti autorizzati. Le tecniche usate in tal senso si basano su liste di persone autorizzate e registrate (registration) e sulla verifica dell'identità basata su autenticazione (login e password privata). Un application gateway è un server che verifica tutte le applicazioni rischiose, consentendone l'uso solo da parte delle persone autorizzate.

L'ultimo aspetto da considerare fa parte in maniera indiretta delle questioni di sicurezza.

Si tratta della segretezza (privacy) dei dati trasmessi in rete (che tutti potrebbero intercettare).

Le soluzioni in uso si basano su tecniche di crittografia e cifratura dei dati, la cui presentazione, tuttavia, esula dai contenuti di questo modulo.

Servizi differenziati e Internet2

Un aspetto critico della comunicazione su Internet è la mancanza di **garanzie sui tempi** di consegna dei dati (qualità del servizio).

- posso imporre che i pacchetti arrivino tutti (servizio connection-oriented di TCP)
- non posso imporre che i pacchetti arrivino entro tempi fissati su Internet
 - manca una progettazione opportuna dei router
 - Problema: i router fanno semplicemente del loro meglio
 - Ma smistano tutti i pacchetti come se avessero tutti la stessa urgenza
- Internet2 e i Servizi Differenziati
 - Una nuova infrastruttura per Internet2: nuove linee dorsali e nuovi router
 - I router spediscono prima i pacchetti urgenti, e poi quelli non urgenti.
 - riservare le risorse lungo il cammino per comunicare i dati
 - Garantisce la qualità del servizio su tutto il collegamento mittente-destinatario

Uno degli aspetti critici della comunicazione su Internet al giorno d'oggi è la mancanza di garanzie sui tempi di consegna dei dati (qualità del servizio). Attraverso il protocollo TCP si possono garantire servizi connection-oriented, senza perdita di dati, malgrado il fatto che i router possano perdere o disordinare i pacchetti. Non è però possibile garantire che i pacchetti arrivino entro un tempo fissato. Questo fatto dipende da una carenza progettuale e architetturale di Internet, alla quale i protocolli possono difficilmente porre rimedio. I router di Internet, infatti, smistano tutti i pacchetti con la stessa urgenza, quindi in situazioni di congestione i vincoli di tempo di consegna possono cadere.

La nuova struttura di Internet2 e dei Servizi Differenziati permette di garantire i requisiti di qualità del servizio di comunicazione che Internet non può supportare. Tutto si basa essenzialmente su nuovi router, che sono in grado di spedire prima i pacchetti urgenti e poi tutti gli altri, se avanza tempo. Ciò va di pari passo allo sviluppo di reti sulle quali sia possibile riservare in anticipo le risorse, lungo tutto il cammino tra mittente e destinatario dei dati.

In questo modo è possibile supportare comunicazione con garanzie di qualità del servizio su scala globale.

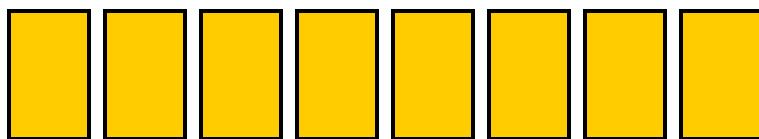
Breve ripasso di aritmetica binaria (corso di reti di calcolatori)

Luciano Bononi

Email: luciano.bononi@unibo.it

La rappresentazione dei dati sul calcolatore

- i bit possono essere considerati in sequenza (in memoria)
 - sequenza di 8 bit = 1 Byte



0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 1

0 0 0 0 0 0 1 0

0 0 0 0 0 0 1 1

⋮

1 1 1 1 1 1 1 0

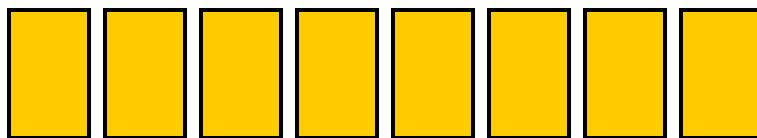
1 1 1 1 1 1 1 1

2^n possibili sequenze
diverse da n bit

n = numero di bit considerati

La rappresentazione dei dati sul calcolatore

- Dato un byte come possiamo associarvi i simboli o valori?
 - sequenza di 8 bit = 1 Byte (primo esempio poco utile)



Solo valori da 1 a 8 ?
non sfrutto tutte le
possibili combinazioni!!!

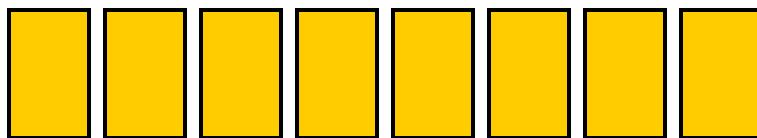
0	0	0	0	0	0	0	1	= valore 1
0	0	0	0	0	0	1	0	= valore 2
0	0	0	0	0	1	0	0	= valore 3
0	0	0	0	1	0	0	0	= valore 4
0	0	0	1	0	0	0	0	= valore 5
0	0	1	0	0	0	0	0	= valore 6
0	1	0	0	0	0	0	0	= valore 7
1	0	0	0	0	0	0	0	= valore 8



La rappresentazione dei dati sul calcolatore

▪ Dato un byte come possiamo associarvi i simboli o valori?

- sequenza di 8 bit = 1 Byte



Idea: sfrutto tutte le possibili combinazioni diverse di 8 bit
256 valori [0..255]

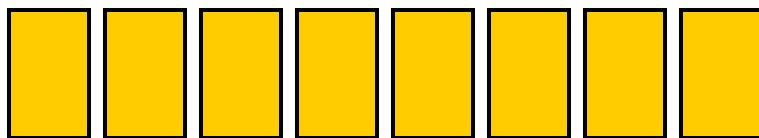
0	0	0	0	0	0	0	0	= valore 0
0	0	0	0	0	0	0	1	= valore 1
0	0	0	0	0	0	1	0	= valore 2
0	0	0	0	0	0	1	1	= valore 3
0	0	0	0	0	1	0	0	= valore 4
:	:	:	:	:	:	:	:	
1	1	1	1	1	1	1	0	= valore 254
1	1	1	1	1	1	1	1	= valore 255



La rappresentazione dei dati sul calcolatore

▪ Dato un byte come possiamo associarvi i simboli o valori?

- sequenza di 8 bit = 1 Byte



Idea: sfrutto tutte le possibili combinazioni diverse di 8 bit
256 simboli

0 0 0 0 0 0 0 0 = simbolo A

0 0 0 0 0 0 0 1 = simbolo B

0 0 0 0 0 0 1 0 = simbolo C

0 0 0 0 0 0 1 1 = simbolo E

0 0 0 0 0 1 0 0 = simbolo F

: : : : : : : : :

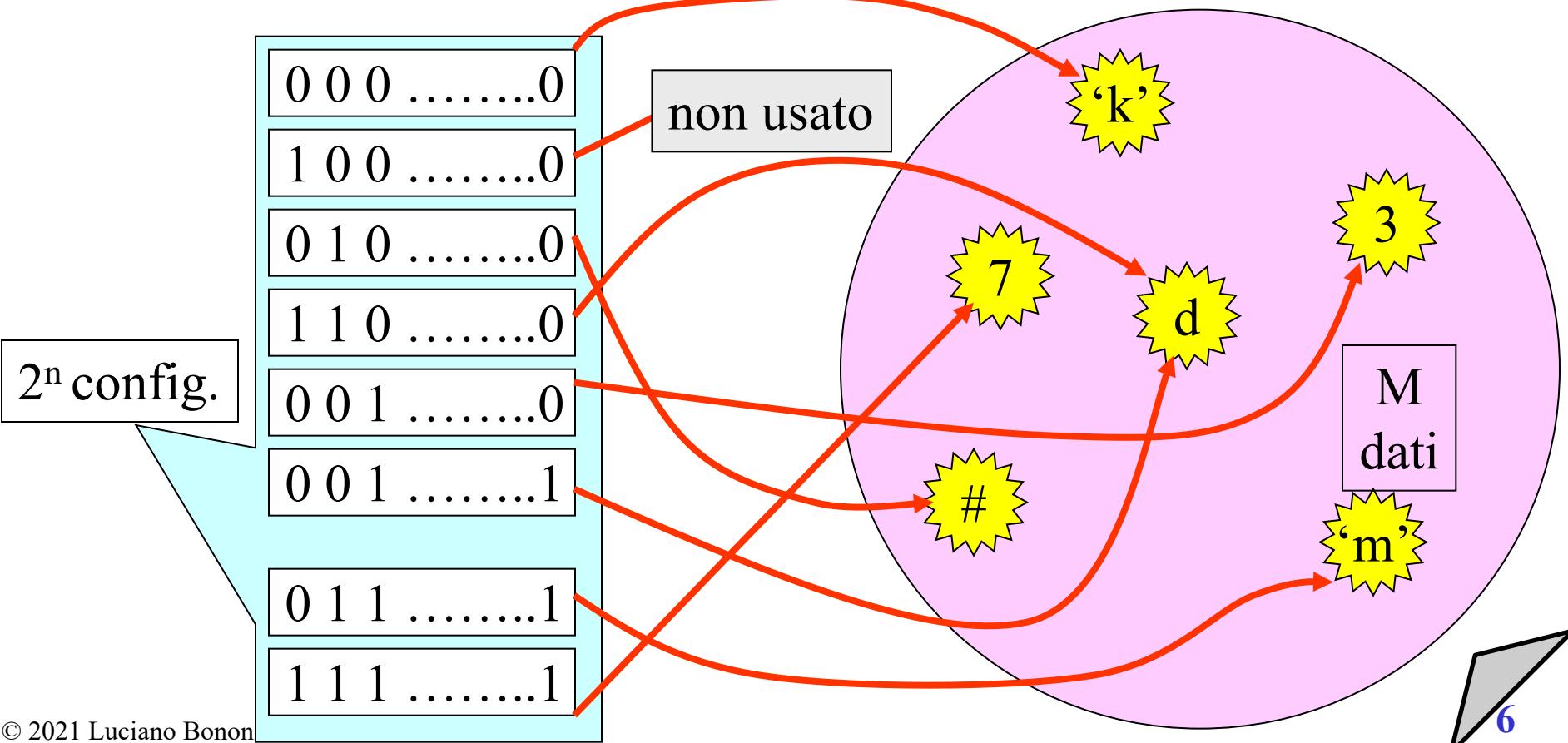
1 1 1 1 1 1 1 0 = simbolo %

1 1 1 1 1 1 1 1 = simbolo \$

Codici binari: convenzioni per rappresentare info.

Funzioni dall'insieme delle 2^n configurazioni di n bit ad un insieme di M informazioni o dati
(valori, simboli, istruzioni, ecc.).

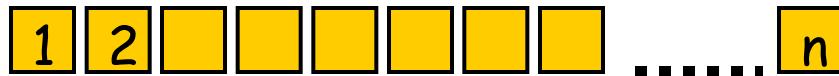
Condizione necessaria per la codifica completa: $2^n \geq M$



Codici binari: quanti sono?

La scelta di un codice è condivisa da sorgente e destinazione ed ha due gradi di libertà:

- il numero di bit n (qualsiasi, a patto che sia $2^n \geq M$)



- l'associazione tra configurazioni e informazioni; a parità di n e di M le associazioni possibili sono

$$C = 2^n! / (2^n - M)!$$

$$n = 1, M = 2$$

$C = 2$ (logica pos. e neg.)

$$n = 2, M = 4$$

$C = 24$

$$n = 3, M = 8$$

$C = 64.320$

$$n = 4, M = 10$$

$C = 29.000.000.000$

Sistemi di numerazione

Posizionali

- il valore di un simbolo dipende dalla posizione che esso occupa all'interno della configurazione, seguendo una legge nota. I vari sistemi di numerazione posizionale differiscono per la scelta della base B . La base B indica il numero di simboli usati.
 - **decimale $B=10$, binario $B=2$, ottale $B=8$, esadecimale $B=16$**

Non posizionali

Il valore di un simbolo non dipende dalla posizione che esso occupa all'interno della configurazione. (es: Numeri Romani)

Interpretazione (funzione valore)

- Nei sistemi posizionali, i simboli di una configurazione possono essere interpretati come i coefficienti del seguente polinomio [1] (detto funzione Valore)

$$V = \sum_{i=-m}^{n-1} d_i \cdot B^i$$

B = base

d_i = i-esima cifra $\in [0..B-1]$

n = numero di cifre parte intera

m = numero di cifre parte frazionaria

La virgola e' posta tra le cifre di posizione 0 e -1.

Interpretazione (funzione valore)

Esempio: sistema decimale

Il numero **245.6** decimale può essere rappresentato come segue:

B = 10

n=3

m=1

d = {**2,4,5,6**}

base

numero cifre parte intera

numero cifre parte frazionaria

insieme delle cifre

cifra	2	4	5	6
-------	---	---	---	---

posizione	2	1	0	-1
-----------	---	---	---	----

peso	10^2	10^1	10^0	10^{-1}
------	--------	--------	--------	-----------

$$V = \sum_{i=-m}^{n-1} d_i \cdot B^i = 2 \cdot 10^2 + 4 \cdot 10^1 + 5 \cdot 10^0 + 6 \cdot 10^{-1} = 245.6$$

Esempio di interpretazione valore binario naturale

Esempio: Quale è il valore decimale corrispondente al numero binario **1101.010₂** ?

cifra ₂	1	1	0	1	.	0	1	0...
peso	2^3	2^2	2^1	2^0	.	2^{-1}	2^{-2}	2^{-3}
valore	$1 \cdot 8$	$1 \cdot 4$	$0 \cdot 2$	$1 \cdot 1$.	$0 \cdot 1/2$	$1 \cdot 1/4$	$0 \cdot 1/8$

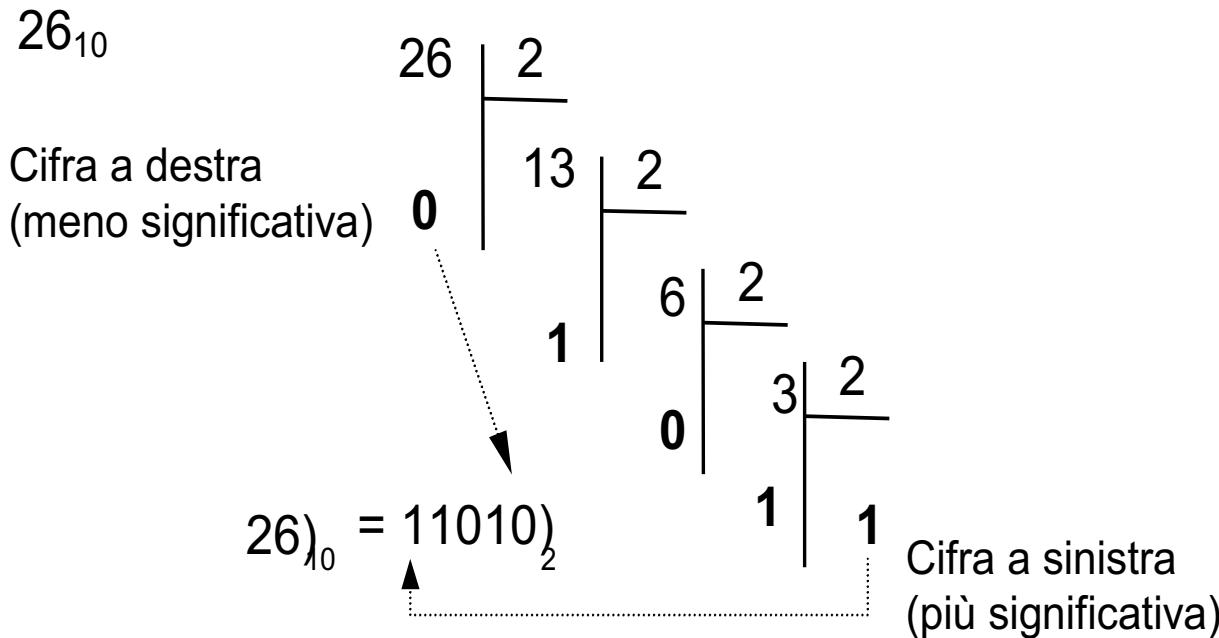
$$1101.010_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^0 + 1 \cdot 2^0 = 13.25_{10}$$

Metodo della divisione: Base 10 -> Base B

Es. base 2:

Per valori interi positivi:

Per ottenere il valore in base B, di un numero intero codificato nel sistema decimale, si procede utilizzando un metodo iterativo di successive divisioni per la base: al termine del procedimento i resti delle divisioni, dall'ultimo al primo, rappresentano il valore iniziale in base B.



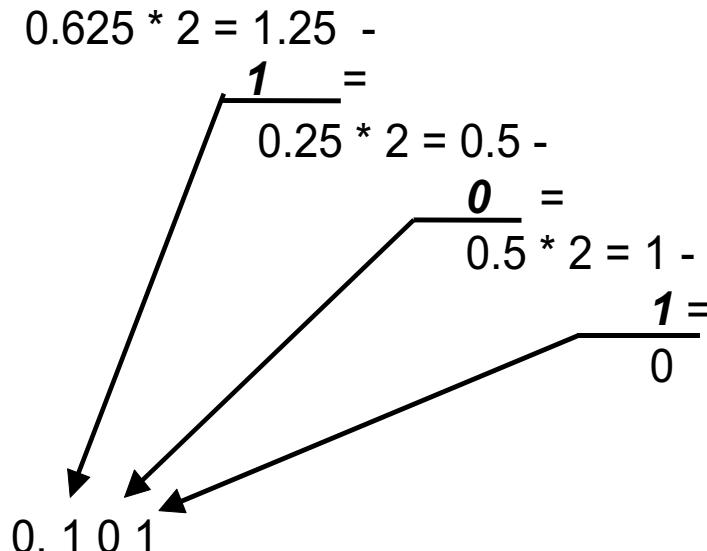
Metodo della divisione: Base 10 -> binario naturale

Es. base 2

Per valori con cifre decimali: si separa la parte intera da quella frazionaria, La parte intera si calcola come nel caso precedente La parte frazionaria si ottiene come segue:

1. Si moltiplica la parte frazionaria per 2
2. Se il numero ottenuto è maggiore di 1, si sottrae 1 e si considera come prima cifra dopo la virgola un ‘1’.
3. Se invece il numero è nella forma 0,..... => la cifra da inserire è uno ‘0’.
4. Si ripete dal passo 1 fino a che il numero di partenza non è zero.

Esempio: $0.625_{10} = 0.101_2$



Metodo veloce (pratico): Base 10 -> binario naturale

Con la pratica, solo dopo avere bene compreso la metodologia è possibile usare un metodo alternativo veloce.

1. Dato il numero N in base 10 iniziale (es. 349), si deve cercare a mente velocemente la più alta potenza del 2 inferiore al numero stesso (sia essa $2^k = 256$, k=8)
2. Si sottrae 2^k da N ottenendo $349 - 256 = 92$
3. Se il risultato è zero abbiamo finito, altrimenti iteriamo il procedimento fino a che non otteniamo zero dalla sottrazione.
4. Inseriamo un bit a UNO nella notazione binaria di N in posizione k. (considerare che le posizioni partono da 0 a n-1, con n corrispondente al numero di bit totali della rappresentazione binaria usata e consentita).

$$349 - 256 = 93 - 64 = 29 - 16 = 13 - 8 = 5 - 4 = 1 - 1 = 0$$

$$\begin{array}{cccccccc} & 2^8 & & 2^6 & & 2^4 & & 2^3 \\ & \diagdown & & \diagdown & & \diagdown & & | \\ 349 \text{ in binario} = & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ & & & & & & & 0 \\ & & & & & & & \diagup \\ & & & & & & & 1 \end{array}$$

$$\text{Infatti, } V = \sum_{i=-m}^{n-1} d_i \cdot B^i = 1 * 2^8 + 0 * 2^7 + 1 * 2^6 + 0 * 2^5 + 1 * 2^4 + 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0$$
$$= 1 * 256 + 0 + 1 * 64 + 0 + 1 * 16 + 1 * 8 + 1 * 4 + 1 * 1$$

Somma di valori in binario naturale

Assumiamo di avere solo $n=3$ bit a disposizione per rappresentare i valori e il risultato, quindi posso rappresentare $B^n = 8$ valori diversi interi positivi (da 0 a 7).

$$5_{10} + 1_{10} = 6_{10}$$

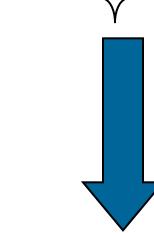
$$\begin{array}{r} 1 \\ 101 + \\ 001 = \\ \hline 110 \end{array}$$

$$2_{10} + 3_{10} = 5_{10}$$

$$\begin{array}{r} 1 \\ 010 + \\ 011 = \\ \hline 101 \end{array}$$

$$5_{10} + 3_{10} = 8_{10}$$

$$\begin{array}{r} 111 \\ 101 + \\ 011 = \\ \hline 1000 \end{array}$$



Overflow!!

Per rappresentare il risultato

della somma di $5_{10} + 3_{10}$ sono necessari 4 bit !

Altre basi: ottale e esadecimale

Quando per la rappresentazione di un numero si utilizzano molte cifre binarie può convenire usare altri sistemi di numerazione.

I sistemi **ottale** ed **esadecimale** sono utilizzati principalmente per rappresentare in modo più compatto i numeri binari.

L'algoritmo della divisione per passare da base 10 a ottale o esadecimale vale anche in questo caso. Provare.

I simboli del sistema **Ottale** sono 8 (da 0 a 7):

{ 0, 1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 13, 14, 15, 16, 17, 20, }

I simboli del sistema **Esadecimale** sono 16 (da 0 a F):

{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 10, 11, 12, 13, 14, }

Cambiamenti di base (metodo veloce)

Esiste un metodo veloce quando base di partenza BP e di arrivo BA sono esprimibili come $BA=BP^k$.

BP:Binario -> BA:Ottale , K=3, (8=2³)

Per passare dalla codifica Binaria a quella Ottale, si raggruppano le cifre binarie a gruppi di 3 (**a partire da destra, allineandosi alla virgola**) e le si sostituiscono con una cifra del sistema ottale.

Esempio : 111**001**010₂ = 7**12**₈

Ottale -> Binario

Per passare dalla codifica Ottale a quella Binaria, si sostituisce ad ogni cifra ottale la corrispondente codifica binaria (composta da 3 cifre).

Esempio : 3**02**₈ = 011**000**010₂

Cambiamenti di base (metodo veloce)

BP: Binario -> BA:esadecimale , K=4, (16=2^4)

Per passare dal codice Binario a quello Esadecimale, si raggruppano le cifre a gruppi di 4 (a partire da destra) e le si sostituiscono con una cifra del sistema esadecimale.

Esempio : $100100011111_2 = 91F_{16}$

Esadecimale -> Binario

Per passare dal codice Esadecimale a quello Binario, si sostituisce ad ogni cifra esadecimale la corrispondente configurazione binaria (composta da 4 cifre).

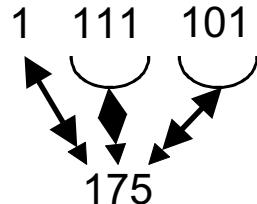
Esempio : $A7F_{16} = 101001111111_2$

Esempi

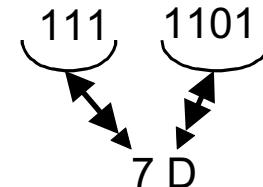
Esempio 1

Codifica del numero $125_{10} = 1111101_2$

In codice Ottale:



In codice Esadecimale:



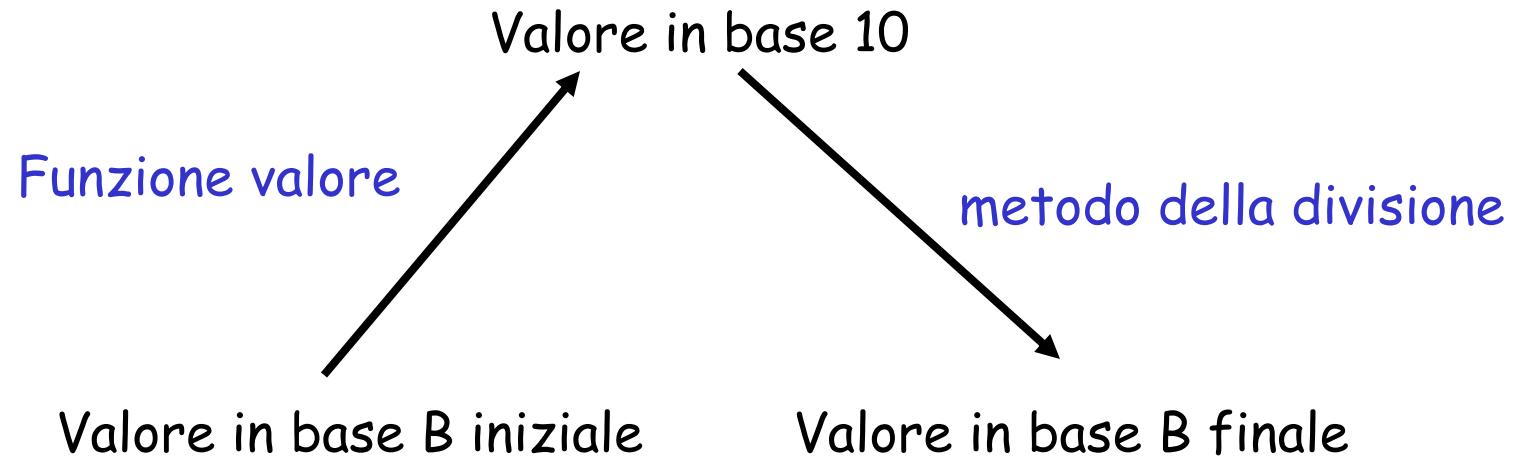
Esempio 2

Decimale	Binario	Ottale	Esadecimale
5	101	5	5
12	1100	14	C
78	1001110	116	4E
149	10010101	225	95

Esempi

Decimale	Binario	Ottale	Esadecimale
0	0000	00	0
1	0001	01	1
2	0010	02	2
3	0011	03	3
4	0100	04	4
5	0101	05	5
6	0110	06	6
7	0111	07	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Riassunto: passaggi di base generici



Codifica dei caratteri ASCII (7 bit -> 128 caratteri)

Il codice ASCII è non ridondante, perchè i simboli che vengono codificati sono in numero pari alle configurazioni ottenibili con 7 cifre binarie.

	000	001	010	011	100	101	110	111	MSB
0000	NUL	DLE		0	@	P	°	p	
0001	SOH	DC1	!	1	A	Q	a	q	
0010	STX	DC2	“	2	B	R	b	r	
0011	ETX	DC3	#	3	C	S	c	s	
0100	EOT	DC4	\$	4	D	T	d	t	
0101	ENQ	NAK	%	5	E	U	e	u	
0110	ACK	SYN	&	6	F	V	f	v	
0111	BEL	ETB	‘	7	G	W	g	w	
1000	BS	CAN	(8	H	X	h	x	
1001	HT	EM)	9	I	Y	i	y	
1010	LF	SUB	*	:	J	Z	j	z	
1011	VT	ESC	+	;	K	[k	{	
1100	FF	FS	,	<	L	\	l		
1101	CR	GS	-	=	M]	m	}	
1110	SO	RS	.	>	N	^	n	~	
1111	SI	US	/	?	O	_	o	DEL	

LSB

Wireshark Filtering in Computer Networks

Federico Montori, PhD
University of Bologna

Who am I

Dr. Federico Montori

PostDoc research fellow in Internet of Things, Mobile Crowdsensing and IoT Data Analytics

- UniBo: <https://www.unibo.it/sitoweb/federico.montori2/>
- IoT Prism Lab: <http://iot-prism-lab.nws.cs.unibo.it/team/federico-montori/>
- email: federico.montori2@unibo.it

Goals

- Understanding by doing
- Look for specific network information
- See in practice things that you learned
- Give a kickoff to getting hands dirty

Outline

1. General introduction to packet sniffing
2. Tutorial on Wireshark
3. Capture filters and Display filters
4. Hands-on exercises

The slides are inspired to the WireShark course by Dr. Luca Bedogni

All material is a courtesy of WireShark Labs, J.F. Kurose, K.W. Ross
(https://gaia.cs.umass.edu/kurose_ross/wireshark.htm)

Pre-requisites

In order to participate actively in the hands-on tutorial you need to do a couple of steps:

- Download and install WireShark
(<https://www.wireshark.org/download.html>)
 - You may want to do it in a Virtual Machine if you feel you want to go hardcore later...
 - Linux users can find it on the repo
- Download the pre-set WireShark traces at
<http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip>

Packet Sniffers

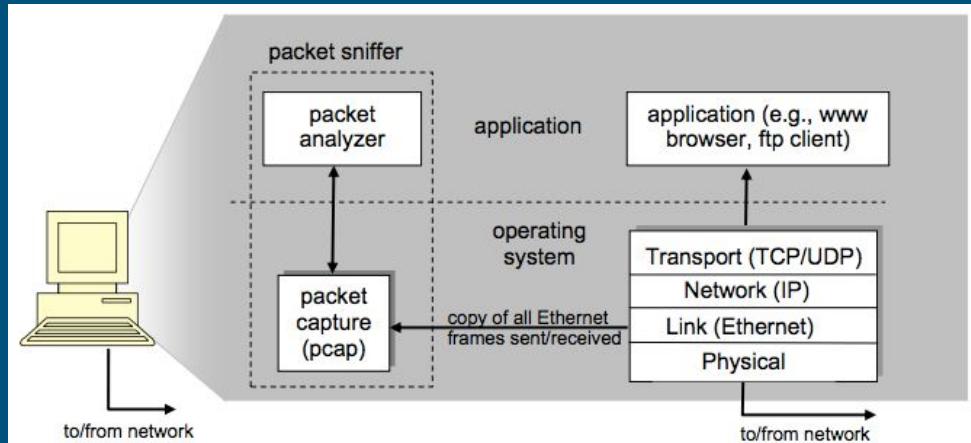
Packet sniffing is the operation of capturing data flowing through the network to look for information in network packets.

Frequently used by system administrator to troubleshoot network issues:

- Why traffic is slow
- Detect intrusions

Considered security tools:

- Because it gives all the tools to assess it



Packet Sniffers

It is a passive technique:

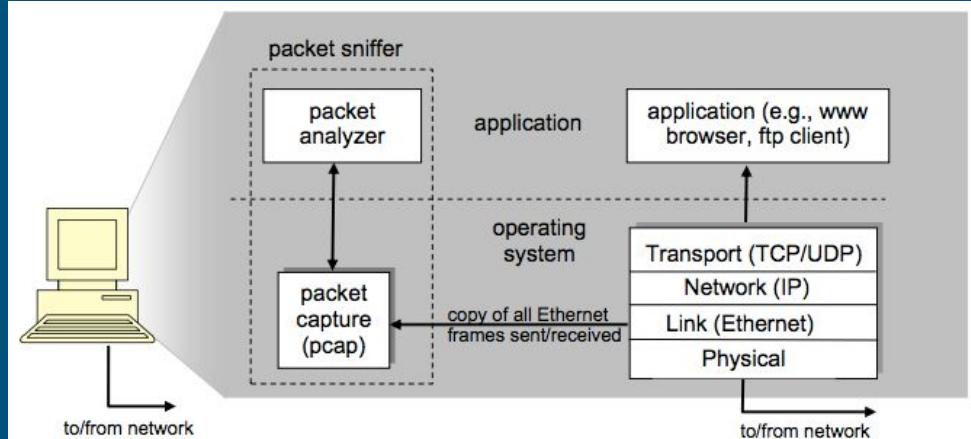
- One or more of your network interfaces to listen for everything (or a subset...)
- Packets are copied and displayed to the user
- The packet sniffer is just listening
- Actually, information does not change

The packet capture library (PCAP):

- Receives the packets with filters.

The packet analyzer:

- Shows the packet contents
- Decodes nested fields



PROMISCUOUS MODE:

- Don't throw it away if it isn't for you

How do they work?

Intuitively, you may think that when systems communicate over the network their packets go directly to the destination...

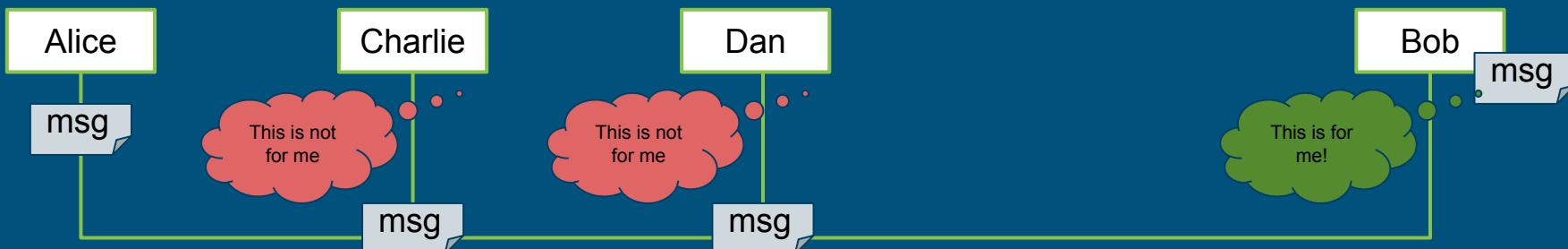
- Instead they are sent in broadcast
- Every node in the network overhears the packet
- The node checks if it is the destination, or if it needs to reroute it, or discard it
- Something against it? Wait for it...



How do they work?

Intuitively, you may think that when systems communicate over the network their packets go directly to the destination...

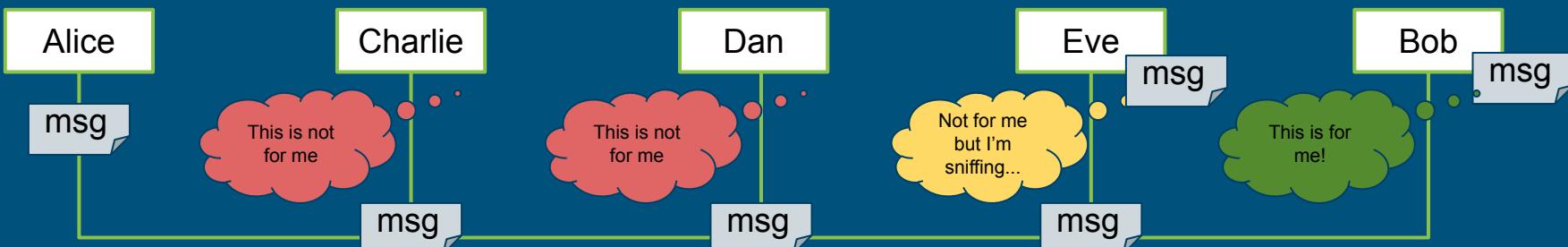
- Instead they are sent in broadcast
- Every node in the network overhears the packet
- The node checks if it is the destination, or if it needs to reroute it, or discard it
- Something against it? Wait for it...



How do they work?

Intuitively, you may think that when systems communicate over the network their packets go directly to the destination...

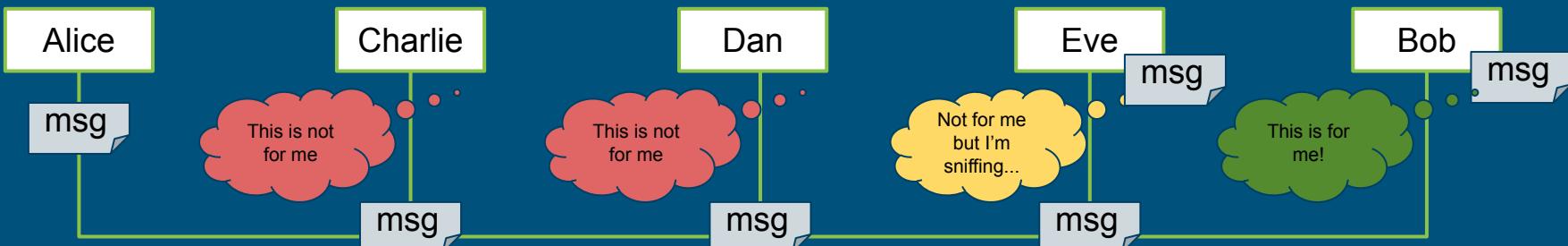
- Instead they are sent in broadcast
- Every node in the network overhears the packet
- The node checks if it is the destination, or if it needs to reroute it, or discard it
- Something against it? Wait for it...



How do they work?

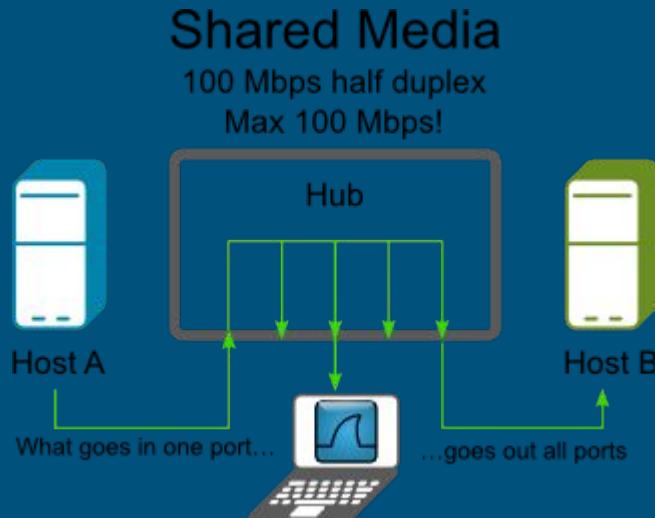
Well, ACTUALLY over the last couple of decades you can't really sniff everything....

- Switched ethernet LANs
 - Your host is only on one segment... ARP may be your enemy here
- Only Broadcasts, multicasts and your segment (plus all wireless obviously)



Old Network Setups

If your sharing media is below OSI layer 2, well, then it works like wireless pretty much...



Remember:

- Hubs (layer 1)
- Repeaters (layer 1)
- Switches (layer 2)
- Routers (layer 3)

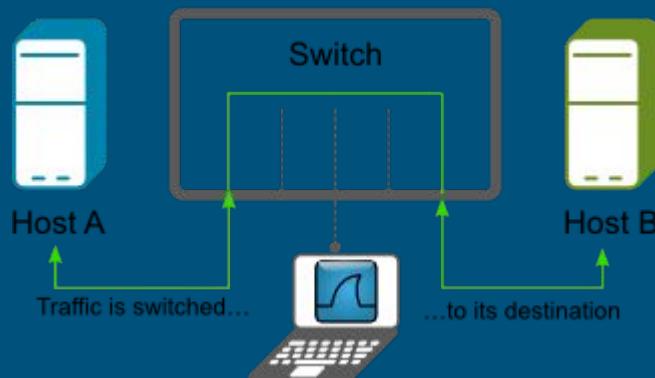
Read more at:

https://wiki.wireshark.org/CaptureSetup/Ethernet#Switched_Ethernet

Really Sniffing Everything

This applies only **if you are the network administrator** or you can mess with the wires of the switch (not advised).

Switched Media

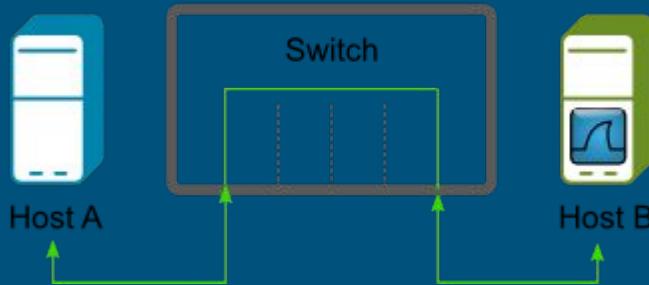


Read more at:

https://wiki.wireshark.org/CaptureSetup/Ethernet#Switched_Ethernet

Really Sniffing Everything

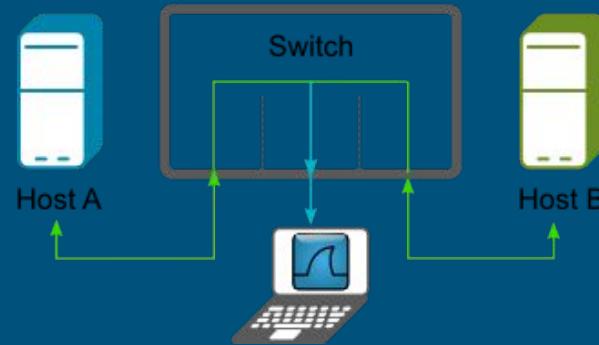
Switched Media — Same Computer



If you want to capture the traffic to/from B, just sniff from B

Needless to say, you need to have access to B...

Switch + Monitor Port

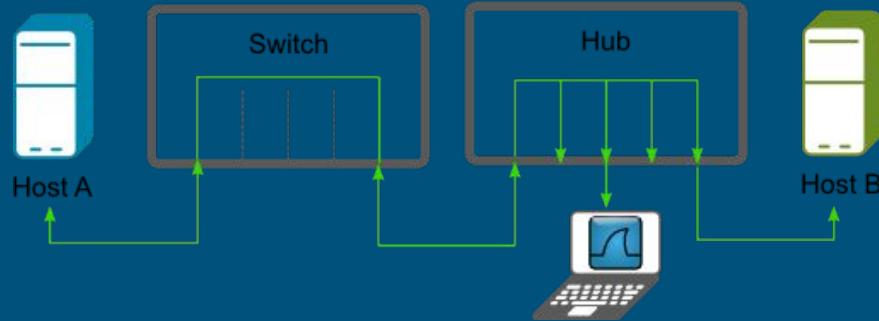


Use a router or a switch with a monitor port

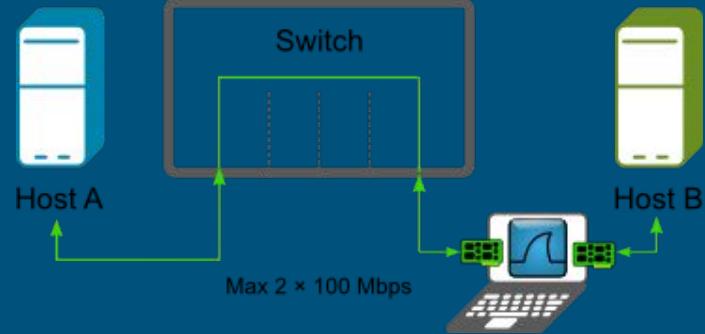
PORT MIRRORING \$\$

Really Sniffing Everything

Switched Media — "Hubbing Out"



Machine-in-the-middle



Use an old hub on the network segment.

You can also use a TAP, which is more sophisticated.

You need to unplug stuff...

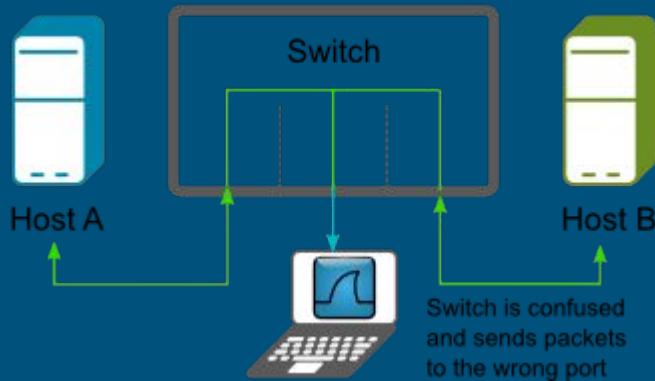
You can set up your monitoring machine as a bridge, however you will need two separate NICs...

Really Sniffing Everything

The **ILLEGAL** ways...

- ARP Poisoning
 - Trick the machines into believing that your MAC is the MAC of the other machine, so all the traffic is directed to you.
- MAC Flooding
 - Send plenty of fake ARP messages to the router until its table is filled and no ARP is used anymore to keep up the pace.

Switch — Man In The Middle



← Please note these are not nice and you should not try them unless the LAN is yours.

Did I say Wireless?

Well, that's also a problem... If wireless cuts you out at the PHY level then you're done.

- TDMA in general
 - LTE...
- For WiFi we are pretty much covered...

Simple Example

Suppose you want to visit unibo.it (consider a wireless environment)

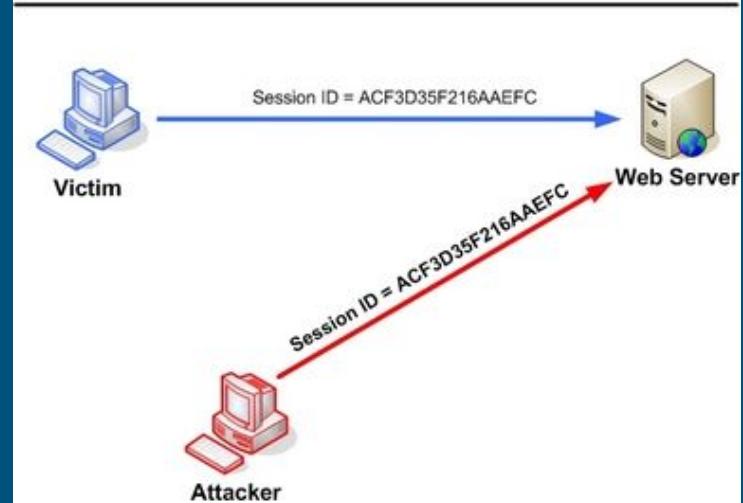
- Basically, you shout “Somebody give me unibo.it!”
- The message is overheard by anyone on the network
 - Including the router, who is the intended recipient
- The router sends it to the destination
- Once it receives the answer, it send the message again on the network
- Everybody overhears it
 - Including you, who are the intended recipient

What can you sniff?

DATA LINK FRAMES!

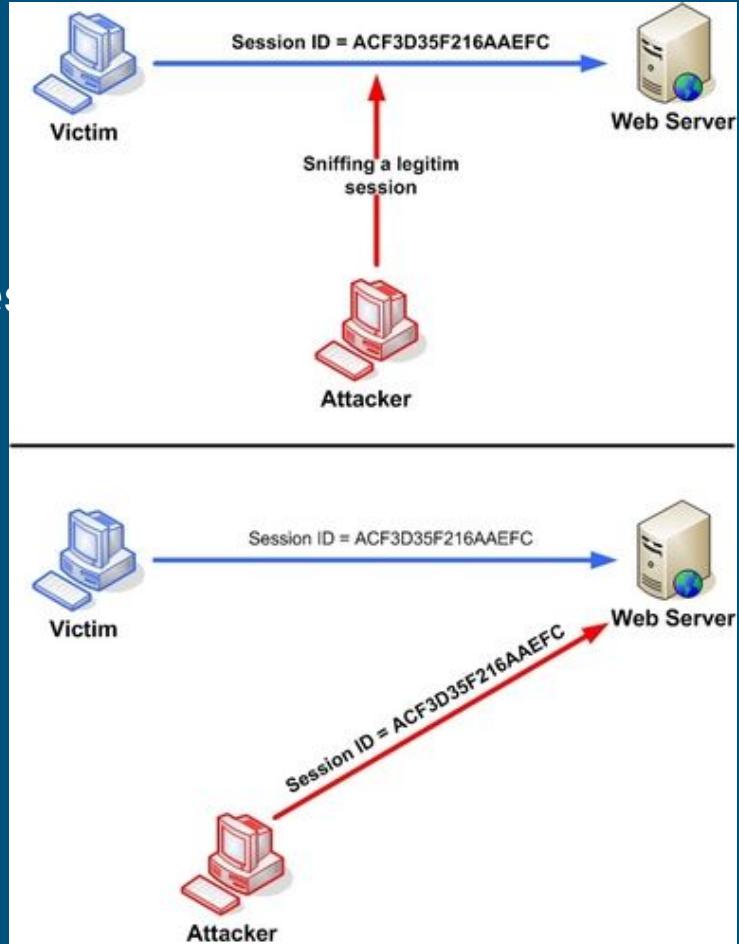
Then the analyzer decodes them (unless they are encoded) and gives you back the highest layer (plus all the envelopes).

Clearly, you find the higher layer datagram in the payload of the smaller layer one...



What can you sniff?

- Basically, all the information sent in clear
- Anyone with a packet sniffer can gain access to such information
- If the connection is encrypted, the information is more secure
 - But still, you are receiving it
- Consider if your user credentials for a harmless website are sent in plain text
 - And you use the same credentials for gmail
 - ... and for your bank account ...
- Typical man-in-the-middle
 - Example: cookie hijacking



Wireshark



What is WireShark?

It is a network analyzer tool: it allows us to see all the packets that go through a network...

- Why is my network stuck every Friday evening from 6PM to 8PM?
- Why computer X can't connect to the Internet?
- Why the A department can't connect to the internal servers?

Wireshark helps us troubleshoot the network (correct use)

Available for Windows/MAC OS/Linux: <https://www.wireshark.org/download.html>

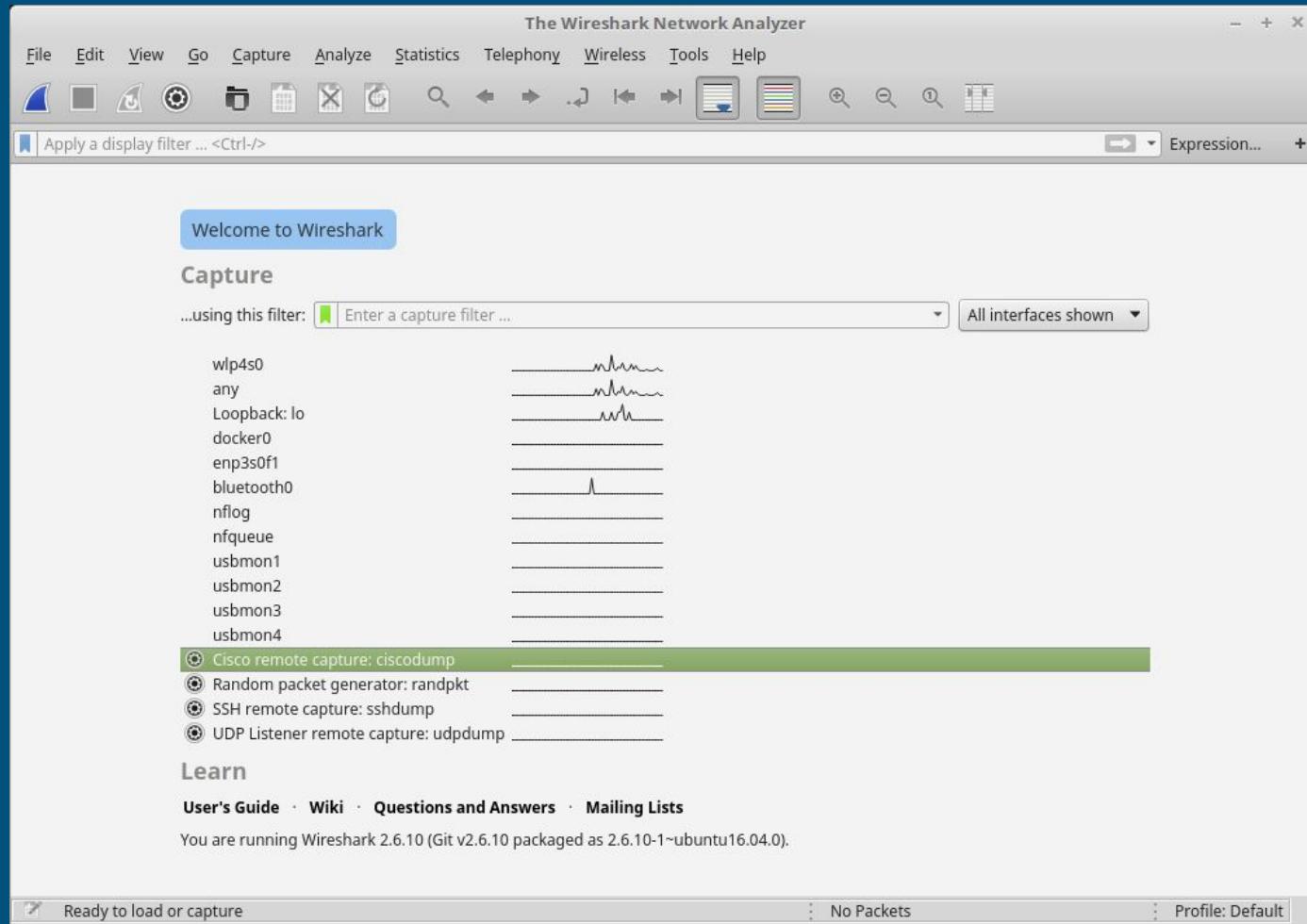
- Open Source with GUI



What is Wireshark?

- What people use wireshark for:
 - Network administrators -> troubleshoot network problems
 - Network security engineers -> examine security problems
 - QA engineers -> verify network applications
 - Developers -> debug protocol implementations
 - People -> learn network protocol internals
 - **Attackers** -> you can imagine...
- Some features:
 - Capture live packet data and display it
 - Import network traces and save them
 - Filtering/Coloring/Search
 - Create network statistics

How it looks like



How it looks like

The screenshot shows the Wireshark interface with several sections highlighted by green boxes:

- Toolbar**: Located at the top left.
- Packet List**: A large section on the right displaying a list of network packets. The 57th packet is selected, showing detailed information about a DNS query from 192.168.1.175 to 192.168.1.254.
- Packet Details**: A box below the packet list showing the detailed structure of the selected packet (Frame 57).
- Packet Content**: A box below the details showing the raw hex and ASCII representation of the selected packet.
- Stats**: A box at the bottom right showing summary statistics: Packets: 1088 · Displayed: 1088 (100.0%) · Dropped: 0 (0.0%).

Key details from the selected packet (Frame 57):

- Time: 2001-07-15T11:45:00.000Z
- Source: 192.168.1.175
- Destination: 192.168.1.254
- Protocol: DNS
- Length: 125 bytes
- Info: Standard query 0x0b07 A e0l.unibo.it
- Hex: 0000 29 b0 01 b3 f9 6a f8 59 71 85 2e ab 08 00 45 08 ...j.Y q...E
0010 00 3a 03 e7 40 00 40 11 b1 c0 c8 a5 01 af c8 a8 ...@0...
0020 01 fe 8d 59 08 35 08 26 ab d2 88 b7 01 00 00 01 ...Y-5-4...
0030 00 00 00 00 00 00 03 65 6f 05 75 06 09 62 01 ...-e cl-unib...
0040 02 09 74 00 00 01 00 04 ...it...
- ASCII: 29:b0:01:b3:f9:6a:f8:59:71:85:2e:ab:08:00:45:08:j.Y:q:E
00:3a:03:e7:40:00:40:11:b1:c0:c8:a5:01:af:c8:a8:@0:
01:fe:8d:59:08:35:08:26:ab:d2:88:b7:01:00:00:01:Y-5-4-
00:00:00:00:00:00:00:03:65:6f:05:75:06:09:62:01:-e:cl-unib...
02:09:74:00:00:01:00:04:it...

Wireshark Menu

- **File** – Open, merge, export and print capture files
- **Edit** – Search packets, mark them, preferences
- **View** – Coloring packets and view options
- **Go** – Through this menu it is possible to go to a specific packet
- **Capture** – To start capturing and edit capture filters
- **Analyze** – Filtering packets, dissecting protocols
- **Statistics** – To generate and display statistics
- **Telephony** – Telephony related statistics
- **Wireless** – To show wireless related statistics
- **Tools** – Various tools available in wireshark
- **Help** – Help, manual pages



Wireshark Toolbar



- Start
 - Stop
 - Restart
 - Options
 - Open
 - Save
 - Close
-
- Reload
 - Find
 - Go to packets
 - Auto scroll
 - Colorize
 - Zoom options
 - Resize Columns

Wireshark Filtering Toolbar



- Bookmarks
- Filter Input
- Clear
- Apply

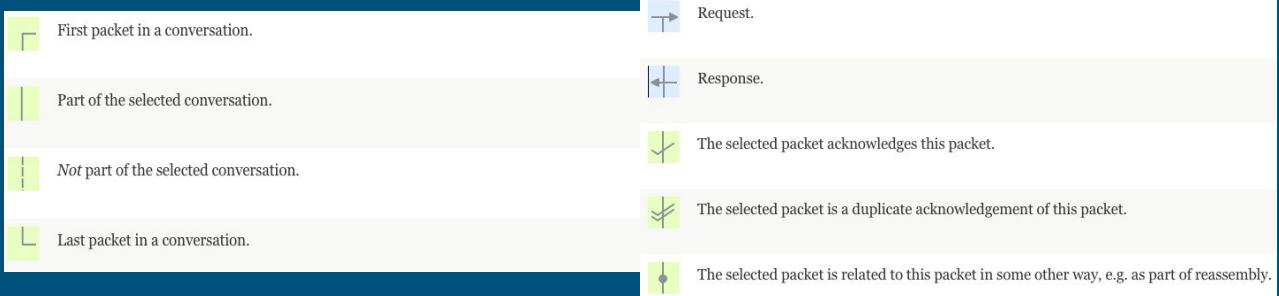
It is probably one of the most powerful tools of wireshark: we'll see how many packets are generated even in low populated networks in short time

- Filtering is essential

The Packet List Panel

No.	Time	Source	Destination	Protocol	Length	Info
19	1.154399103	192.168.1.175	51.124.58.146	TCP	78	38848 → 443 [ACK] Seq=1 Ack=33 Win=342 Len=0 TStamp=855823 TSectr=1018636982 SLE=32 SRE=33
20	2.910263971	2001:b07:6465:2d71::	2001:67c:156b:8003::	NTP	110	NTP Version 4, client
21	2.944619689	2001:67c:156b:8003::	2001:b07:6465:2d71::	NTP	110	NTP Version 4, server
22	3.382992870	149.154.167.91	192.168.1.175	SSL	171	Continuation Data
23	3.424141025	192.168.1.175	149.154.167.91	TCP	66	51140 → 443 [ACK] Seq=1 Ack=106 Win=237 Len=0 TStamp=856391 TSectr=950955734
24	4.164229845	2001:b07:6465:2d71::	2a00:1450:4002:809::	TCP	86	44018 → 443 [ACK] Seq=1 Ack=1 Win=249 Len=0 TStamp=856576 TSectr=3451740999
25	4.164258148	2001:b07:6465:2d71::	2a00:1450:4002:807::	TCP	86	46576 → 443 [ACK] Seq=1 Ack=1 Win=258 Len=0 TStamp=856576 TSectr=429346995
26	4.164263273	2001:b07:6465:2d71::	2a00:1450:4002:807::	TCP	86	46578 → 443 [ACK] Seq=1 Ack=1 Win=334 Len=0 TStamp=856576 TSectr=2026388062
27	4.164269097	2001:b07:6465:2d71::	2a00:1450:4002:805::	TCP	86	51432 → 443 [ACK] Seq=1 Ack=1 Win=336 Len=0 TStamp=856576 TSectr=3418027130
28	4.178266651	2a00:1450:4002:807::	2001:b07:6465:2d71::	TCP	86	[TCP ACKed unseen segment] 443 → 46576 [ACK] Seq=1 Ack=2 Win=266 Len=0 TStamp=4293991962 TSectr=833785
29	4.1784099630	2a00:1450:4002:809::	2001:b07:6465:2d71::	TCP	86	[TCP ACKed unseen segment] 443 → 44018 [ACK] Seq=1 Ack=2 Win=266 Len=0 TStamp=3451786057 TSectr=833777
30	4.178425011	2a00:1450:4002:807::	2001:b07:6465:2d71::	TCP	86	[TCP ACKed unseen segment] 443 → 46578 [ACK] Seq=1 Ack=2 Win=289 Len=0 TStamp=2026433120 TSectr=833840
31	4.178476429	2a00:1450:4002:805::	2001:b07:6465:2d71::	TCP	86	[TCP ACKed unseen segment] 443 → 51432 [ACK] Seq=1 Ack=2 Win=270 Len=0 TStamp=3418073916 TSectr=833622
32	5.557765126	192.168.1.175	192.168.1.254	DNS	89	Standard query 0xc984 A d1-debug.dropbox.com
33	5.557833831	192.168.1.175	192.168.1.254	DNS	89	Standard query 0x773e AAAA d1-debug.dropbox.com
34	5.589383760	192.168.1.254	192.168.1.175	DNS	468	Standard query response 0xc984 A d1-debug.dropbox.com CNAME edge-block-debug-env.dropbox-dns.com A 162.125.69.17 NS dns4.p06.nsone.net NS dns3.p06.nsone.net NS dns1.p06.nsone.net NS...
35	5.575927412	192.168.1.254	192.168.1.175	DNS	420	Standard query response 0x773e AAAA d1-debug.dropbox.com CNAME edge-block-debug-env.dropbox-dns.com AAA 2620:190:6025:17::a7d:4511 NS dns1.p06.nsone.net NS dns4.p06.nsone.net NS...
36	5.576628959	2001:b07:6465:2d71::	2620:100:6025:17::a..	TCP	94	59800 → 443 [SYN] Seq=0 Win=28400 Len=0 MSS=1420 SACK_PERM=1 TStamp=856922 TSectr=0 WS=128
37	5.589399688	2620:100:6025:17::a..	2001:b07:6465:2d71::	TCP	94	443 → 59800 [SYN, ACK] Seq=0 Ack=1 Win=27760 Len=0 MSS=1220 SACK_PERM=0 TStamp=2210307151 TSectr=856929 WS=512
38	5.589363856	2001:b07:6465:2d71::	2620:100:6025:17::a..	TCP	86	59800 → 443 [ACK] Seq=1 Ack=1 Win=28416 Len=0 TStamp=856932 TSectr=2210307151
39	5.589559299	2001:b07:6465:2d71::	2620:100:6025:17::a..	TLSv1.2	603	Client Hello
40	5.603173907	2620:100:6025:17::a..	2001:b07:6465:2d71::	TCP	86	443 → 59800 [ACK] Seq=1 Ack=518 Win=29184 Len=0 TStamp=2210307163 TSectr=856932
41	5.603461703	2620:100:6025:17::a..	2001:b07:6465:2d71::	TLSv1.2	238	Server Hello, Change Cipher Spec, Encrypted Handshake Message
42	5.603566557	2620:100:6025:17::a..	2620:100:6025:17::a..	TCP	86	59800 → 443 [ACK] Seq=158 Ack=153 Win=29568 Len=0 TStamp=856935 TSectr=2210307164
43	5.603824496	2001:b07:6465:2d71::	2620:100:6025:17::a..	TLSv1.2	137	Change Cipher Spec, Encrypted Handshake Message
44	5.604110952	2001:b07:6465:2d71::	2620:100:6025:17::a..	TLSv1.2	1294	Application Data, Application Data, Application Data

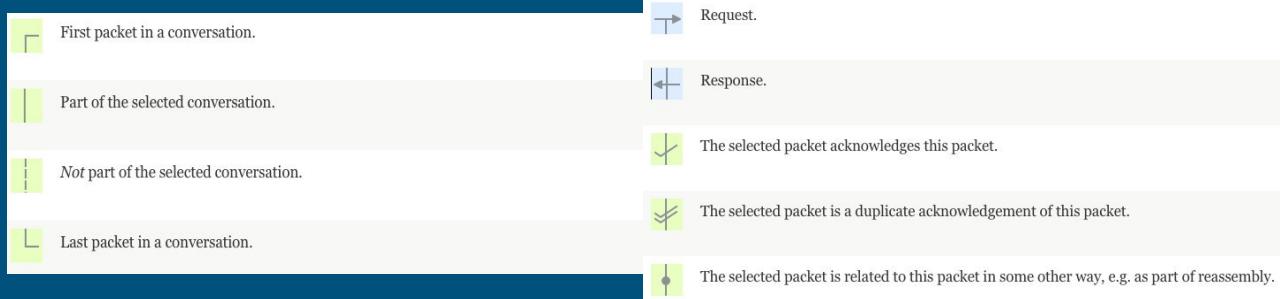
- One packet per line
- If selected, info about the flow
- Source, destination, protocol, etc...



The Packet List Panel

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	2a00:1450:4002:809:...	2001:b07:6465:2d71:...	TLSv1.2	461	Application Data
2	0.000141018	2001:b07:6465:2d71:...	2a00:1450:4002:809:...	TCP	86	57286 - 443 [ACK] Seq=1 Ack=376 Win=249 Len=0 TSval=855535 TSecr=2394000319
3	0.000257517	2a00:1450:4002:809:...	2001:b07:6465:2d71:...	TLSv1.2	191	Application Data, Application Data
4	0.000275655	2a00:1450:4002:809:...	2001:b07:6465:2d71:...	TLSv1.2	125	Application Data
5	0.000373928	2001:b07:6465:2d71:...	2a00:1450:4002:809:...	TCP	86	57286 - 443 [ACK] Seq=1 Ack=520 Win=249 Len=0 TSval=855535 TSecr=2394000361
6	0.001496178	2001:b07:6465:2d71:...	2a00:1450:4002:809:...	TLSv1.2	125	Application Data
7	0.018739019	2a00:1450:4002:809:...	2001:b07:6465:2d71:...	TCP	86	443 - 57286 [ACK] Seq=520 Ack=40 Win=266 Len=0 TSval=2394000380 TSecr=855535
8	0.028146122	2001:b07:6465:2d71:...	2a00:1450:4013:c01:...	TCP	86	43360 - 443 [ACK] Seq=1 Ack=1 Win=1419 Len=0 TSval=855542 TSecr=1957605142
9	0.516079190	2001:b07:6465:2d71:...	2a00:1450:4013:c06:...	TLSv1.2	443	Application Data
10	0.516126491	2001:b07:6465:2d71:...	2a00:1450:4013:c06:...	TCP	86	54352 - 443 [ACK] Seq=1 Ack=358 Win=260 Len=0 TSval=855664 TSecr=3231954553
11	0.516145642	2001:b07:6465:2d71:...	2001:b07:6465:2d71:...	TLSv1.2	139	Application Data
12	0.516151068	2001:b07:6465:2d71:...	2a00:1450:4013:c06:...	TCP	86	54352 - 443 [ACK] Seq=1 Ack=411 Win=260 Len=0 TSval=855664 TSecr=3231954553
13	0.527617797	2a00:1450:4013:c06:...	2001:b07:6465:2d71:...	TLSv1.2	139	[TCP Spurious Retransmission], Application Data
14	0.527654385	2001:b07:6465:2d71:...	2a00:1450:4013:c06:...	TCP	98	[TCP Dup ACK 12#1] 54352 - 443 [ACK] Seq=1 Ack=411 Win=260 Len=0 TSval=855666 TSecr=3231954626 SLE=358 SRE=411
15	1.130009354	51.124.58.146	192.168.1.175	TLSv1.2	97	Encrypted Alert
16	1.1300655501	192.168.1.175	51.124.58.146	TCP	66	38848 - 443 [ACK] Seq=1 Ack=32 Win=342 Len=0 TSval=855817 TSecr=1018636887
17	1.130185487	51.124.58.146	192.168.1.175	TCP	66	443 - 38848 [FIN, ACK] Seq=32 Ack=1 Win=38 Len=0 TSval=1018636887 TSecr=837050
18	1.154254414	51.124.58.146	192.168.1.175	TCP	66	[TCP Retransmission] 443 - 38848 [FIN, ACK] Seq=32 Ack=1 Win=38 Len=0 TSval=1018636982 TSecr=837050
19	1.154309103	192.168.1.175	51.124.58.146	TCP	78	38848 - 443 [ACK] Seq=1 Ack=33 Win=342 Len=0 TSval=855823 TSecr=1018636982 SLE=32 SRE=33
20	2.910263971	2001:b07:6465:2d71:...	2001:67c:1560:8003:...	NTP	110	NTP Version 4, client
21	2.944619689	2001:67c:1560:8003:...	2001:b07:6465:2d71:...	NTP	110	NTP Version 4, server
22	3.382902870	149.154.167.91	192.168.1.175	SSL	171	Continuation Data
23	3.4241141025	192.168.1.175	149.154.167.91	TCP	66	51104 - 443 [ACK] Seq=1 Ack=106 Win=237 Len=0 TSval=856391 TSecr=950955734
24	4.164229845	2001:b07:6465:2d71:...	2a00:1450:4002:809:...	TCP	86	44018 - 443 [ACK] Seq=1 Ack=1 Win=249 Len=0 TSval=856576 TSecr=3451740999
25	4.164258148	2001:b07:6465:2d71:...	2a00:1450:4002:807:...	TCP	86	46576 - 443 [ACK] Seq=1 Ack=1 Win=258 Len=0 TSval=856576 TSecr=4293946995
26	4.164263273	2001:b07:6465:2d71:...	2a00:1450:4002:807:...	TCP	86	46578 - 443 [ACK] Seq=1 Ack=1 Win=334 Len=0 TSval=856576 TSecr=2026388062

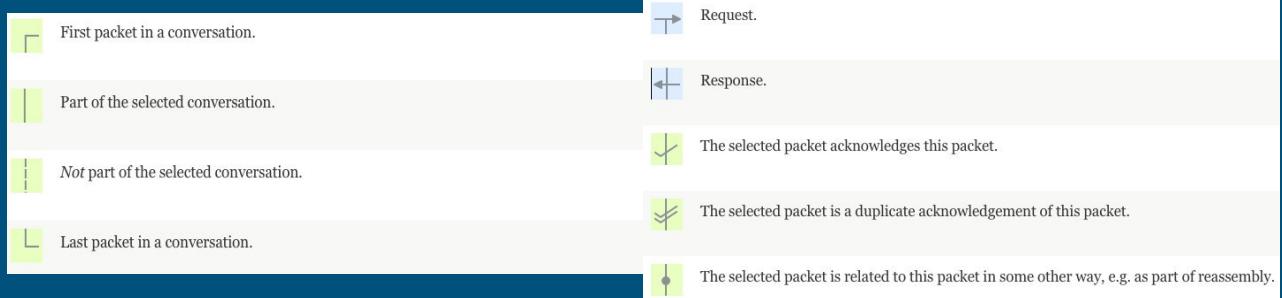
- One packet per line
- If selected, info about the flow
- Source, destination, protocol, etc...



The Packet List Panel

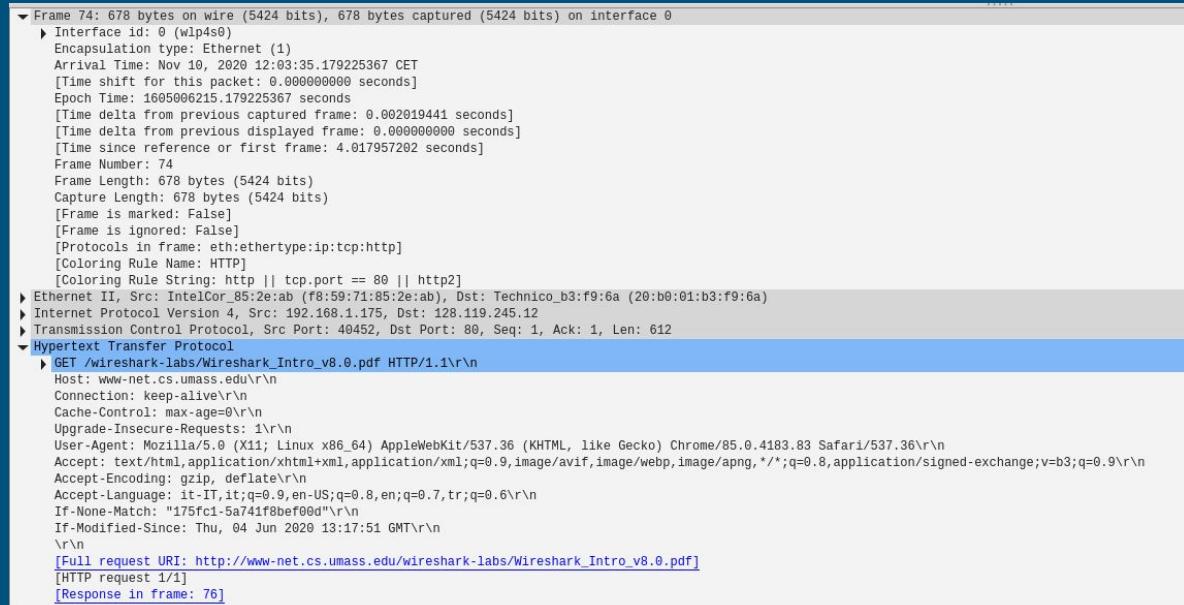
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	2a00:1450:4002:809:...	2001:b07:6465:2d71:...	TLSv1.2	461	Application Data
2	0.000141018	2001:b07:6465:2d71:...	2a00:1450:4002:809:...	TCP	86	57286 - 443 [ACK] Seq=1 Ack=376 Win=249 Len=0 TSval=855535 TSecr=2394000319
3	0.000257517	2a00:1450:4002:809:...	2001:b07:6465:2d71:...	TLSv1.2	191	Application Data, Application Data
4	0.000275655	2a00:1450:4002:809:...	2001:b07:6465:2d71:...	TLSv1.2	125	Application Data
5	0.000373928	2001:b07:6465:2d71:...	2a00:1450:4002:809:...	TCP	86	57286 - 443 [ACK] Seq=1 Ack=520 Win=249 Len=0 TSval=855535 TSecr=2394000319
6	0.001496178	2001:b07:6465:2d71:...	2a00:1450:4002:809:...	TLSv1.2	125	Application Data
7	0.018739018	2a00:1450:4002:809:...	2001:b07:6465:2d71:...	TCP	86	443 - 57286 [ACK] Seq=520 Ack=40 Win=266 Len=0 TSval=2394000380 TSecr=855535
8	0.028146122	2001:b07:6465:2d71:...	2a00:1450:4013:c01:...	TCP	86	43360 - 443 [ACK] Seq=1 Ack=1 Win=1419 Len=0 TSval=855542 TSecr=1957605142
9	0.516079190	2a00:1450:4013:c06:...	2001:b07:6465:2d71:...	TLSv1.2	443	Application Data
10	0.516126401	2001:b07:6465:2d71:...	2a00:1450:4013:c06:...	TCP	86	54352 - 443 [ACK] Seq=1 Ack=358 Win=260 Len=0 TSval=855664 TSecr=3231954553
11	0.516145642	2a00:1450:4013:c06:...	2001:b07:6465:2d71:...	TLSv1.2	139	Application Data
12	0.516151068	2001:b07:6465:2d71:...	2a00:1450:4013:c06:...	TCP	86	54352 - 443 [ACK] Seq=1 Ack=411 Win=260 Len=0 TSval=855664 TSecr=3231954553
13	0.527617797	2a00:1450:4013:c06:...	2001:b07:6465:2d71:...	TLSv1.2	139	[TCP Spurious Retransmission], Application Data
14	0.527654388	2001:b07:6465:2d71:...	2a00:1450:4013:c06:...	TCP	98	[TCP Dup ACK 12#1] 54352 - 443 [ACK] Seq=1 Ack=411 Win=260 Len=0 TSval=855666 TSecr=3231954626 SLE=358 SRE=411
15	1.139009354	51.124.58.146	192.168.1.175	TLSv1.2	97	Encrypted Alert
16	1.1390055501	192.168.1.175	51.124.58.146	TCP	66	38848 - 443 [ACK] Seq=1 Ack=32 Win=342 Len=0 TSval=855817 TSecr=1018636887
17	1.139185487	51.124.58.146	192.168.1.175	TCP	66	443 - 38848 [FIN, ACK] Seq=32 Ack=1 Win=38 Len=0 TSval=1018636887 TSecr=837050
18	1.154254414	51.124.58.146	192.168.1.175	TCP	66	[TCP Retransmission] 443 - 38848 [FIN, ACK] Seq=32 Ack=1 Win=38 Len=0 TSval=1018636982 TSecr=837050
19	1.1543699193	192.168.1.175	51.124.58.146	TCP	78	38848 - 443 [ACK] Seq=1 Ack=33 Win=342 Len=0 TSval=856391 TSecr=950955734
20	2.910263971	2001:b07:6465:2d71:...	2001:67c:1560:8003:...	NTP	110	NTP Version 4, client
21	2.944619681	2001:67c:1560:8003:...	2001:b07:6465:2d71:...	NTP	110	NTP Version 4, server
22	3.382992870	149.167.91	192.168.1.175	SSL	171	Continuation Data
23	3.4241141025	192.168.1.175	149.154.167.91	TCP	66	51140 - 443 [ACK] Seq=1 Ack=106 Win=237 Len=0 TSval=856391 TSecr=950955734
24	4.16429845	2001:b07:6465:2d71:...	2a00:1450:4002:809:...	TCP	86	44018 - 443 [ACK] Seq=1 Ack=1 Win=249 Len=0 TSval=856576 TSecr=3451740999
25	4.164258148	2001:b07:6465:2d71:...	2a00:1450:4002:807:...	TCP	86	46576 - 443 [ACK] Seq=1 Ack=1 Win=258 Len=0 TSval=856576 TSecr=4293946905
26	4.164263273	2001:b07:6465:2d71:...	2a00:1450:4002:807:...	TCP	86	46578 - 443 [ACK] Seq=1 Ack=1 Win=334 Len=0 TSval=856576 TSecr=2026388062

- One packet per line
- If selected, info about the flow
- Source, destination, protocol, etc...



The Packet Details Panel

- It shows the details for a specific selected packet
- It can display additional information enclosed in brackets
- It also shows links if wireshark detects a link with another packet



A screenshot of the Wireshark packet details panel. The selected packet is a GET request for 'Wireshark_Intro_v8.0.pdf' from 'www-net.cs.umass.edu'. The details pane displays various metadata and the raw HTTP request message. Brackets are used to highlight specific fields like 'Frame Number', 'Frame Length', and 'Capture Length'. A blue bar highlights the selected packet's details.

```
Frame 74: 678 bytes on wire (5424 bits), 678 bytes captured (5424 bits) on interface 0
  ▶ Interface id: 0 (wlp4s0)
    Encapsulation type: Ethernet (1)
    Arrival Time: Nov 10, 2020 12:03:35.179225367 CET
      [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1605006215.179225367 seconds
      [Time delta from previous captured frame: 0.0002019441 seconds]
      [Time delta from previous displayed frame: 0.000000000 seconds]
      [Time since reference or first frame: 4.017957202 seconds]
    Frame Number: 74
    Frame Length: 678 bytes (5424 bits)
    Capture Length: 678 bytes (5424 bits)
      [Frame is marked: False]
      [Frame is ignored: False]
      [Protocols in frame: eth:ethertype:ip:tcp:http]
      [Coloring Rule Name: HTTP]
      [Coloring Rule String: http || tcp.port == 80 || http2]
  ▶ Ethernet II, Src: IntelCor_85:2e:ab (f8:59:71:85:2e:ab), Dst: Technico_b3:f9:6a (20:b0:01:b3:f9:6a)
  ▶ Internet Protocol Version 4, Src: 192.168.1.175, Dst: 128.119.245.12
  ▶ Transmission Control Protocol, Src Port: 40452, Dst Port: 80, Seq: 1, Ack: 1, Len: 612
  ▶ Hypertext Transfer Protocol
    ▶ GET /wireshark-labs/Wireshark_Intro_v8.0.pdf HTTP/1.1\r\n
      Host: www-net.cs.umass.edu\r\n
      Connection: keep-alive\r\n
      Cache-Control: max-age=0\r\n
      Upgrade-Insecure-Requests: 1\r\n
      User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/85.0.4183.83 Safari/537.36\r\n
      Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9\r\n
      Accept-Encoding: gzip, deflate\r\n
      Accept-Language: it-IT,it;q=0.9,en-US;q=0.8,en;q=0.7,tr;q=0.6\r\n
      If-None-Match: "175fc1-5a741f8beff00d"\r\n
      If-Modified-Since: Thu, 04 Jun 2020 13:17:51 GMT\r\n
    \r\n
    [Full request URI: http://www-net.cs.umass.edu/wireshark-labs/Wireshark_Intro_v8.0.pdf]
    [HTTP request 1/1]
    [Response in frame: 76]
```

The packet Bytes Panel

```
0000  20 b0 01 b3 f9 6a f8 59  71 85 2e ab 08 00 45 00  ...j.Y q...E
0010  02 98 14 59 40 00 40 06  ec 2b c0 a8 01 af 80 77  ...Y@@.+.+....w
0020  f5 0c 9e 04 00 50 d1 66  f4 9e fa c4 72 cf 80 18  ....P.f ....r...
0030  00 e5 32 8e 00 00 01 01  08 0a 00 1d 0a 99 32 61  ..2.....2a
0040  d8 f7 47 45 54 20 2f 77  69 72 65 73 68 61 72 6b  ..GET /w ireshark
0050  2d 6c 61 62 73 2f 57 69  72 65 73 68 61 72 6b 5f  -labs/Wireshark_
0060  49 6e 74 72 6f 5f 76 38  2e 30 2e 70 64 66 20 48  Intro_v8 .0.pdf H
0070  54 54 50 2f 31 2e 31 0d  0a 48 6f 73 74 3a 20 77  TTP/1.1 Host: w
0080  77 77 2d 6e 65 74 2e 63  73 2e 75 6d 61 73 73 2e  www-net.c.s.umass.
0090  65 64 75 0d 0a 43 6f 6e  6e 65 63 74 69 6f 6e 3a  edu Connection:
00a0  20 6b 65 65 70 2d 61 6c  69 76 65 0d 0a 43 61 63  keep-alive Cache-Control
00b0  68 65 2d 43 6f 6e 74 72  6f 6c 3a 20 6d 61 78 2d  max-age=0 Upgrade-Insecure-Requests
00c0  61 67 65 3d 30 0d 0a 55  70 67 72 61 64 65 2d 49  :1 Use-Agent:
00d0  6e 73 65 63 75 72 65 2d  52 65 71 75 65 73 74 73
00e0  3a 20 31 0d 0a 55 73 65  72 2d 41 67 65 6e 74 3a
```

Shows the dump of the packet

- Shown in hexadecimal or binary

More than one page may available in case wireshark reassembled more than one packet together

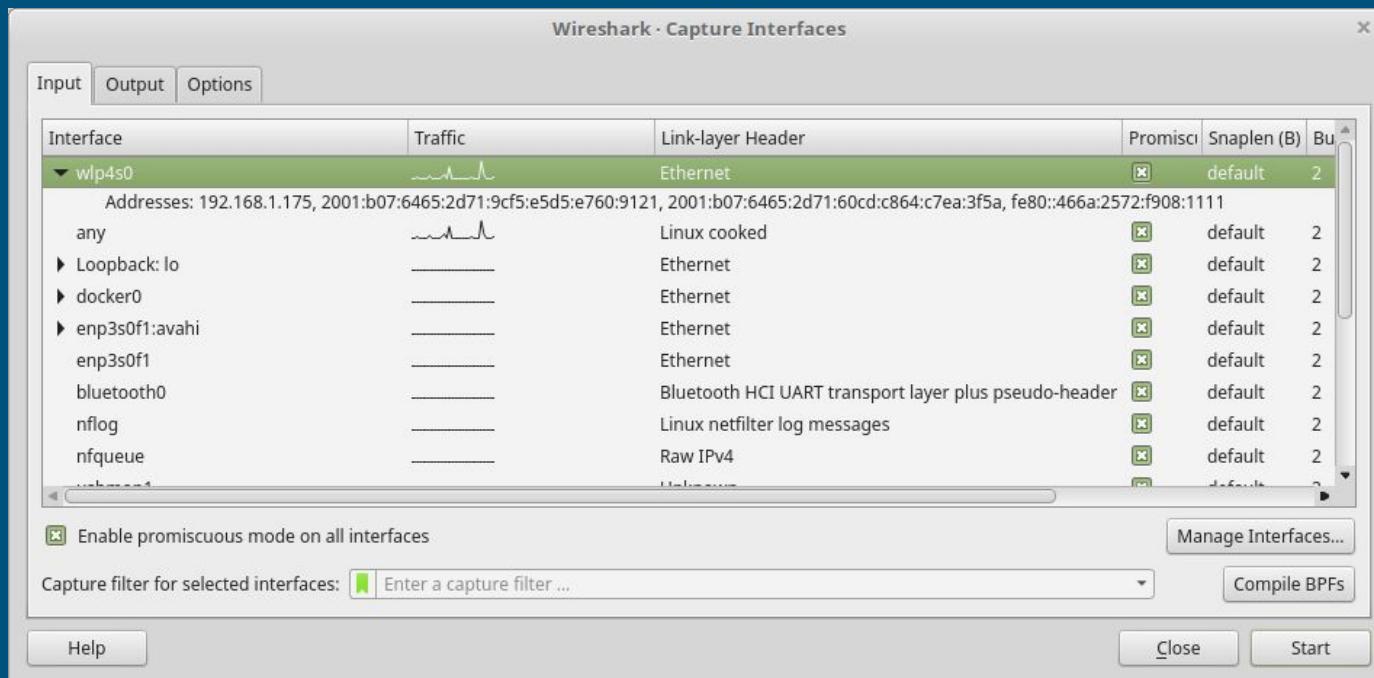
Capturing Live Network Data

Capturing live network data is one of the core components of wireshark

- Can capture from different network interfaces
- Triggers to stop capturing data (elapsed time, number of packets..)
- Live show of packet details
- Live filtering of packets
- Save packets
- Can simultaneously capture from different network interfaces

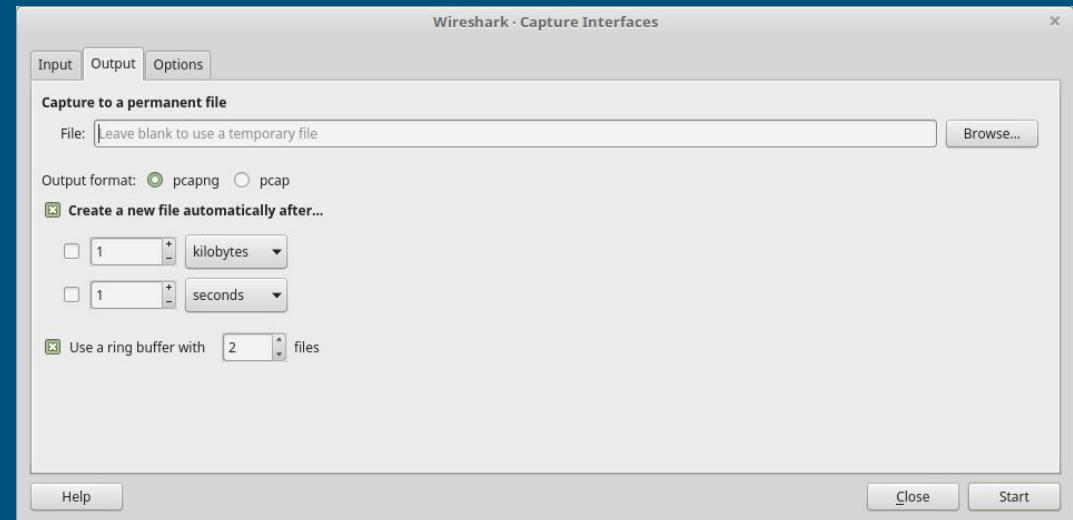
Start a new capture

Capture > Options...



Capture Options

It is possible to save a capture to a file (or multiple files). Consider using this feature if you plan to work with a heavily congested network or if you plan to perform a long-term capture



1. No filename
 - o (wiresharkXXXX)→ Temporary file
2. Filename
 - o (overwritten every time)→ Fixed file
3. Filename + auto
 - o (foo_00001_20100205110102.cap, _00001_20100205110337.cap, ...)→ More files, every time a new one
- 4.Filename + auto + ring buffer
 - o (foo_00001_20100205110102.cap, _00001_20100205110337.cap, ...)→ Files get replaced from the beginning after buffer is full

Let's start with the practice

1. Use your own PC and let's take a look at what we see

Let's start with the practice

1. Use your own PC and let's take a look at what we see

Pretty messy huh? There's so much traffic going on without us being fully aware... even when the machine is idle. Let's see if we can cut this to a minimum.

2. Use a Virtual Machine (several network options, let's try with NAT).

Let's start with the practice

1. Use your own PC and let's take a look at what we see

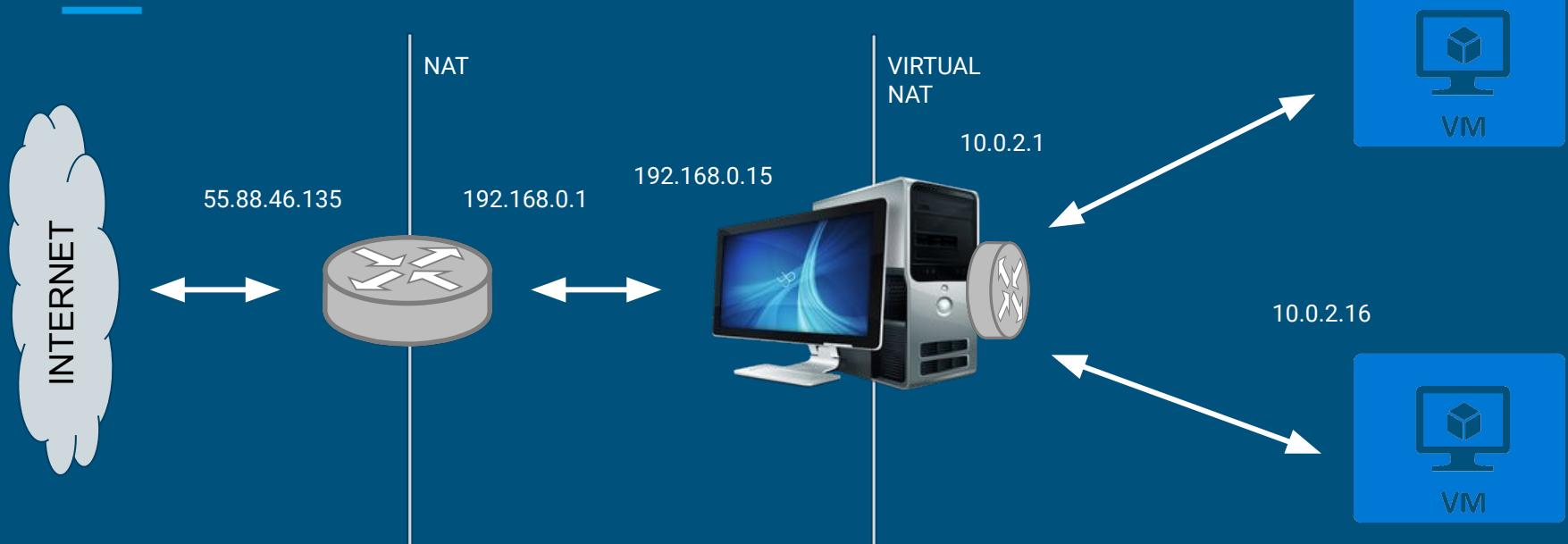
Pretty messy huh? There's so much traffic going on without us being fully aware... even when the machine is idle. Let's see if we can cut this to a minimum.

2. Use a Virtual Machine (several network options, let's try with NAT).

That's good, however it is more difficult to see what the other hosts are doing, because our VM is cut out from the world.

3. Use several virtual machines connected to the same NAT Network!

Little off topic



You can take a look on how to do it quite easily with Virtual Box, check here:
<https://www.dedoimedo.com/computers/virtualbox-nat-networks.html>

Wireshark Filters

Wireshark empowers the user with powerful filtering expressions:

- Can filter based on the content of a packet
 - Can filter based on the IP of a packet
 - Can filter based on the protocol of a packet
 - Many others...
 - It supports comparison and boolean operators
-
- **Capture Filters:** set before capturing (much like IPTABLES)
 - **Display Filters:** set while capturing (only a “visualization filter”)

Capture Filters

Wireshark uses the libpcap filter language. The general syntax is as follows:

[not] primitive [and|or [not] primitive...]

Example: tcp port 23 and host 10.0.0.5

Capture filters are different from Display filters (they use a different syntax)

Primitives:

- [src/dst] host <host>
- ether [src/dst] host <host>
- gateway host <host>
- [src|dst] net <net> [{mask <mask>}|{len <len>}]

- [tcp|udp] [src|dst] port <port>
- less|greater <length>
- ip|ether proto <protocol>
- ether|ip broadcast|multicast
- <expr> relop <expr>

More at

https://www.wireshark.org/docs/wsug_html_chunked/ChCapCaptureFilterSection.html

Display Filters - Comparison Operators

Operator	Description	Example
<code>==</code>	Equal	<code>ip.src == 192.168.1.1</code>
<code>!=</code>	Not equal	<code>ip.src != 192.168.1.1</code>
<code>> < >= <=</code>	Greater/Less than	<code>frame.len > 10</code>
<code>contains</code>	Field contains a value	<code>sip.To contains "a1762"</code>
<code>matches</code>	Field matches a regexp	<code>http.host matches "acme\.(org com net)"</code>
<code>&</code>	Bitwise AND	<code>tcp.flags & 0x02</code>

Display Filters - Combination Operators

Operator	Description	Example
and	Logical AND	ip.src == 192.168.1.1 and tcp.flags.fin
or	Logical OR	ip.src == 192.168.1.1 or tcp.flags.fin
not	Logical NOT	not llc
xor	Logical XOR	ip.dst == 10.0.6.29 xor ip.src == 10.0.6.29
[...]	Slice Operator (see next)	eth.src[0:3] == 00:00:83
in	Membership (see next)	tcp.port in {80 443 8080}

Display Filters - Slice Operator

Used to select subsequences of a sequence

Simply put brackets after a label:

- eth.src[0:3] == 00:00:83
- eth.src[1-2] == 00:83
- eth.src[:4] != 00:83:45:21
- eth.src[4:] != 00:83
- eth.src[4] != 00
- eth.src[0:3,1-2,:4,4:,2] == 00:00:83:00:83:00:00:83:00:20:20:83

: expects a length, - expects the end

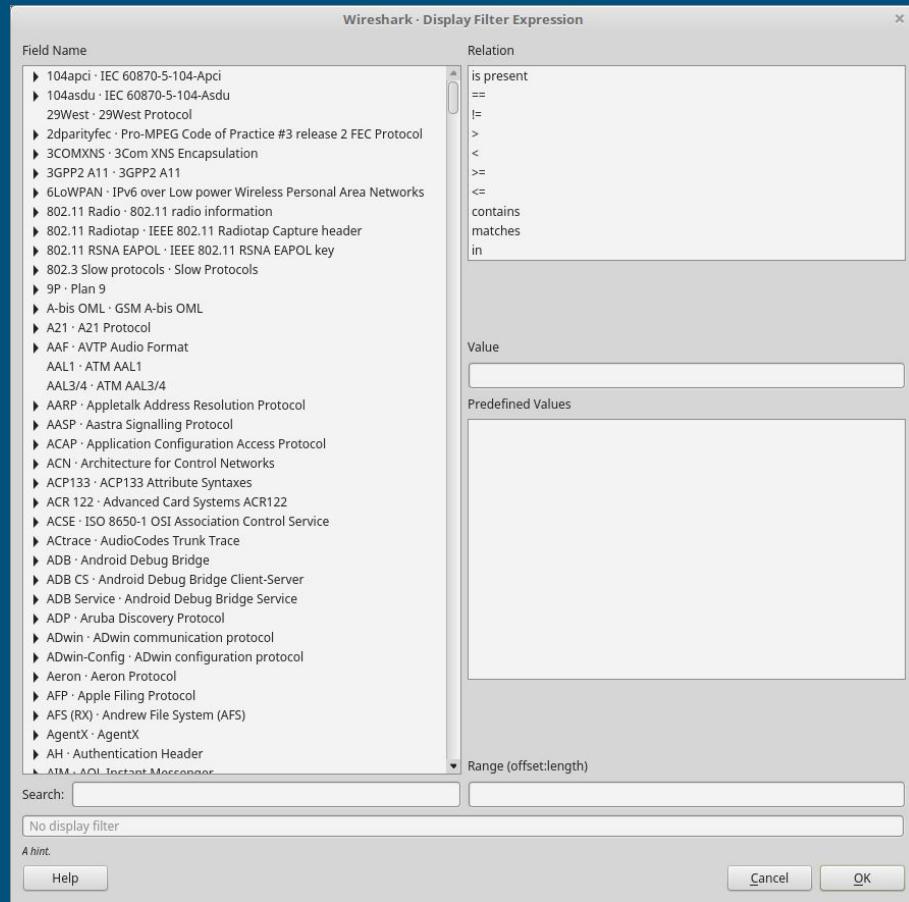
Display Filters - Membership Operator

Used to test a field against a set of values, simply use `in` after a label and put the set inside {}:

- `tcp.port in {80 443 8080}`
 - This is equal to `tcp.port == 80` or `tcp.port == 443` or `tcp.port == 8080`
- `tcp.port in {443 4430..4434}`
 - This is equal to `tcp.port == 443` or (`tcp.port >= 4430` and `tcp.port <= 4434`)
- `http.request.method in {"HEAD" "GET"}`
- `ip.addr in {10.0.0.5 .. 10.0.0.9 192.168.1.1 .. 192.168.1.9}`
- `frame.time_delta in {10 .. 10.5}`
- Wireshark also offers simple functions that can be helpful when dealing with packet content: `upper/lower` and `len/count`

More on Display Filters

- Once you get used to Wireshark, you will use rarely the dialog box
- At the beginning, it is an valuable tool to learn about Wireshark display strings
- It has a search function which makes easier to navigate through all the possible fields
- Also possible to define relations between fields and labels
- Finally, it is also possible to save filters for later use



Example on Capture Filters

host 192.168.1.200 - Capture only packets coming from or going to host 192.168.1.200

ether host 00:00:5e:00:53:00 - Get all packets with source or destination MAC address equal to 00:00:5e:00:53:00

host google.it - Get all the packets coming from or going to host www.google.it

not broadcast and not multicast - Do not capture packets which are either broadcast or multicast

Example on Display Filters

`tcp.port == 80` - Display all packets with the port equals to 80 (default HTTP)

`tcp.port == 80 or tcp.port == 443` - Display all packets with the port equals to 80 (default HTTP) or 443 (default HTTPS). Same as `tcp.port in { 80 443 }`

`tcp.dstport == 80 and (tcp.srcport > 60000 and
tcp.srcport < 64000)` - Display all packets directed to port 80 coming from a port within 60000 and 64000

Reverse Examples

You are seeing an unusual http traffic from IP 192.168.1.200, as it makes a lot of requests to www.iamnotasafesite.danger. You are afraid that all its subnetwork (255.255.255.0) may be compromised. You want to identify all the hosts that are possibly compromised, what do you write?

Reverse Examples

You are seeing an unusual http traffic from IP 192.168.1.200, as it makes a lot of requests to www.iamnotasafesite.danger. You are afraid that all its subnetwork (255.255.255.0) may be compromised. You want to identify all the hosts that are possibly compromised, what do you write?

- Capture: `src net 192.168.1.0/24`
- Display: `http.host www.iamnotasafesite.danger`

Reverse Examples

After analysing the previous attack, you discover that there are many sites using the domain .danger that may be harmful for your hosts. You also discover that there is a page, *nowillstealallyourpersonalbelongings.html*, which is the root cause of the problems. Identify all the hosts that visit such page.

Reverse Examples

After analysing the previous attack, you discover that there are many sites using the domain .danger that may be harmful for your hosts. You also discover that there is a page, *nowillstealallyourpersonalbelongings.html*, which is the root cause of the problems. Identify all the hosts that visit such page.

- Capture: `src net 192.168.1.0/24`
- Display: `http.host contains ".danger" and frame contains "nowillstealallyourpersonalbelongings.html"`

Exercise 0

- Open Wireshark
- Start a capture on at least the network interface through which you are connected to the internet
 - you can try through capture filters to only get IPv4 packets
- Go to www.google.com
- Stop the capture
- How many packets you see?
- Are all of them originated by you?
- How to find the packets related to your last internet search?
 - filter “http” packets... do they help out?

Exercise 0 (cont'd)

- Open Wireshark
- Let's test the lower layer stuff
- Try to ping 8.8.8.8
 - Observe the ICMP packet, what's inside at layers 3 and 2...
- Try to ping something in your local network (e.g. your router).
 - Any ARP packets coming along? If not, why? What do they mean?
- Try to force DHCP to give you another address (usually dhclient on linux).
 - How is the negotiation taking place?

Exercise 1

We will carry out this exercise concerning ethernet

- We are at ISO/OSI Level 2
- We need the **ethernet--ethereal-trace-1** from the wireshark-traces.zip file
- We will also investigate ARP related messages

Exercise 1

- Network interfaces have a unique address, called MAC
- Part of it identify the manufacturer, the rest is a progressive counter
- Computer receive messages through their MAC
- Through a protocol (ARP) it is possible to uniquely identify a computer on a network
- ARP builds its table dynamically
- How many mac addresses? With a 48 bit addressing space,
281,474,976,710,656

Exercise 1

Open the **ethernet-ethereal-trace-1** file look into HTTP part

- What is the ethernet address of your computer?
- What is the ethernet address of the destination?
- Is it the MAC address of gaia.cs.umass.edu?
- What device has this as its Ethernet address?
- What is the hexadecimal value for the 2-byte Frame type field for IP?
- How many bytes from the very start of the Ethernet frame does the ASCII “G” in “GET” appear in the Ethernet frame?
- What is the source Ethernet address in the HTTP OK answer?
- Is it the address of gaia.cs.umass.edu?
- What is the destination address in the HTTP OK answer?

Exercise 1 (ans)

Open the **ethernet-ethereal-trace-1** file look into HTTP part

- What is the ethernet address of your computer? 00:d0:59:a9:3d:68
- What is the ethernet address of the destination? 00:06:25:da:af:73
- Is it the address of gaia.cs.umass.edu? No
- What device has this as its Ethernet address? The router (Linksys)
- What is the hexadecimal value for the 2-byte Frame type field for IP? 0x0800
- How many bytes from the very start of the Ethernet frame does the ASCII “G” in “GET” appear in the Ethernet frame? 54 B. (14B Ethernet, 20B IP, 20B TCP)
- What is the source Ethernet address in the HTTP OK answer? 00:06:25:da:af:73
- Is it the address of gaia.cs.umass.edu? No
- What is the destination address in the HTTP OK answer? 00:d0:59:a9:3d:68

Exercise 1

Open the **ethernet-ethereal-trace-1** file look into the ARP part

- What are the source/destination addresses of the ARP request?
- What is the hexadecimal value for the 2-byte Frame type field?
- Does the ARP message contains the sender IP (if yes, what it is?)?
- In the ARP reply, what are the Ethernet and IP addresses of the machine having the Ethernet address whose corresponding IP address is being queried?

Exercise 1 (ans)

Open the **ethernet-ethereal-trace-1** file look into the ARP part

- What are the source/destination addresses of the ARP request? 00:d0:59:a9:3d:68 and ff:ff:ff:ff:ff:ff
- What is the hexadecimal value for the 2-byte Frame type field? 0x0806 (ARP)
- Does the ARP message contains the sender IP (if yes, what it is?)? Yes, 192.168.1.105
- In the ARP reply, what are the Ethernet and IP addresses of the machine having the Ethernet address whose corresponding IP address is being queried? 00:06:25:da:af:73 - 192.168.1.1.

Chapter 8

Security

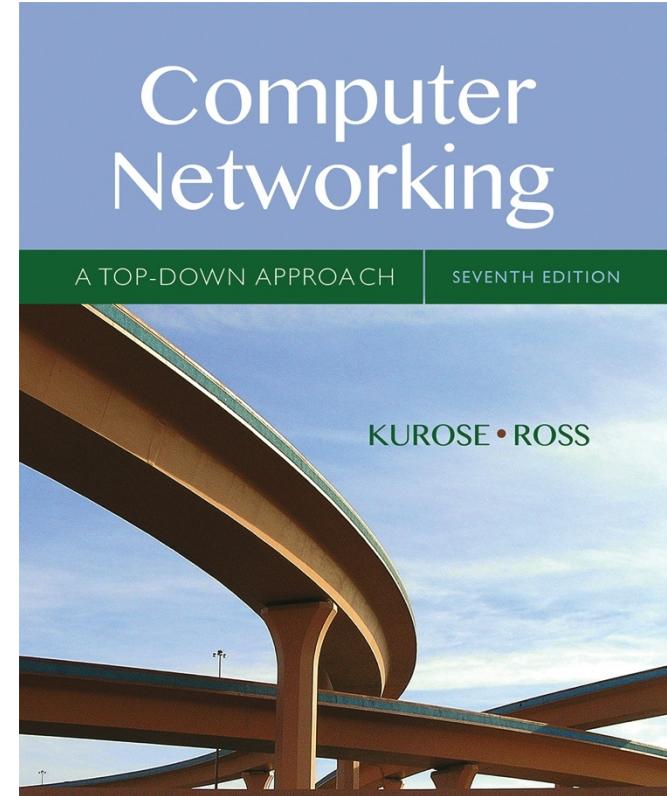
A note on the use of these Powerpoint slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR

© All material copyright 1996-2016
J.F Kurose and K.W. Ross, All Rights Reserved



*Computer
Networking: A Top
Down Approach*

7th edition

Jim Kurose, Keith Ross
Pearson/Addison Wesley
April 2016

Chapter 8: Network Security

Chapter goals:

- understand principles of network security:
 - cryptography and its *many* uses beyond “confidentiality”
 - authentication
 - message integrity
- security in practice:
 - firewalls and intrusion detection systems
 - security in application, transport, network, link layers

Chapter 8 roadmap

8.1 *What is network security?*

8.2 Principles of cryptography

8.3 Message integrity, authentication

8.4 Securing e-mail

8.5 Securing TCP connections: SSL

8.6 Network layer security: IPsec

8.7 Securing wireless LANs

8.8 Operational security: firewalls and IDS

What is network security?

confidentiality: only sender, intended receiver should “understand” message contents

- sender encrypts message
- receiver decrypts message

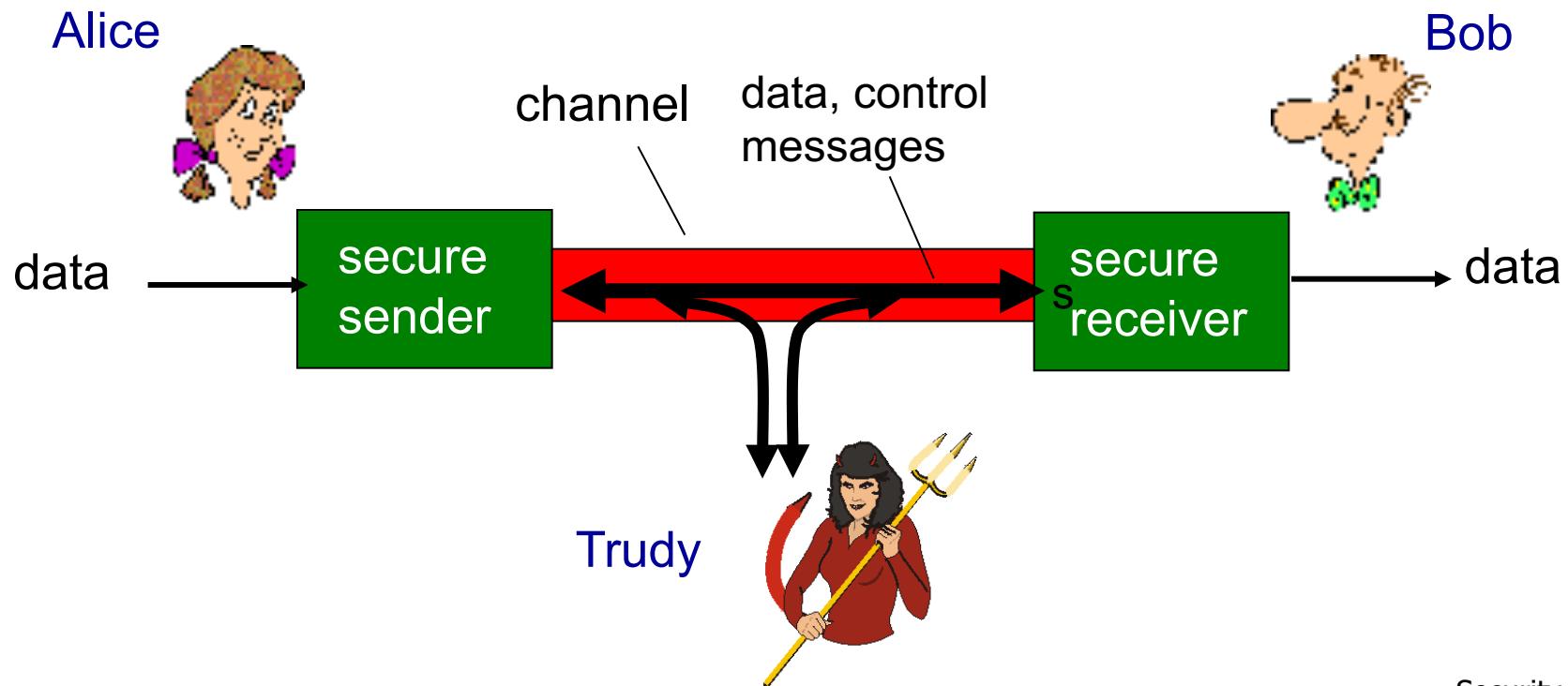
authentication: sender, receiver want to confirm identity of each other

message integrity: sender, receiver want to ensure message not altered (in transit, or afterwards) without detection

access and availability: services must be accessible and available to users

Friends and enemies: Alice, Bob, Trudy

- well-known in network security world
- Bob, Alice (lovers!) want to communicate “securely”
- Trudy (intruder) may intercept, delete, add messages



Who might Bob, Alice be?

- ... well, *real-life* Bobs and Alices!
- Web browser/server for electronic transactions
(e.g., on-line purchases)
- on-line banking client/server
- DNS servers
- routers exchanging routing table updates
- other examples?

There are bad guys (and girls) out there!

Q: What can a “bad guy” do?

A: A lot! See section 1.6

- *eavesdrop*: intercept messages
- actively *insert* messages into connection
- *impersonation*: can fake (spoof) source address in packet (or any field in packet)
- *hijacking*: “take over” ongoing connection by removing sender or receiver, inserting himself in place
- *denial of service*: prevent service from being used by others (e.g., by overloading resources)

Chapter 8 roadmap

8.1 What is network security?

8.2 *Principles of cryptography*

8.3 Message integrity, authentication

8.4 Securing e-mail

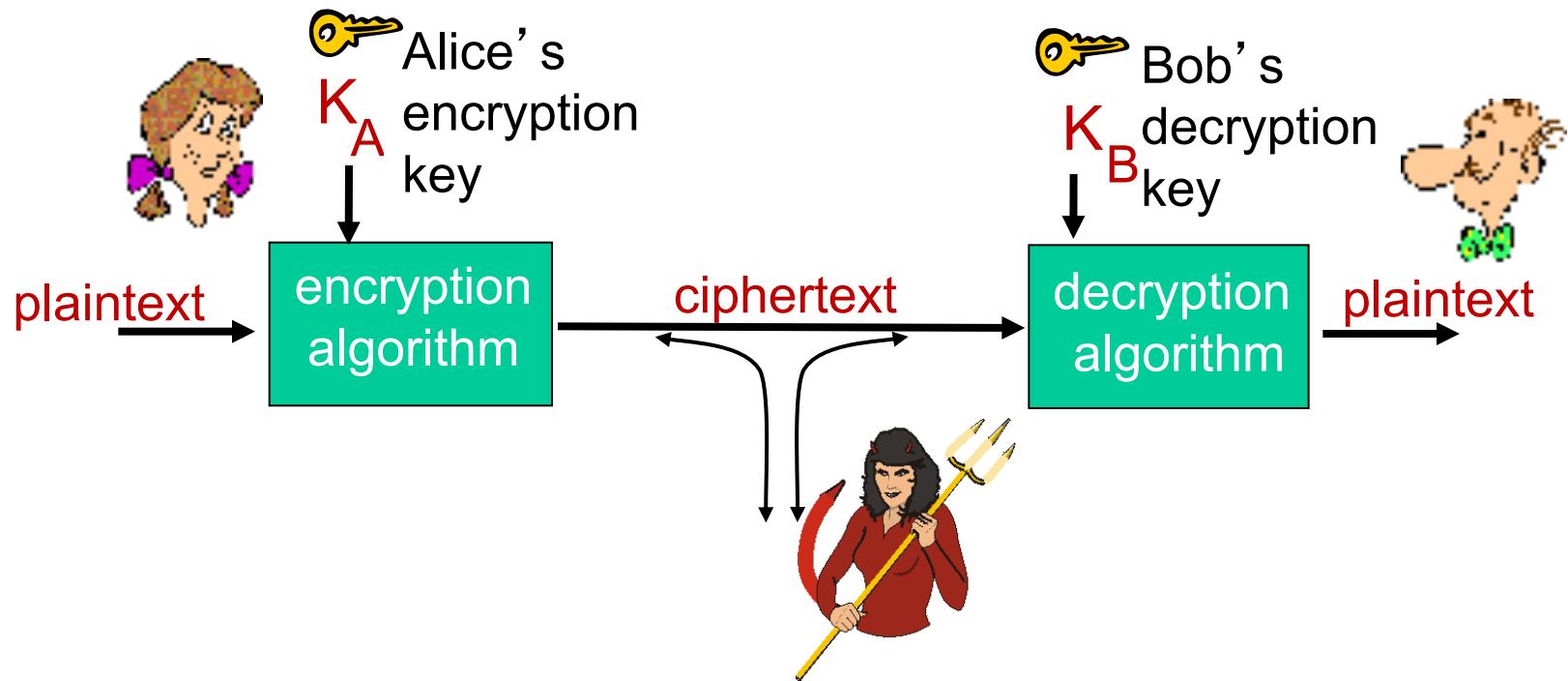
8.5 Securing TCP connections: SSL

8.6 Network layer security: IPsec

8.7 Securing wireless LANs

8.8 Operational security: firewalls and IDS

The language of cryptography



m plaintext message

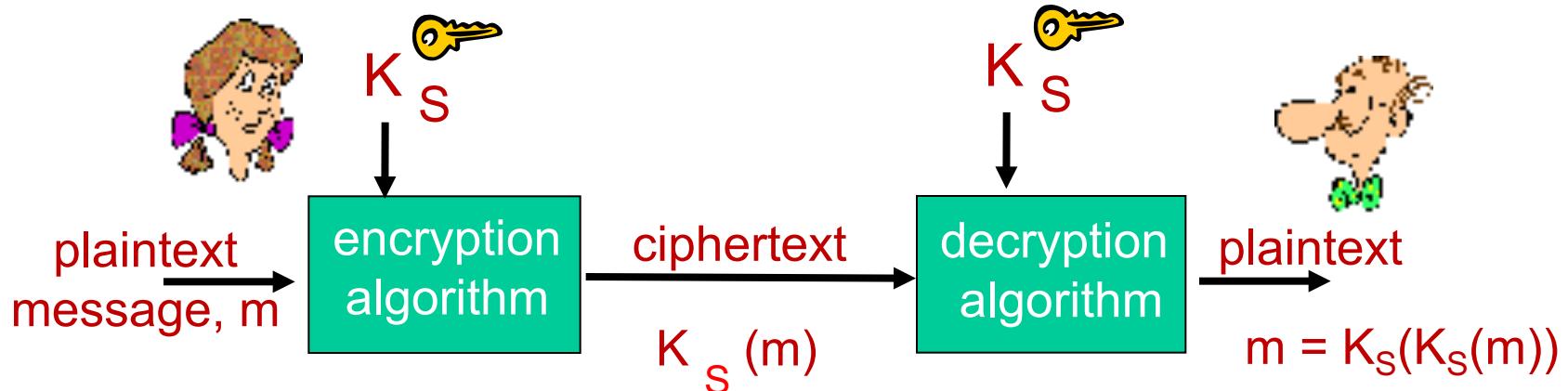
$K_A(m)$ ciphertext, encrypted with key K_A

$m = K_B(K_A(m))$

Breaking an encryption scheme

- **cipher-text only attack:**
Trudy has ciphertext she can analyze
- **two approaches:**
 - brute force: search through all keys
 - statistical analysis
- **known-plaintext attack:**
Trudy has plaintext corresponding to ciphertext
 - e.g., in monoalphabetic cipher, Trudy determines pairings for a,l,i,c,e,b,o,
- **chosen-plaintext attack:**
Trudy can get ciphertext for chosen plaintext

Symmetric key cryptography



symmetric key crypto: Bob and Alice share same (symmetric) key: K_S

- e.g., key is knowing substitution pattern in mono alphabetic substitution cipher

Q: how do Bob and Alice agree on key value?

Simple encryption scheme

substitution cipher: substituting one thing for another

- monoalphabetic cipher: substitute one letter for another

plaintext: abcdefghijklmnopqrstuvwxyz
ciphertext: mnbvcxzasdfghjklpoiuytrewq

The diagram illustrates a monoalphabetic substitution cipher. It shows two rows of letters. The top row is labeled "plaintext" and contains the letters a through z. The bottom row is labeled "ciphertext" and contains the letters m through q. Red arrows point vertically from each letter in the plaintext to its corresponding letter in the ciphertext, showing the mapping rule.

e.g.: Plaintext: bob. i love you. alice

ciphertext: nkn. s gktc wky. mgsbc



Encryption key: mapping from set of 26 letters
to set of 26 letters

A more sophisticated encryption approach

- n substitution ciphers, M_1, M_2, \dots, M_n
- cycling pattern:
 - e.g., $n=4: M_1, M_3, M_4, M_3, M_2; M_1, M_3, M_4, M_3, M_2; \dots$
- for each new plaintext symbol, use subsequent substitution pattern in cyclic pattern
 - dog: d from M_1 , o from M_3 , g from M_4



Encryption key: n substitution ciphers, and cyclic

pattern

- key need not be just n-bit pattern

Symmetric key crypto: DES

DES: Data Encryption Standard

- US encryption standard [NIST 1993]
- 56-bit symmetric key, 64-bit plaintext input
- block cipher with cipher block chaining
- how secure is DES?
 - DES Challenge: 56-bit-key-encrypted phrase decrypted (brute force) in less than a day
 - no known good analytic attack
- making DES more secure:
 - 3DES: encrypt 3 times with 3 different keys

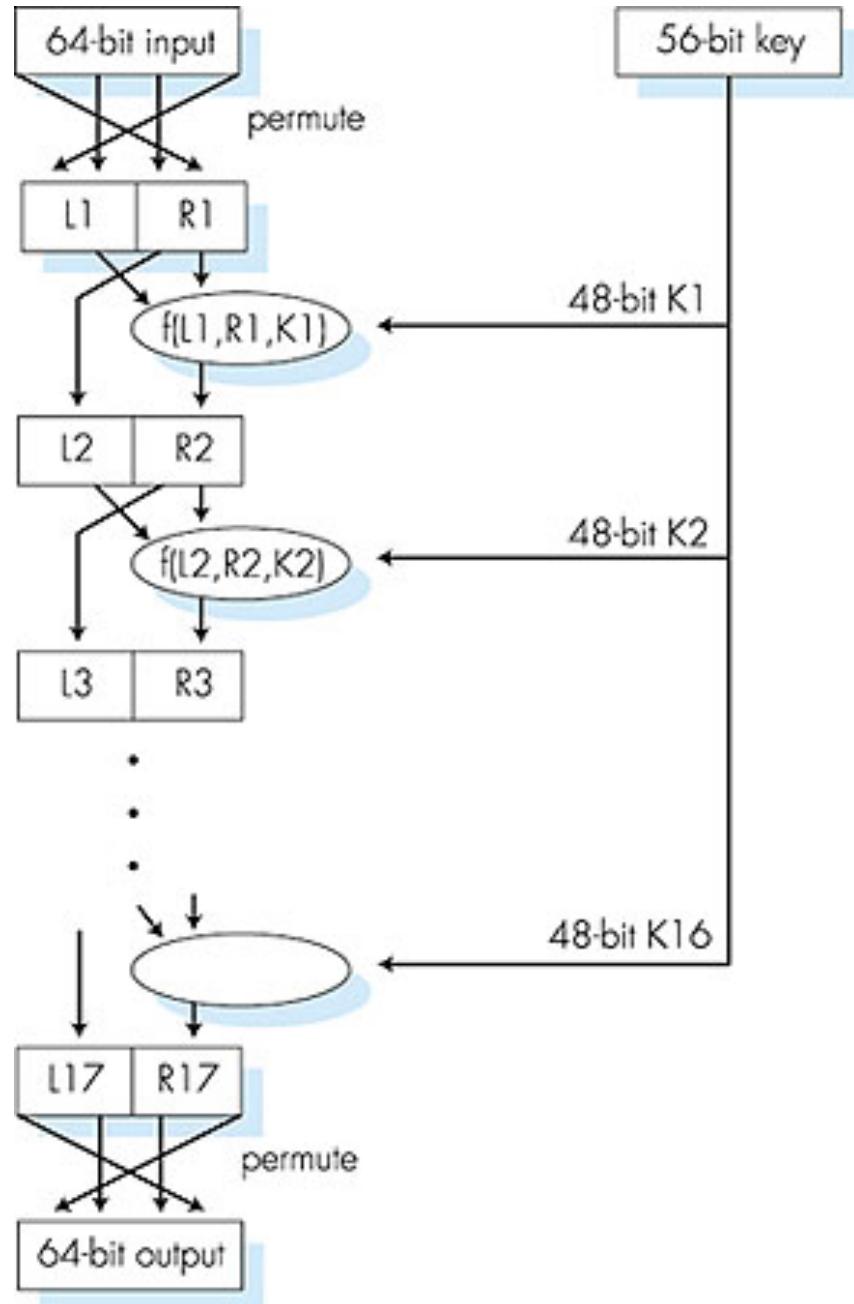
Symmetric key crypto: DES

DES operation

initial permutation

16 identical “rounds” of function application,
each using different 48 bits of key

final permutation



AES: Advanced Encryption Standard

- symmetric-key NIST standard, replaced DES (Nov 2001)
- processes data in 128 bit blocks
- 128, 192, or 256 bit keys
- brute force decryption (try each key) taking 1 second on DES, takes 149 trillion years for AES

Public Key Cryptography

symmetric key crypto

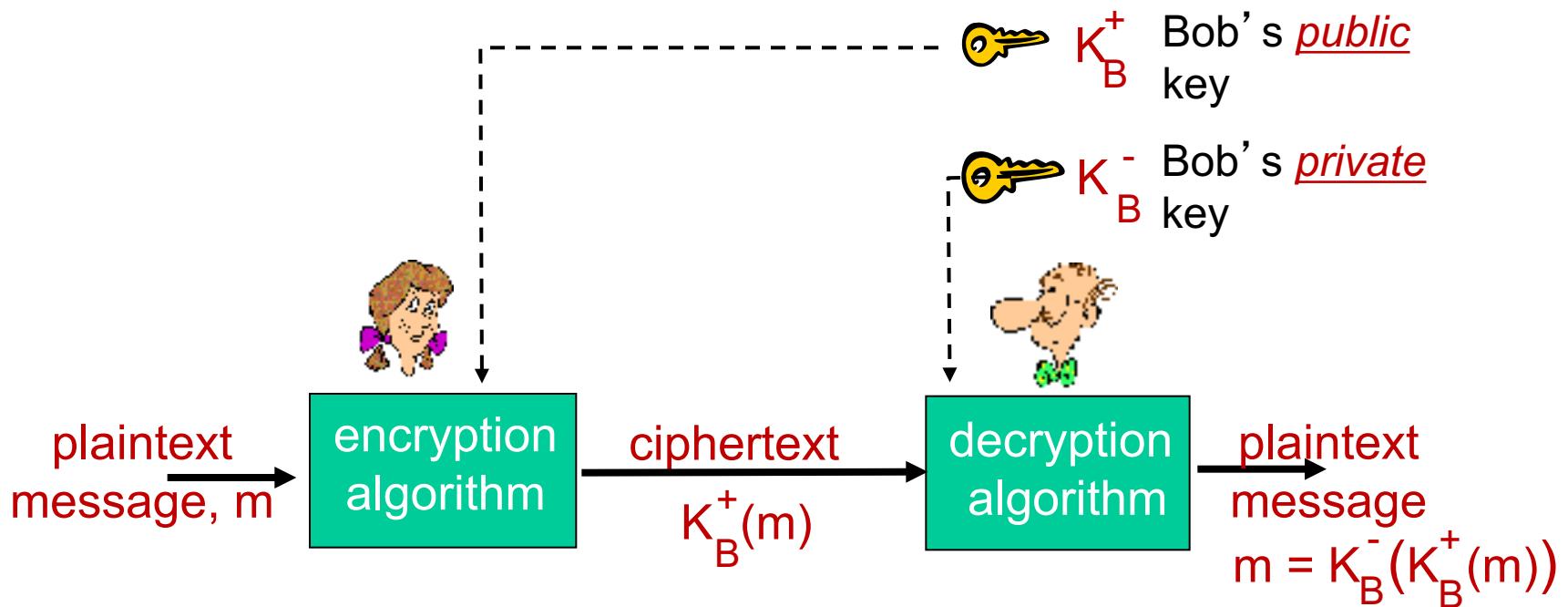
- requires sender, receiver know shared secret key
- Q: how to agree on key in first place (particularly if never “met”)?

public key crypto

- radically different approach [Diffie-Hellman76, RSA78]
- sender, receiver do *not* share secret key
- *public* encryption key known to *all*
- *private* decryption key known only to receiver



Public key cryptography



Public key encryption algorithms

requirements:

- ① need $K_B^+(.)$ and $K_B^-(.)$ such that

$$K_B^-(K_B^+(m)) = m$$

- ② given public key K_B^+ , it should be impossible to compute private key K_B^-

RSA: Rivest, Shamir, Adelson algorithm

Prerequisite: modular arithmetic

- $x \bmod n = \text{remainder of } x \text{ when divide by } n$
- facts:
 - $[(a \bmod n) + (b \bmod n)] \bmod n = (a+b) \bmod n$
 - $[(a \bmod n) - (b \bmod n)] \bmod n = (a-b) \bmod n$
 - $[(a \bmod n) * (b \bmod n)] \bmod n = (a*b) \bmod n$
- thus
$$(a \bmod n)^d \bmod n = a^d \bmod n$$
- example: $x=14, n=10, d=2:$
$$(x \bmod n)^d \bmod n = 4^2 \bmod 10 = 6$$
$$x^d = 14^2 = 196 \quad x^d \bmod 10 = 6$$

RSA: getting ready

- message: just a bit pattern
- bit pattern can be uniquely represented by an integer number
- thus, encrypting a message is equivalent to encrypting a number

example:

- $m = 10010001$. This message is uniquely represented by the decimal number 145.
- to encrypt m , we encrypt the corresponding number, which gives a new number (the ciphertext).

RSA: Creating public/private key pair

1. choose two large prime numbers p, q .
(e.g., 1024 bits each)
2. compute $n = pq$, $z = (p-1)(q-1)$
3. choose e (with $e < n$) that has no common factors with z (e, z are “relatively prime”).
4. choose d such that $ed - 1$ is exactly divisible by z .
(in other words: $ed \bmod z = 1$).
5. public key is $\underbrace{(n,e)}_{K_B^+}$. private key is $\underbrace{(n,d)}_{K_B^-}$.

RSA: encryption, decryption

0. given (n,e) and (n,d) as computed above

I. to encrypt message $m (< n)$, compute

$$c = m^e \bmod n$$

2. to decrypt received bit pattern, c , compute

$$m = c^d \bmod n$$

magic happens! $m = \underbrace{(m^e \bmod n)^d}_{c} \bmod n$

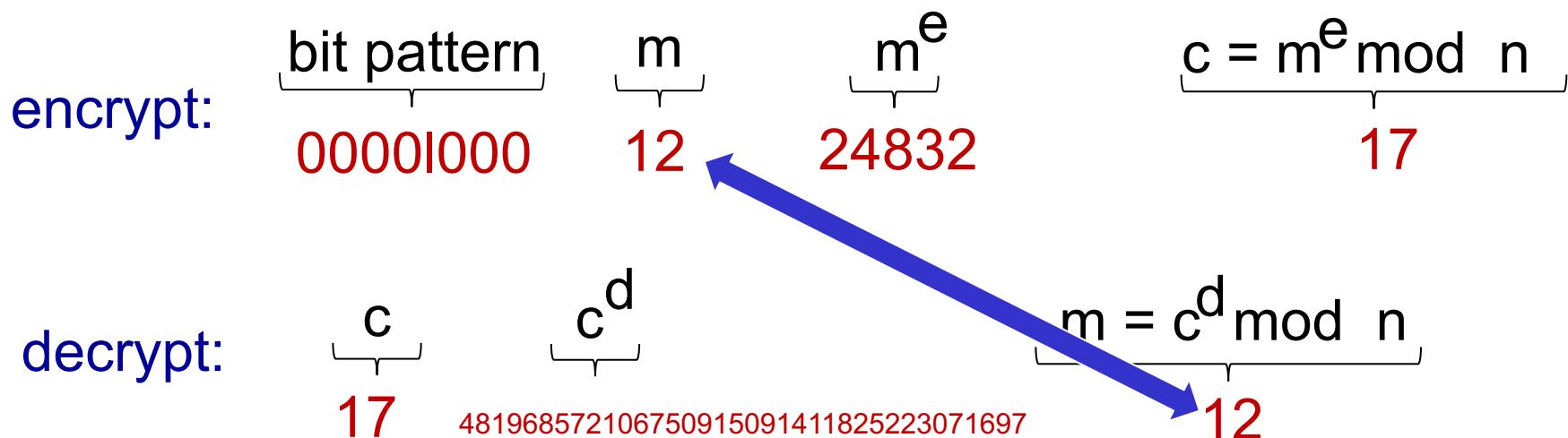
RSA example:

Bob chooses $p=5$, $q=7$. Then $n=35$, $z=24$.

$e=5$ (so e , z relatively prime).

$d=29$ (so $ed-1$ exactly divisible by z).

encrypting 8-bit messages.



Why does RSA work?

- must show that $c^d \bmod n = m$
where $c = m^e \bmod n$
- fact: for any x and y : $x^y \bmod n = x^{(y \bmod z)} \bmod n$
 - where $n = pq$ and $z = (p-1)(q-1)$
- thus,
$$\begin{aligned} c^d \bmod n &= (m^e \bmod n)^d \bmod n \\ &= m^{ed} \bmod n \\ &= m^{(ed \bmod z)} \bmod n \\ &= m^l \bmod n \\ &= m \end{aligned}$$

RSA: another important property

The following property will be **very** useful later:

$$\underbrace{K_B^-(K_B^+(m))}_{\text{use public key first,}} = m = \underbrace{K_B^+(K_B^-(m))}_{\text{use private key first, followed by public key}}$$

use public key first,
followed by
private key

use private key
first, followed by
public key

result is the same!

Why $K_B^{-1}(K_B^+(m)) = m = K_B^+(K_B^{-1}(m))$?

follows directly from modular arithmetic:

$$\begin{aligned}(m^e \bmod n)^d \bmod n &= m^{ed} \bmod n \\&= m^{de} \bmod n \\&= (m^d \bmod n)^e \bmod n\end{aligned}$$

Why is RSA secure?

- suppose you know Bob's public key (n, e) . How hard is it to determine d ?
- essentially need to find factors of n without knowing the two factors p and q
 - fact: factoring a big number is hard

RSA in practice: session keys

- exponentiation in RSA is computationally intensive
- DES is at least 100 times faster than RSA
- use public key crypto to establish secure connection, then establish second key – symmetric session key – for encrypting data

session key, K_S

- Bob and Alice use RSA to exchange a symmetric key K_S
- once both have K_S , they use symmetric key cryptography

Chapter 8 roadmap

- 8.1 What is network security?
- 8.2 Principles of cryptography
- 8.3 Message integrity, *authentication*
- 8.4 Securing e-mail
- 8.5 Securing TCP connections: SSL
- 8.6 Network layer security: IPsec
- 8.7 Securing wireless LANs
- 8.8 Operational security: firewalls and IDS

Authentication

Goal: Bob wants Alice to “prove” her identity to him

Protocol 1.0: Alice says “I am Alice”



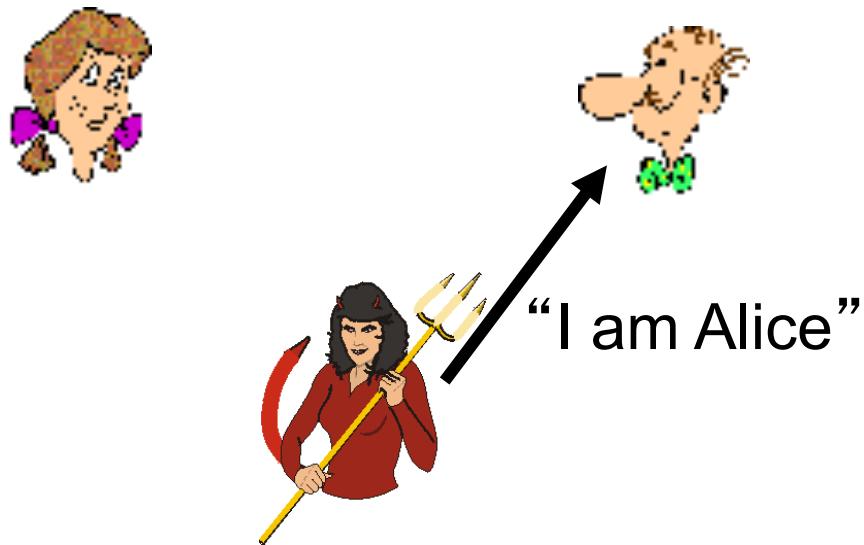
Failure scenario??



Authentication

Goal: Bob wants Alice to “prove” her identity to him

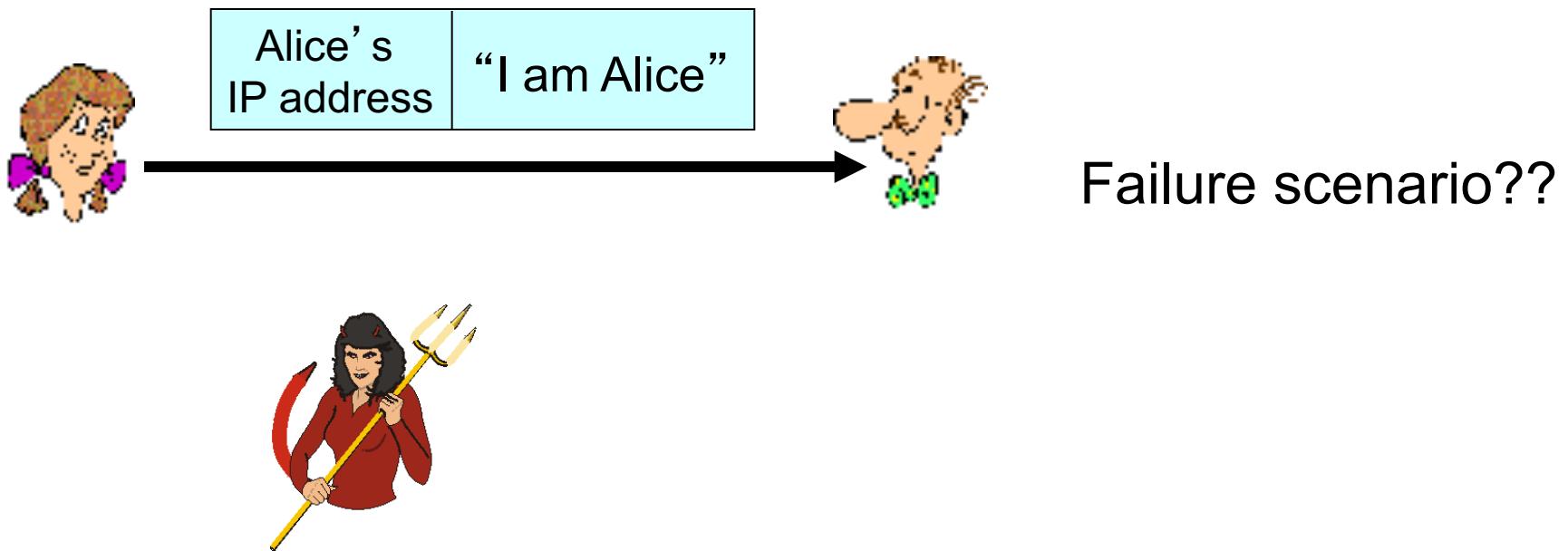
Protocol ap1.0: Alice says “I am Alice”



in a network,
Bob can not “see” Alice,
so Trudy simply declares
herself to be Alice

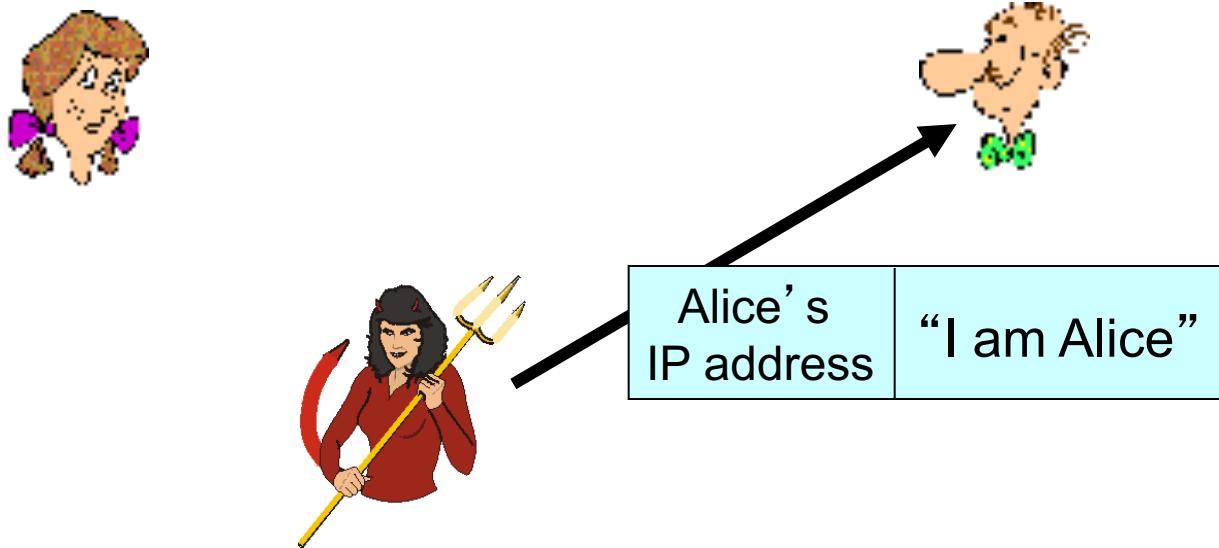
Authentication: another try

Protocol ap2.0: Alice says “I am Alice” in an IP packet containing her source IP address



Authentication: another try

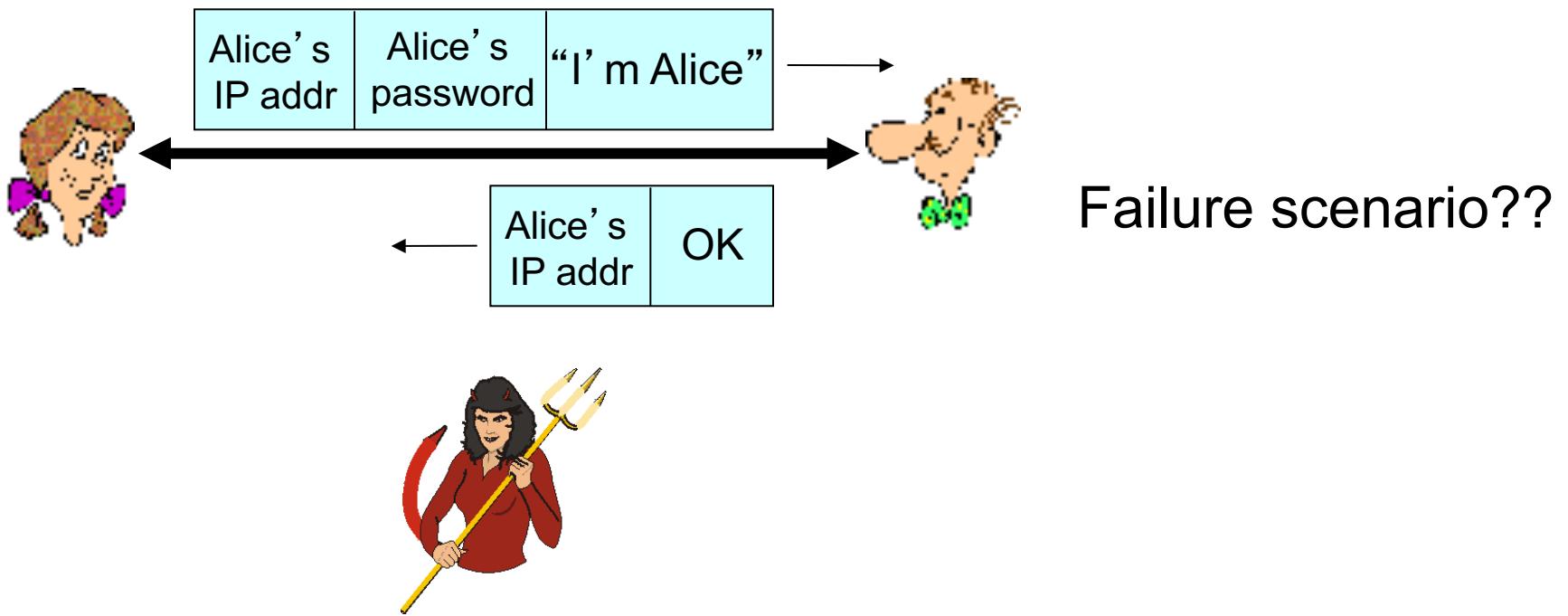
Protocol ap2.0: Alice says “I am Alice” in an IP packet containing her source IP address



Trudy can create
a packet
“spoofing”
Alice’s address

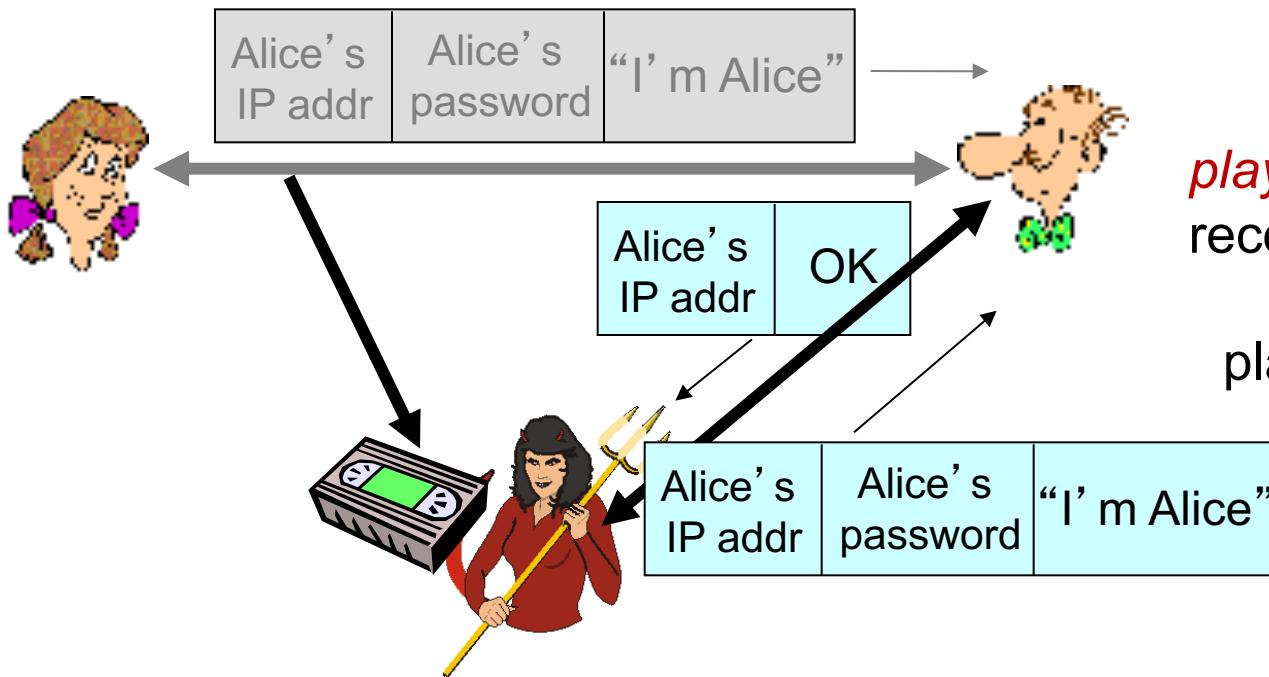
Authentication: another try

Protocol ap3.0: Alice says “I am Alice” and sends her secret password to “prove” it.



Authentication: another try

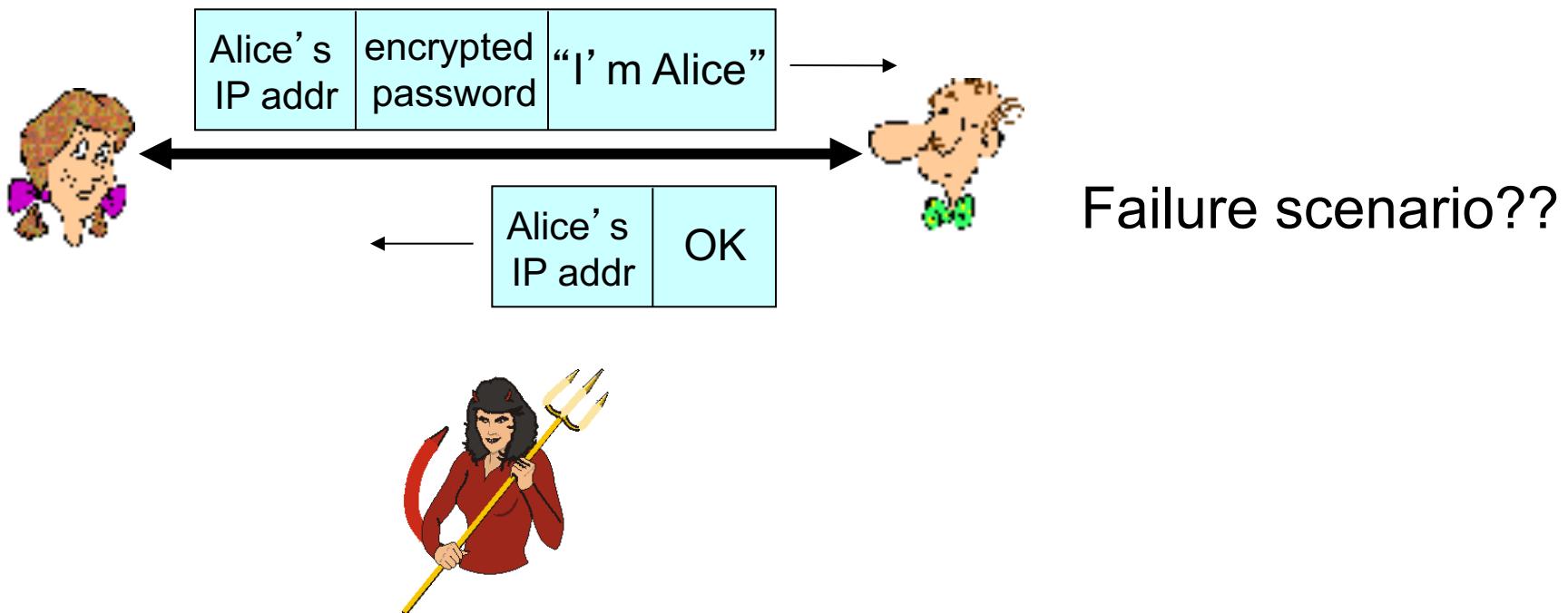
Protocol ap3.0: Alice says “I am Alice” and sends her secret password to “prove” it.



playback attack: Trudy records Alice's packet and later plays it back to Bob

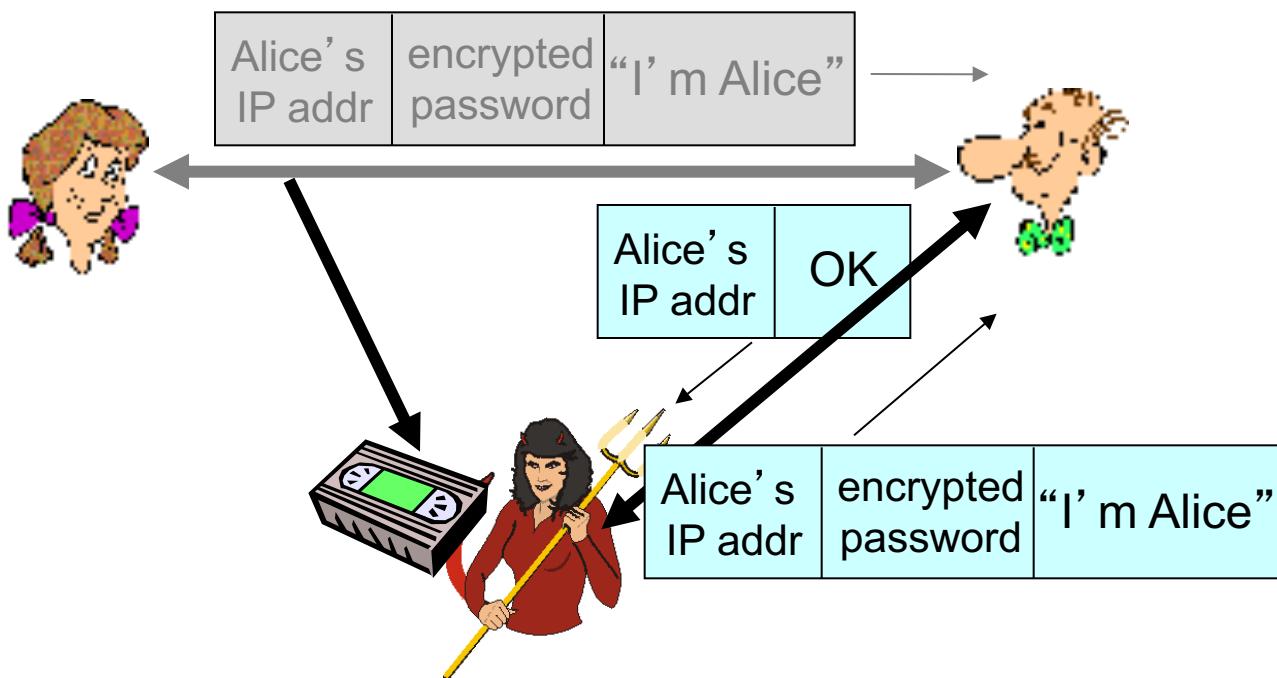
Authentication: yet another try

Protocol ap3.1: Alice says “I am Alice” and sends her **encrypted** secret password to “prove” it.



Authentication: yet another try

Protocol ap3.1: Alice says “I am Alice” and sends her **encrypted** secret password to “prove” it.



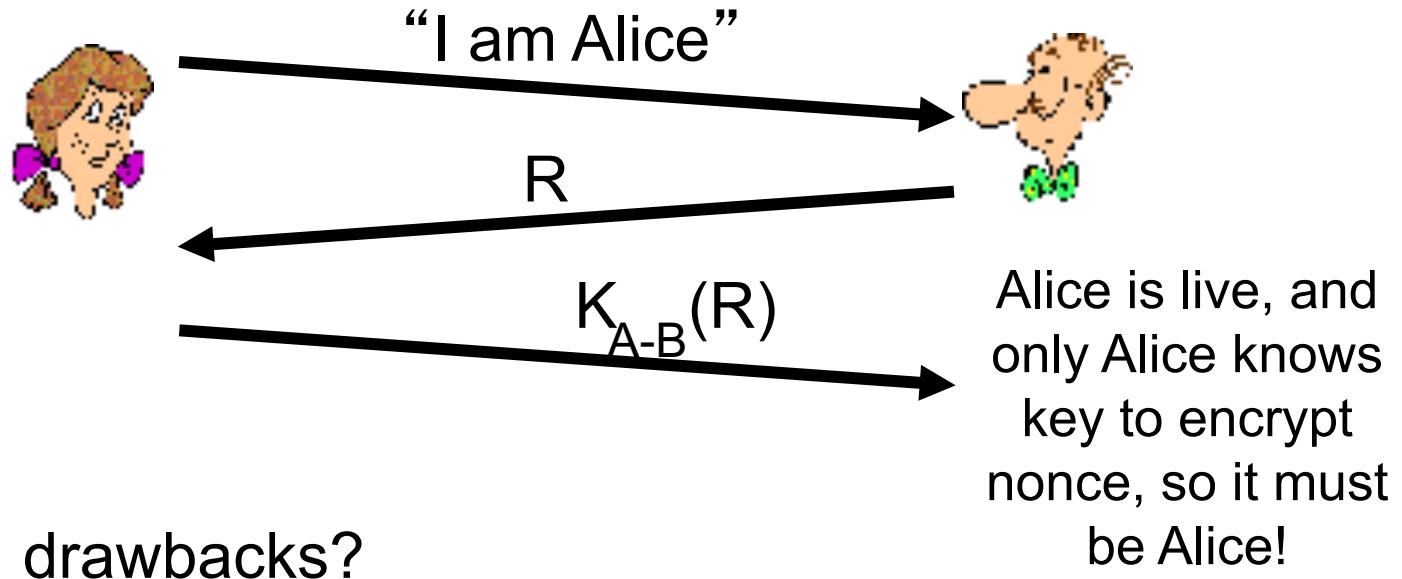
record
and
playback
still works!

Authentication: yet another try

Goal: avoid playback attack

nonce: number (R) used only *once-in-a-lifetime*

ap4.0: to prove Alice “live”, Bob sends Alice **nonce**, R . Alice must return R , encrypted with shared secret key



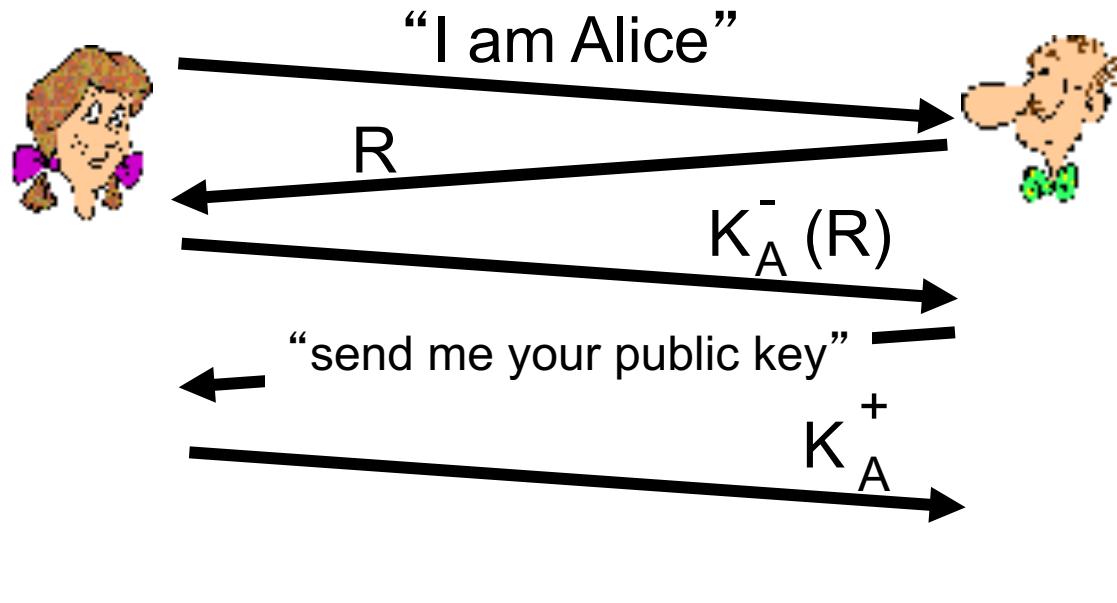
Failures, drawbacks?

Authentication: ap5.0

ap4.0 requires shared symmetric key

- can we authenticate using public key techniques?

ap5.0: use nonce, public key cryptography

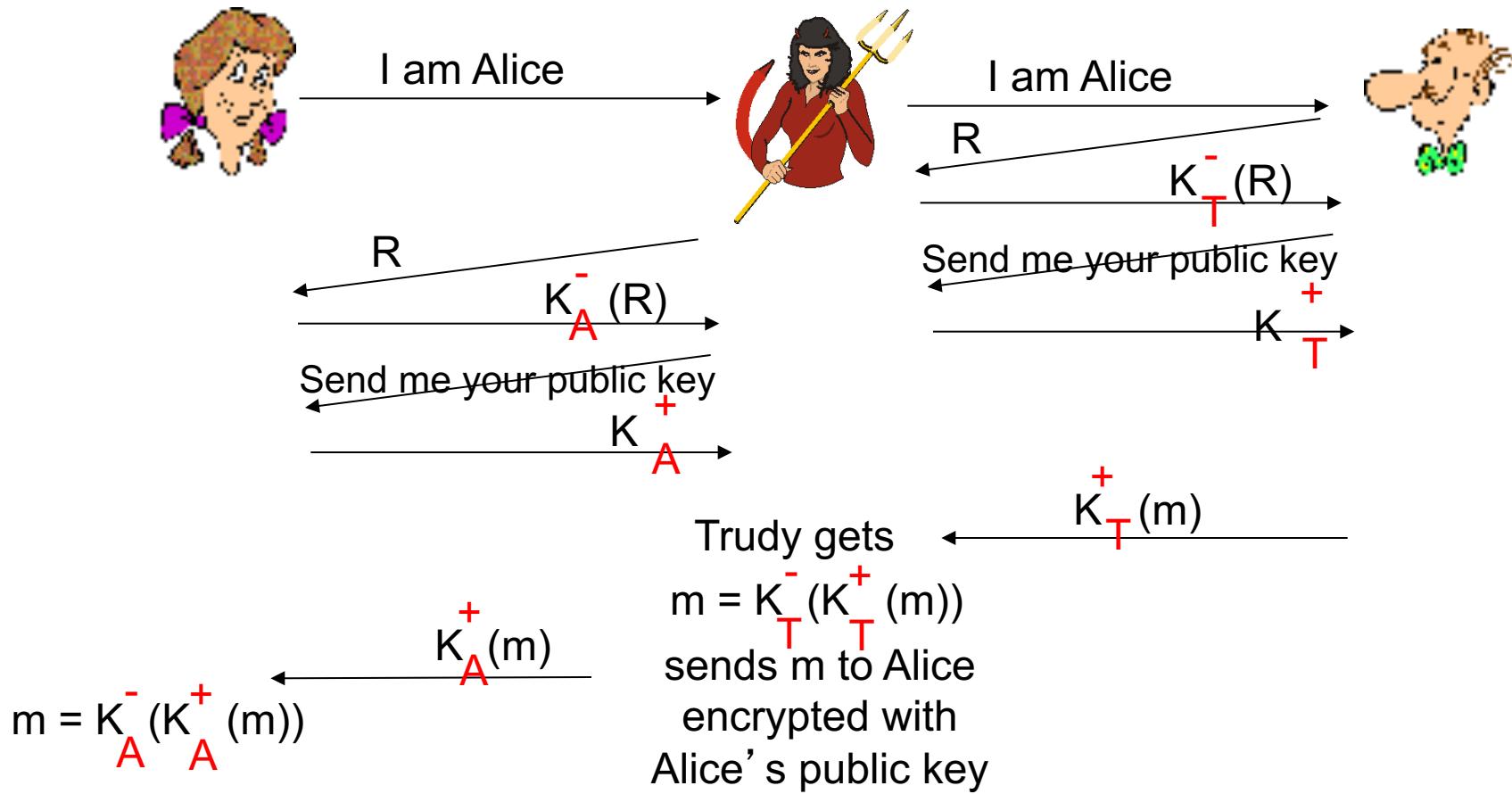


Bob computes
 $K_A^+(K_A^-(R)) = R$

and knows only Alice could have the private key, that encrypted R such that
 $K_A^+(K_A^-(R)) = R$

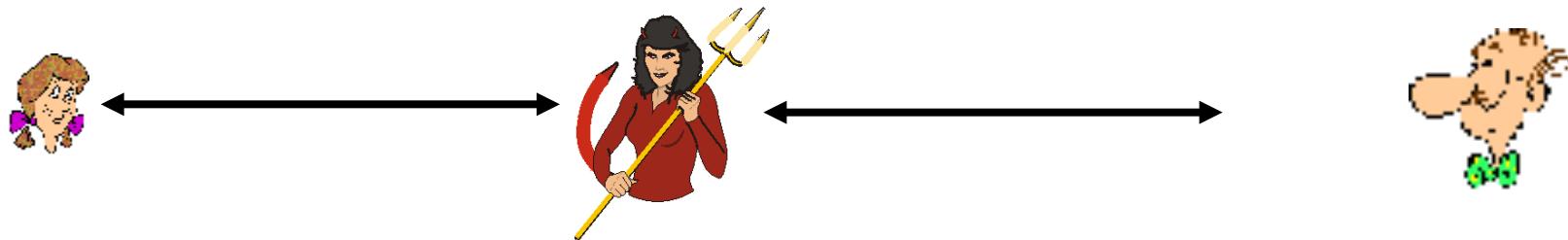
ap5.0: security hole

man (or woman) in the middle attack: Trudy poses as Alice (to Bob) and as Bob (to Alice)



ap5.0: security hole

man (or woman) in the middle attack: Trudy poses as Alice (to Bob) and as Bob (to Alice)



difficult to detect:

- Bob receives everything that Alice sends, and vice versa.
(e.g., so Bob, Alice can meet one week later and recall conversation!)
- problem is that Trudy receives all messages as well!

Chapter 8 roadmap

- 8.1 What is network security?
- 8.2 Principles of cryptography
- 8.3 *Message integrity*, authentication
- 8.4 Securing e-mail
- 8.5 Securing TCP connections: SSL
- 8.6 Network layer security: IPsec
- 8.7 Securing wireless LANs
- 8.8 Operational security: firewalls and IDS

Digital signatures

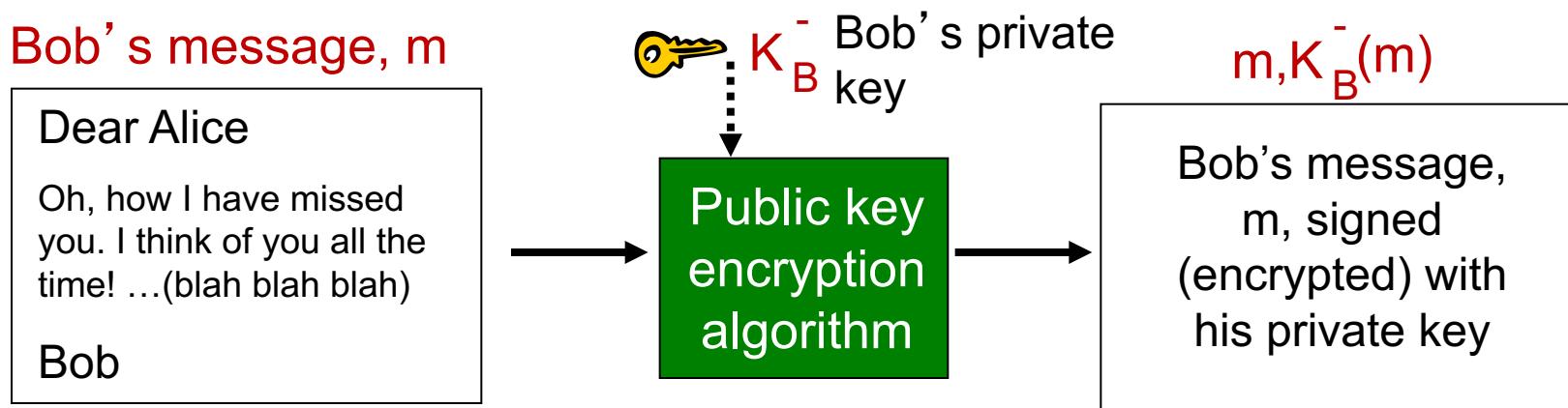
cryptographic technique analogous to hand-written signatures:

- sender (Bob) digitally signs document, establishing he is document owner/creator.
- *verifiable, nonforgeable*: recipient (Alice) can prove to someone that Bob, and no one else (including Alice), must have signed document

Digital signatures

simple digital signature for message m :

- Bob signs m by encrypting with his private key K_B^- , creating “signed” message, $K_B^-(m)$



Digital signatures

- suppose Alice receives msg m , with signature: $m, K_B^-(m)$
- Alice verifies m signed by Bob by applying Bob's public key K_B^+ to $K_B^-(m)$ then checks $K_B(K_B^+(K_B^-(m))) = m$.
- If $K_B(K_B^-(m)) = m$, whoever signed m must have used Bob's private key.

Alice thus verifies that:

- Bob signed m
- no one else signed m
- Bob signed m and not m'

non-repudiation:

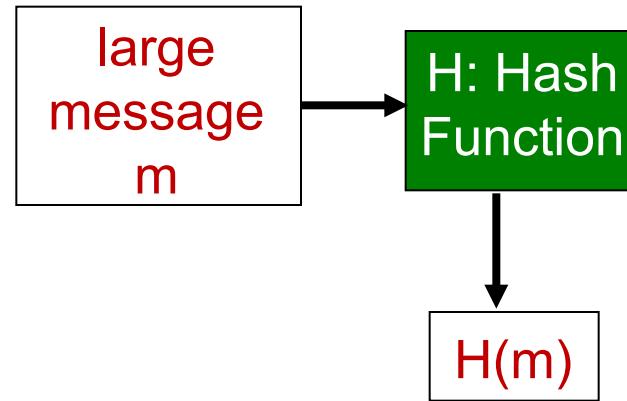
- ✓ Alice can take m , and signature $K_B^-(m)$ to court and prove that Bob signed m

Message digests

computationally expensive to public-key-encrypt long messages

goal: fixed-length, easy-to-compute digital “fingerprint”

- apply hash function H to m , get fixed size message digest, $H(m)$.



Hash function properties:

- many-to-1
- produces fixed-size msg digest (fingerprint)
- given message digest x , computationally infeasible to find m such that $x = H(m)$

Internet checksum: poor crypto hash function

Internet checksum has some properties of hash function:

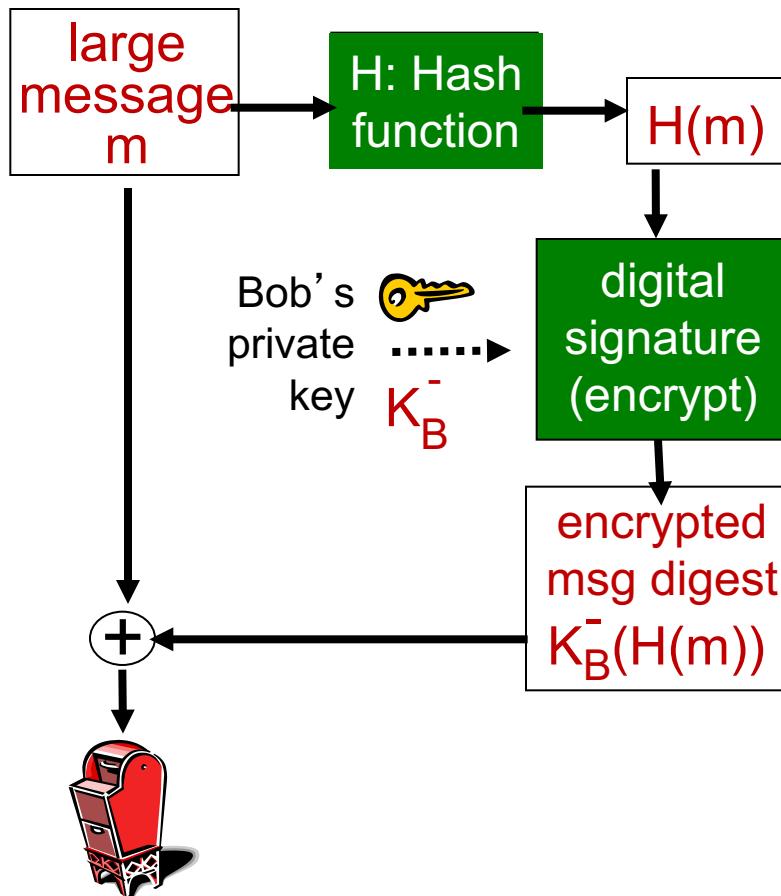
- produces fixed length digest (16-bit sum) of message
- is many-to-one

But given message with given hash value, it is easy to find another message with same hash value:

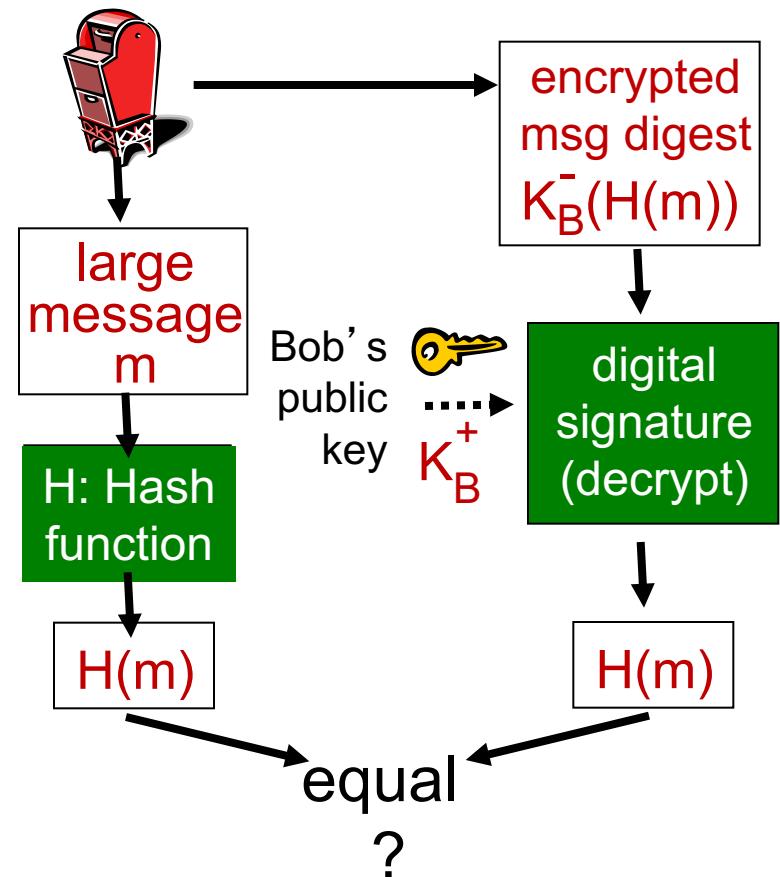
<u>message</u>	<u>ASCII format</u>	<u>message</u>	<u>ASCII format</u>
I O U 1	49 4F 55 31	I O U <u>9</u>	49 4F 55 <u>39</u>
0 0 . 9	30 30 2E 39	0 0 . <u>1</u>	30 30 2E <u>31</u>
9 B O B	39 42 D2 42	9 B O B	39 42 D2 42
<hr/>		<hr/>	
B2 C1 D2 AC		different messages but identical checksums!	

Digital signature = signed message digest

Bob sends digitally signed message:



Alice verifies signature, integrity of digitally signed message:

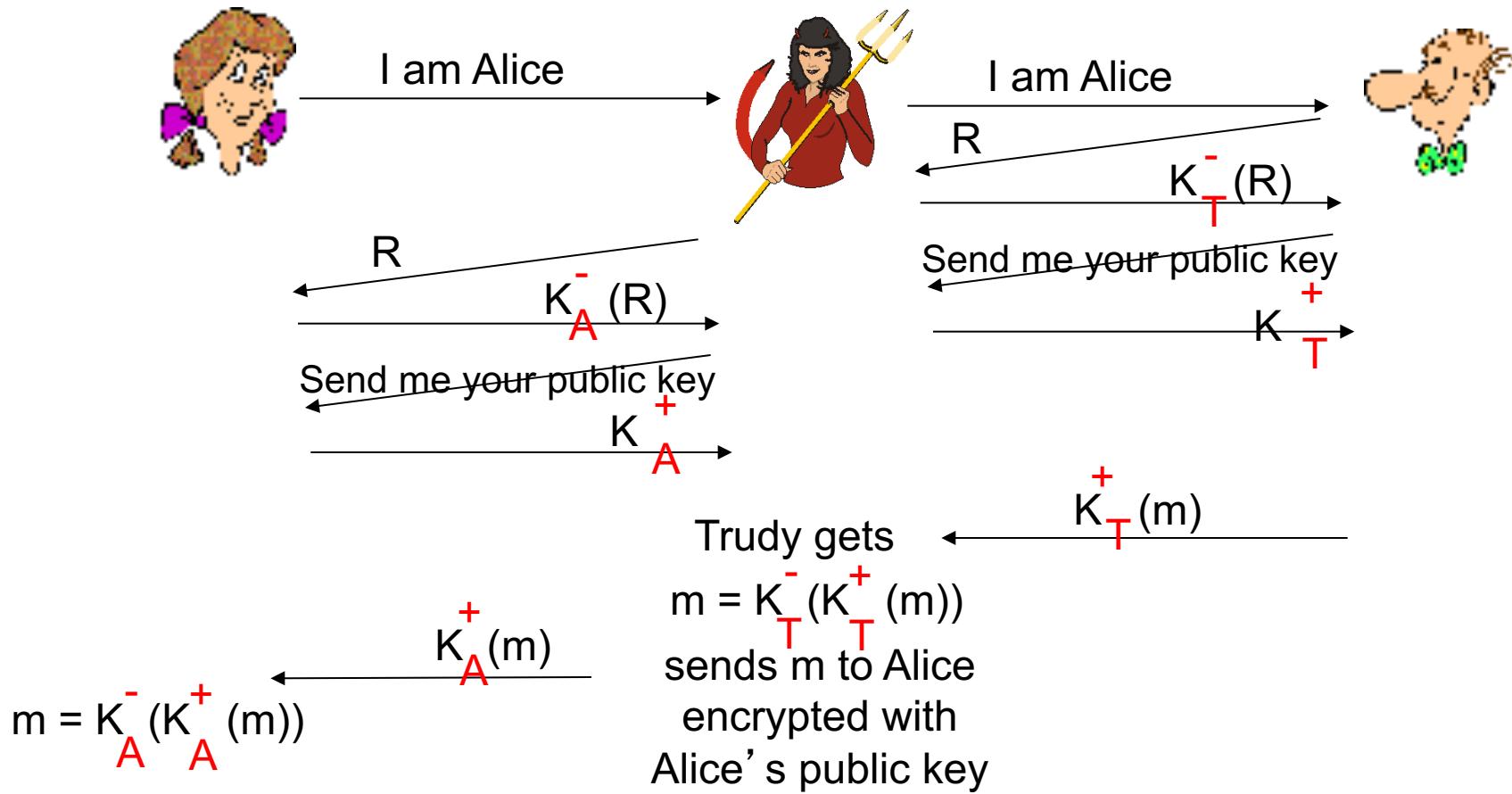


Hash function algorithms

- **MD5 hash function widely used (RFC 1321)**
 - computes 128-bit message digest in 4-step process.
 - arbitrary 128-bit string x , appears difficult to construct msg m whose MD5 hash is equal to x
- **SHA-1 is also used**
 - US standard [NIST, FIPS PUB 180-1]
 - 160-bit message digest

Recall: ap5.0 security hole

man (or woman) in the middle attack: Trudy poses as Alice (to Bob) and as Bob (to Alice)

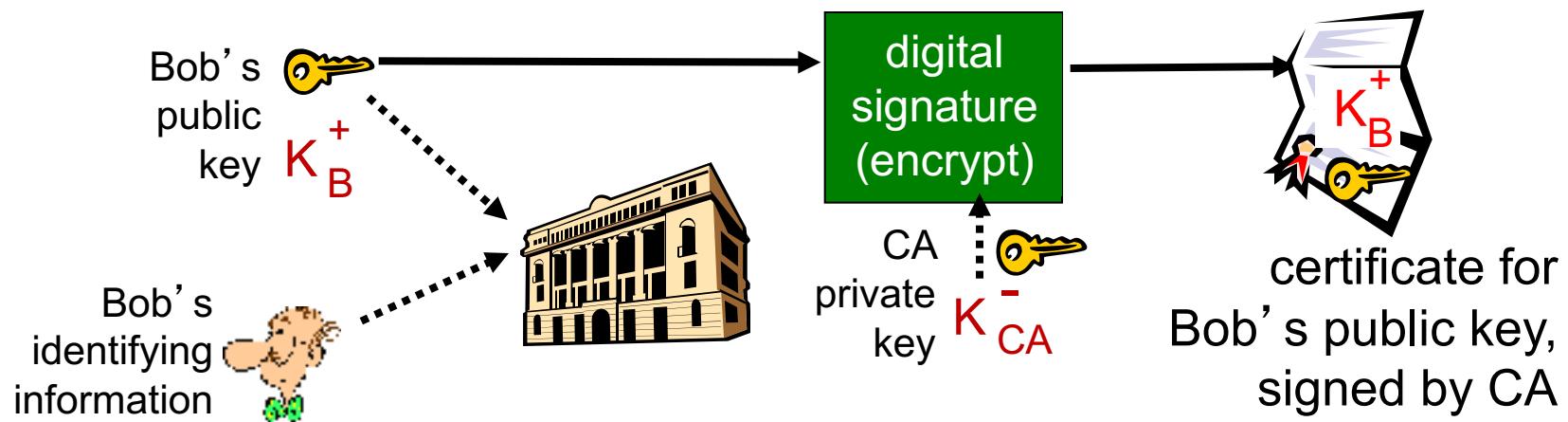


Public-key certification

- motivation: Trudy plays pizza prank on Bob
 - Trudy creates e-mail order:
Dear Pizza Store, Please deliver to me four pepperoni pizzas. Thank you, Bob
 - Trudy signs order with her private key
 - Trudy sends order to Pizza Store
 - Trudy sends to Pizza Store her public key, but says it's Bob's public key
 - Pizza Store verifies signature; then delivers four pepperoni pizzas to Bob
 - Bob doesn't even like pepperoni

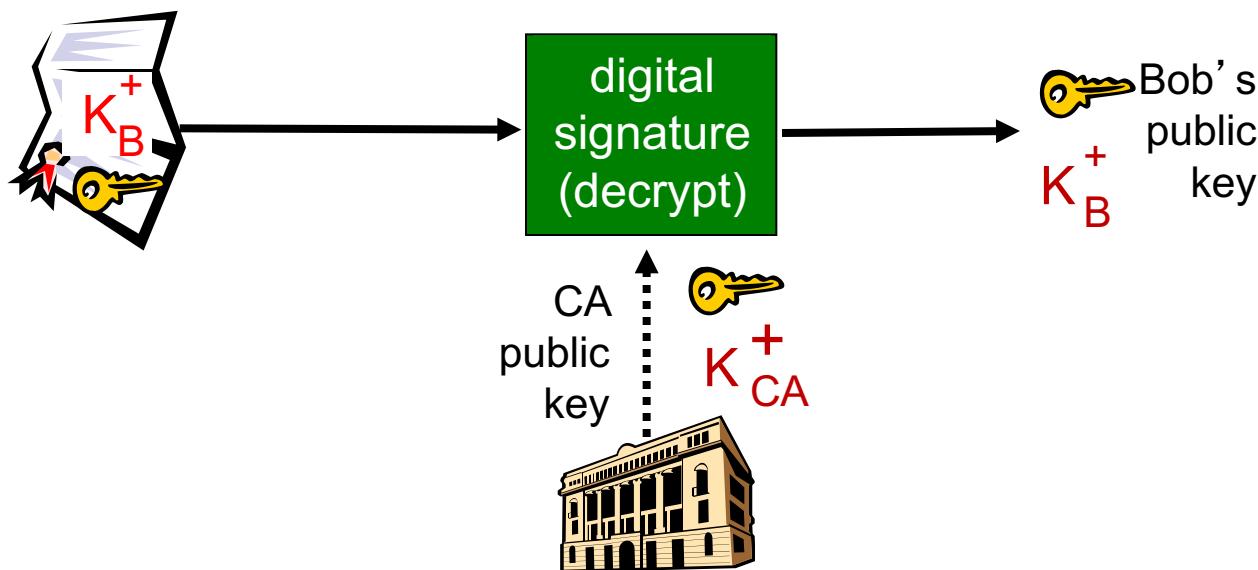
Certification authorities

- ***certification authority (CA)***: binds public key to particular entity, E.
- E (person, router) registers its public key with CA.
 - E provides “proof of identity” to CA.
 - CA creates certificate binding E to its public key.
 - certificate containing E’s public key digitally signed by CA – CA says “this is E’s public key”



Certification authorities

- when Alice wants Bob's public key:
 - gets Bob's certificate (Bob or elsewhere).
 - apply CA's public key to Bob's certificate, get Bob's public key

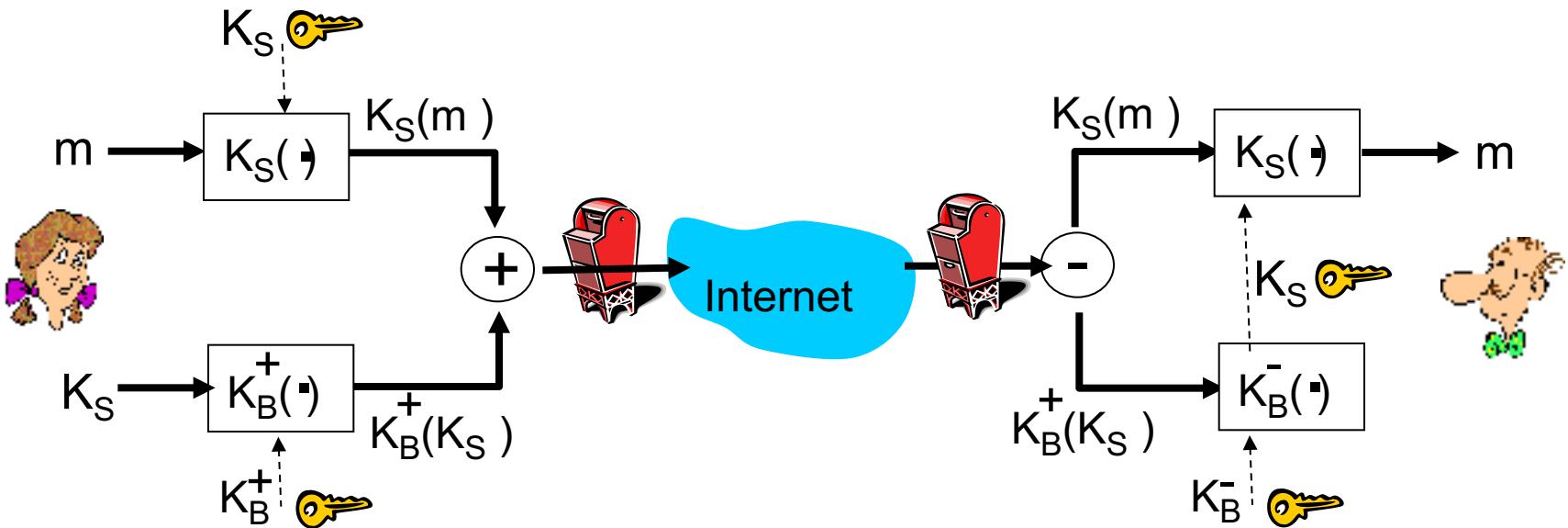


Chapter 8 roadmap

- 8.1 What is network security?
- 8.2 Principles of cryptography
- 8.3 Message integrity, authentication
- 8.4 *Securing e-mail*
- 8.5 Securing TCP connections: SSL
- 8.6 Network layer security: IPsec
- 8.7 Securing wireless LANs
- 8.8 Operational security: firewalls and IDS

Secure e-mail

Alice wants to send confidential e-mail, m , to Bob.

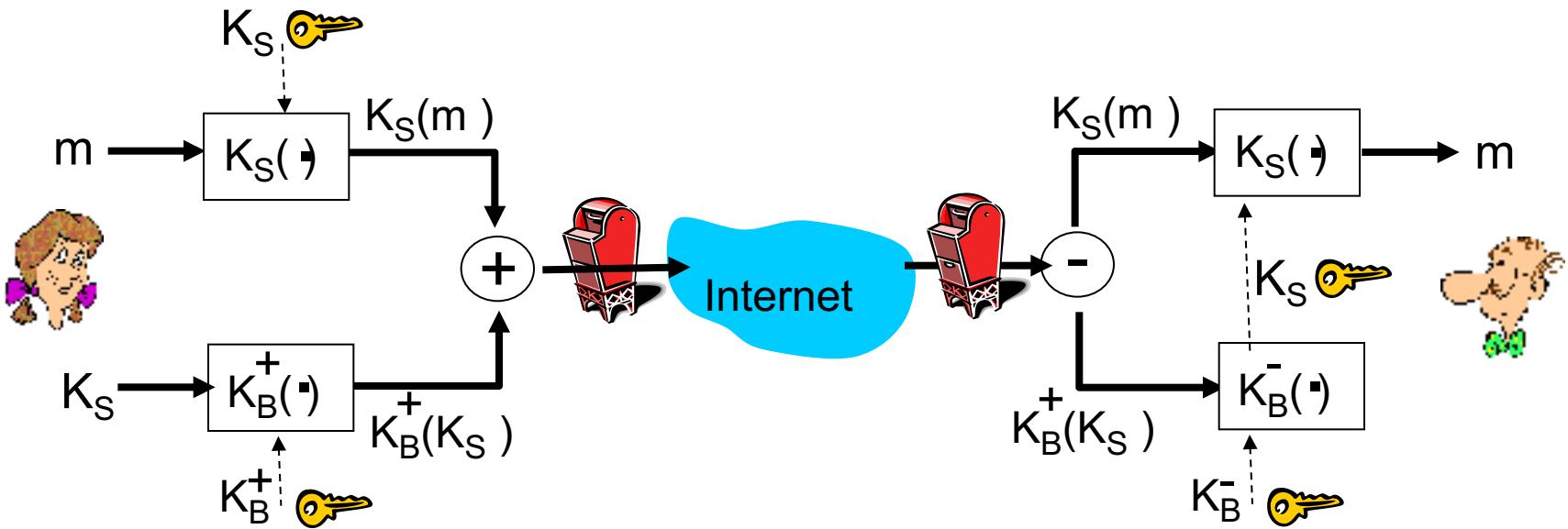


Alice:

- generates random *symmetric* private key, K_S
- encrypts message with K_S (for efficiency)
- also encrypts K_S with Bob's public key
- sends both $K_S(m)$ and $K_B^+(K_S)$ to Bob

Secure e-mail

Alice wants to send confidential e-mail, m , to Bob.

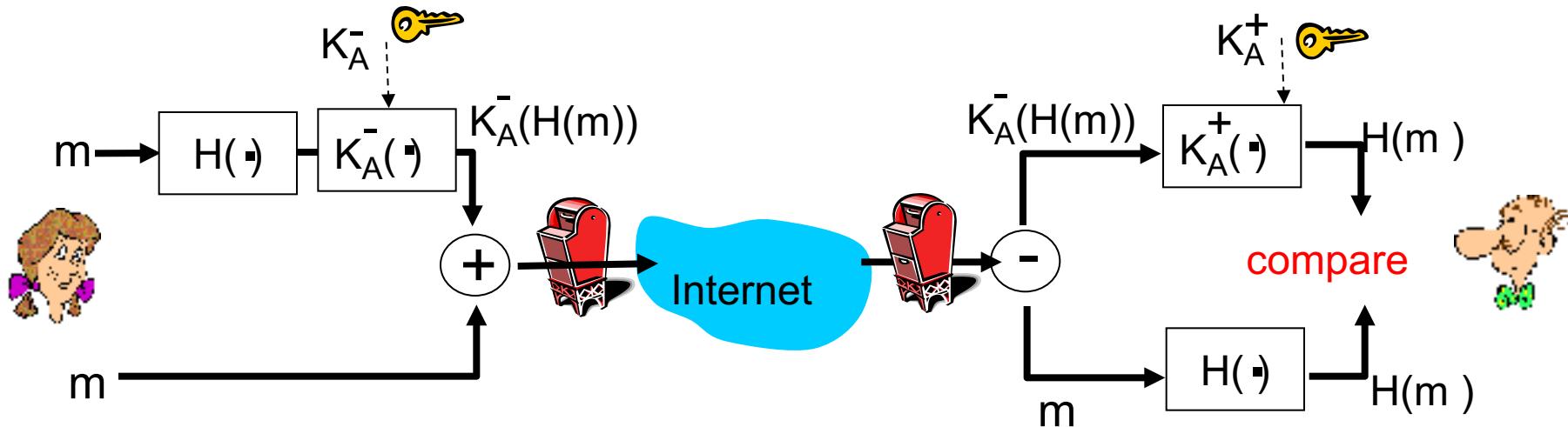


Bob:

- uses his private key to decrypt and recover K_S
- uses K_S to decrypt $K_S(m)$ to recover m

Secure e-mail (continued)

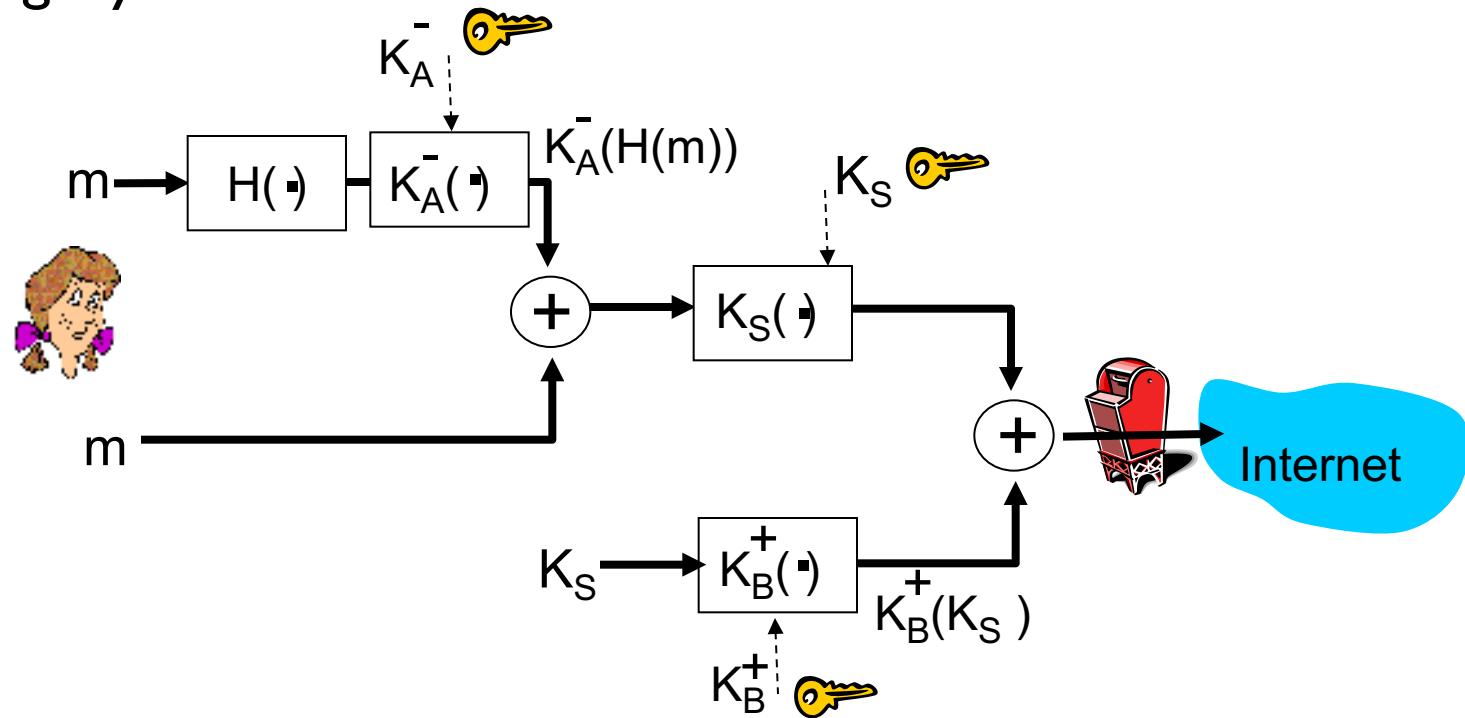
Alice wants to provide sender authentication message integrity



- Alice digitally signs message
- sends both message (in the clear) and digital signature

Secure e-mail (continued)

Alice wants to provide secrecy, sender authentication, message integrity.



Alice uses three keys: her private key, Bob's public key, newly created symmetric key

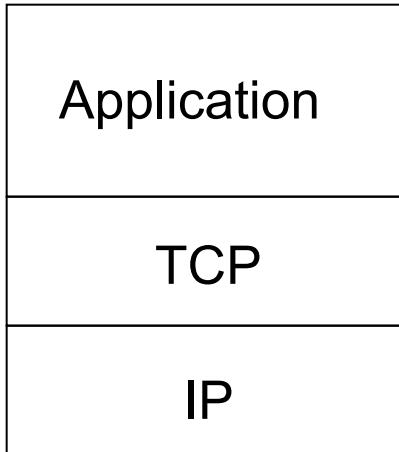
Chapter 8 roadmap

- 8.1 What is network security?
- 8.2 Principles of cryptography
- 8.3 Message integrity
- 8.4 Securing e-mail
- 8.5 *Securing TCP connections: SSL*
- 8.6 Network layer security: IPsec
- 8.7 Securing wireless LANs
- 8.8 Operational security: firewalls and IDS

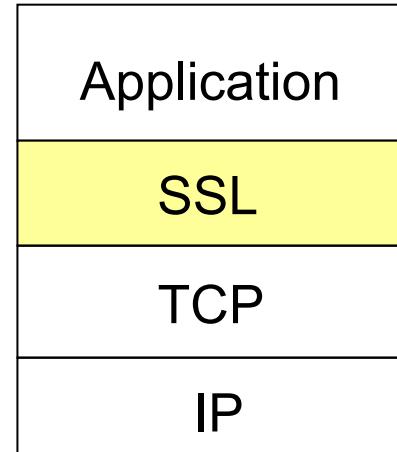
SSL: Secure Sockets Layer

- widely deployed security protocol
 - supported by almost all browsers, web servers
 - https
 - billions \$/year over SSL
- mechanisms: [Woo 1994], implementation: Netscape
- variation -TLS: transport layer security, RFC 2246
- provides
 - *confidentiality*
 - *integrity*
 - *authentication*
- original goals:
 - Web e-commerce transactions
 - encryption (especially credit-card numbers)
 - Web-server authentication
 - optional client authentication
 - minimum hassle in doing business with new merchant
- available to all TCP applications
 - secure socket interface

SSL and TCP/IP



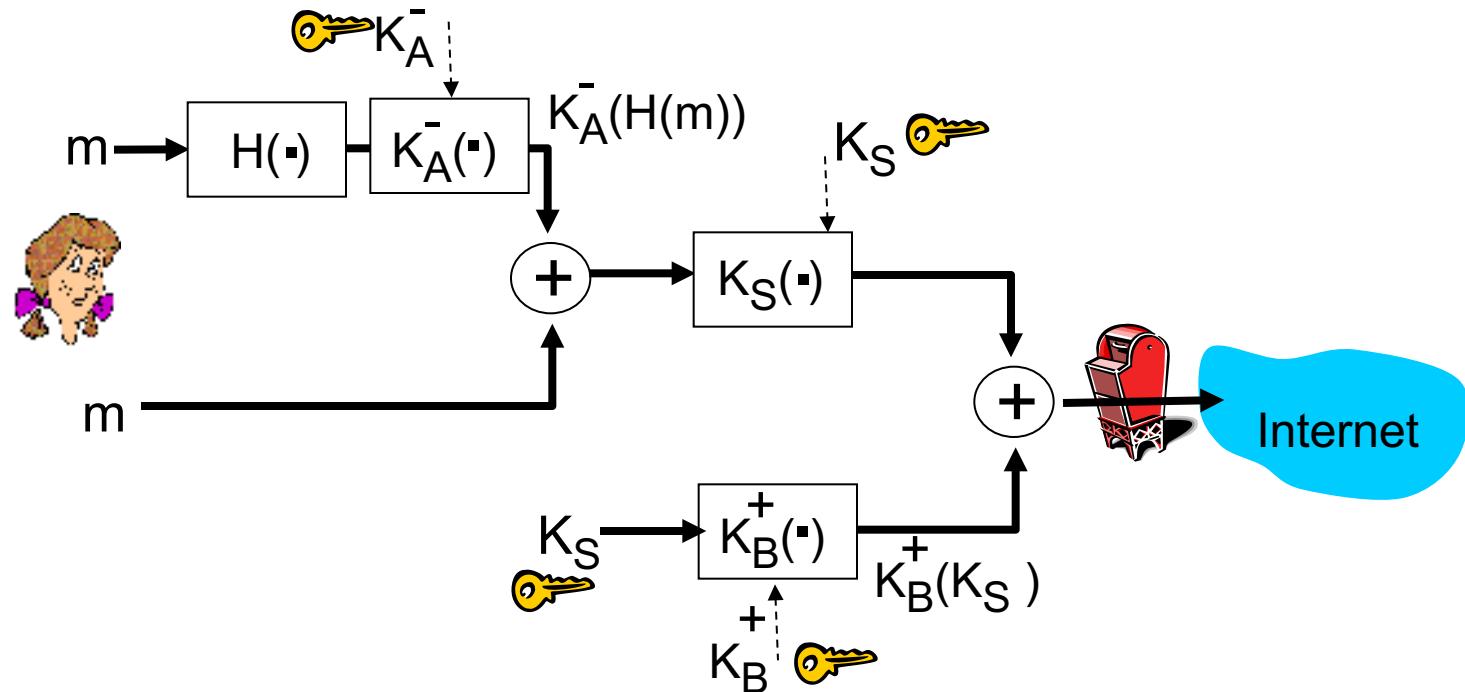
normal application



application with SSL

- SSL provides application programming interface (API) to applications
- C and Java SSL libraries/classes readily available

Could do something like PGP:

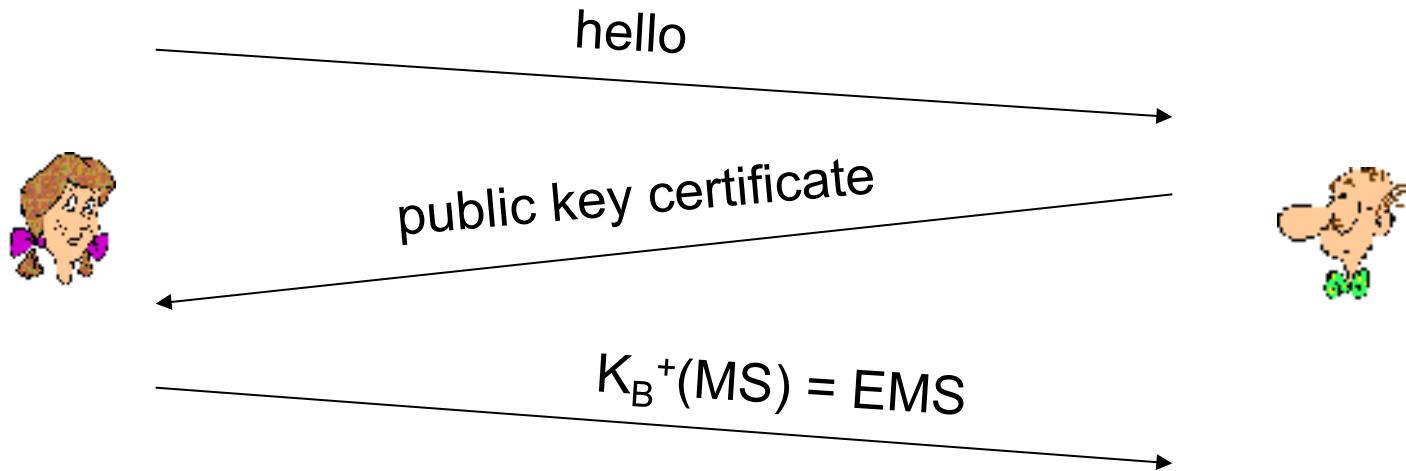


- but want to send byte streams & interactive data
- want set of secret keys for entire connection
- want certificate exchange as part of protocol: handshake phase

Toy SSL: a simple secure channel

- *handshake*: Alice and Bob use their certificates, private keys to authenticate each other and exchange shared secret
- *key derivation*: Alice and Bob use shared secret to derive set of keys
- *data transfer*: data to be transferred is broken up into series of records
- *connection closure*: special messages to securely close connection

Toy: a simple handshake



MS: master secret

EMS: encrypted master secret

Toy: key derivation

- considered bad to use same key for more than one cryptographic operation
 - use different keys for message authentication code (MAC) and encryption
- four keys:
 - K_c = encryption key for data sent from client to server
 - M_c = MAC key for data sent from client to server
 - K_s = encryption key for data sent from server to client
 - M_s = MAC key for data sent from server to client
- keys derived from key derivation function (KDF)
 - takes master secret and (possibly) some additional random data and creates the keys

Toy: data records

- why not encrypt data in constant stream as we write it to TCP?
 - where would we put the MAC? If at end, no message integrity until all data processed.
 - e.g., with instant messaging, how can we do integrity check over all bytes sent before displaying?
- instead, break stream in series of records
 - each record carries a MAC
 - receiver can act on each record as it arrives
- issue: in record, receiver needs to distinguish MAC from data
 - want to use variable-length records



Toy: sequence numbers

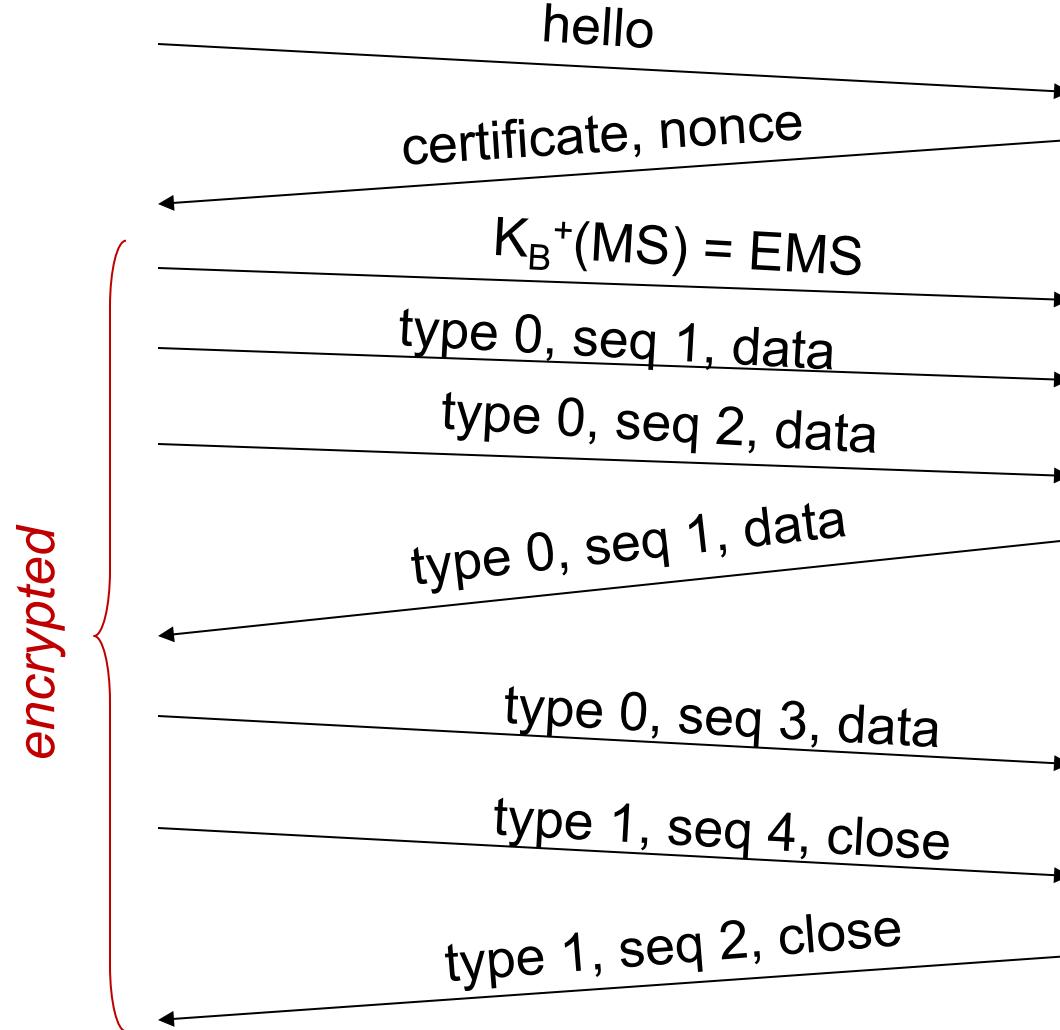
- *problem:* attacker can capture and replay record or re-order records
- *solution:* put sequence number into MAC:
 - $\text{MAC} = \text{MAC}(M_x, \text{sequence} \parallel \text{data})$
 - note: no sequence number field
- *problem:* attacker could replay all records
- *solution:* use nonce

Toy: control information

- *problem:* truncation attack:
 - attacker forges TCP connection close segment
 - one or both sides thinks there is less data than there actually is.
- *solution:* record types, with one type for closure
 - type 0 for data; type 1 for closure
- $\text{MAC} = \text{MAC}(M_x, \text{sequence} \parallel \text{type} \parallel \text{data})$



Toy SSL: summary



bob.com

Toy SSL isn't complete

- how long are fields?
- which encryption protocols?
- want negotiation?
 - allow client and server to support different encryption algorithms
 - allow client and server to choose together specific algorithm before data transfer

SSL cipher suite

- cipher suite
 - public-key algorithm
 - symmetric encryption algorithm
 - MAC algorithm
- SSL supports several cipher suites
- negotiation: client, server agree on cipher suite
 - client offers choice
 - server picks one

common SSL symmetric ciphers

- DES – Data Encryption Standard: block
- 3DES – Triple strength: block
- RC2 – Rivest Cipher 2: block
- RC4 – Rivest Cipher 4: stream

SSL Public key encryption

- RSA

Real SSL: handshake (I)

Purpose

1. server authentication
2. negotiation: agree on crypto algorithms
3. establish keys
4. client authentication (optional)

Real SSL: handshake (2)

1. client sends list of algorithms it supports, along with client nonce
2. server chooses algorithms from list; sends back: choice + certificate + server nonce
3. client verifies certificate, extracts server's public key, generates pre_master_secret, encrypts with server's public key, sends to server
4. client and server independently compute encryption and MAC keys from pre_master_secret and nonces
5. client sends a MAC of all the handshake messages
6. server sends a MAC of all the handshake messages

Real SSL: handshaking (3)

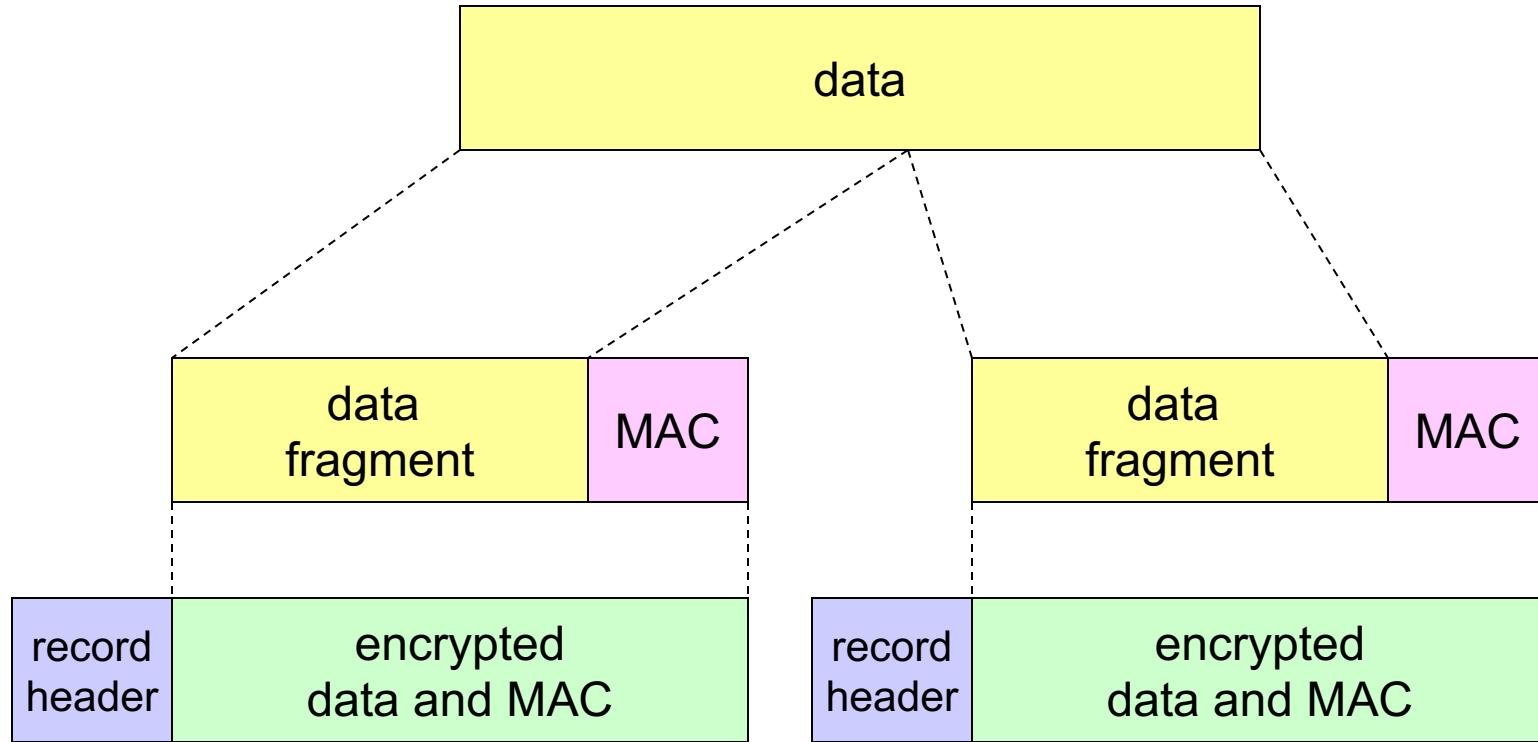
last 2 steps protect handshake from tampering

- client typically offers range of algorithms, some strong, some weak
- man-in-the middle could delete stronger algorithms from list
- last 2 steps prevent this
 - last two messages are encrypted

Real SSL: handshaking (4)

- why two random nonces?
- suppose Trudy sniffs all messages between Alice & Bob
- next day, Trudy sets up TCP connection with Bob, sends exact same sequence of records
 - Bob (Amazon) thinks Alice made two separate orders for the same thing
 - solution: Bob sends different random nonce for each connection. This causes encryption keys to be different on the two days
 - Trudy's messages will fail Bob's integrity check

SSL record protocol

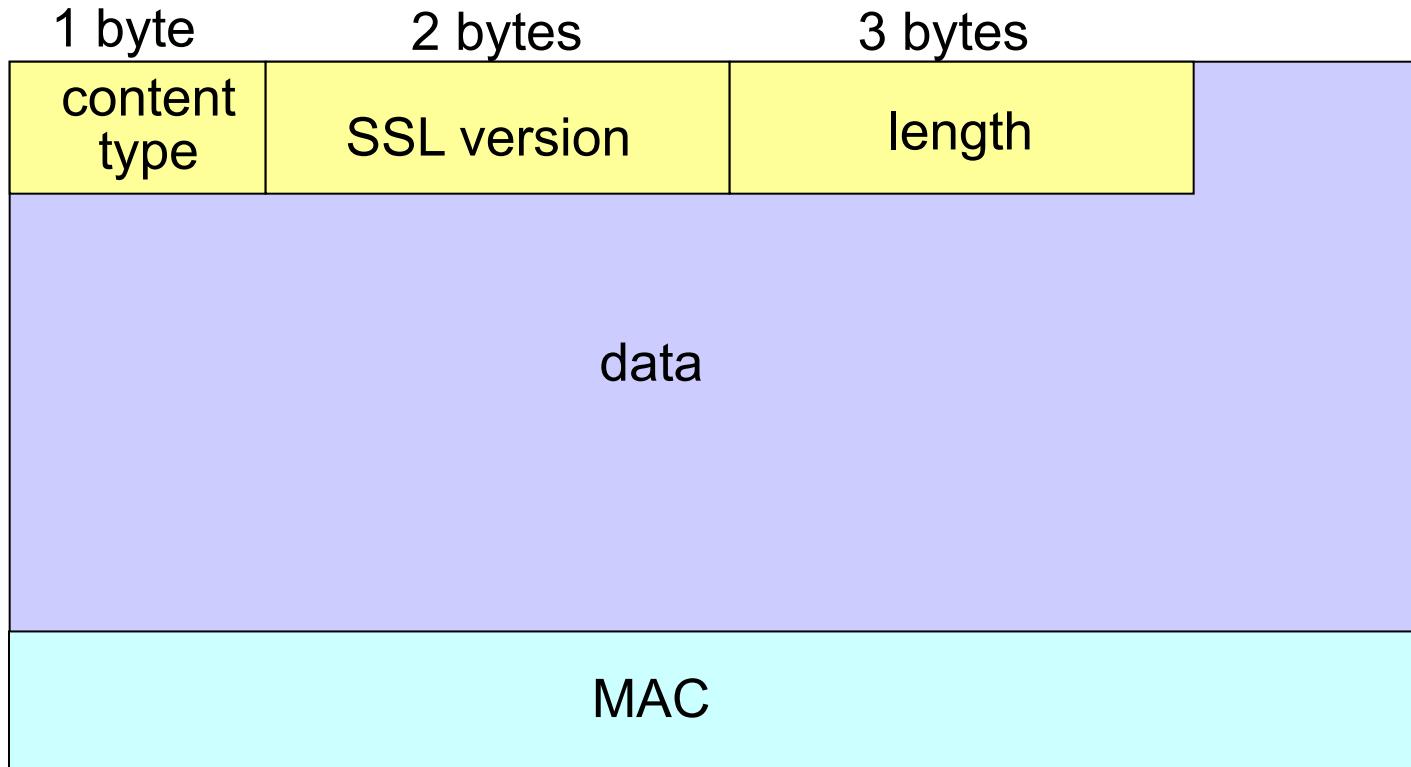


record header: content type; version; length

MAC: includes sequence number, MAC key M_x

fragment: each SSL fragment 2^{14} bytes (~ 16 Kbytes)

SSL record format

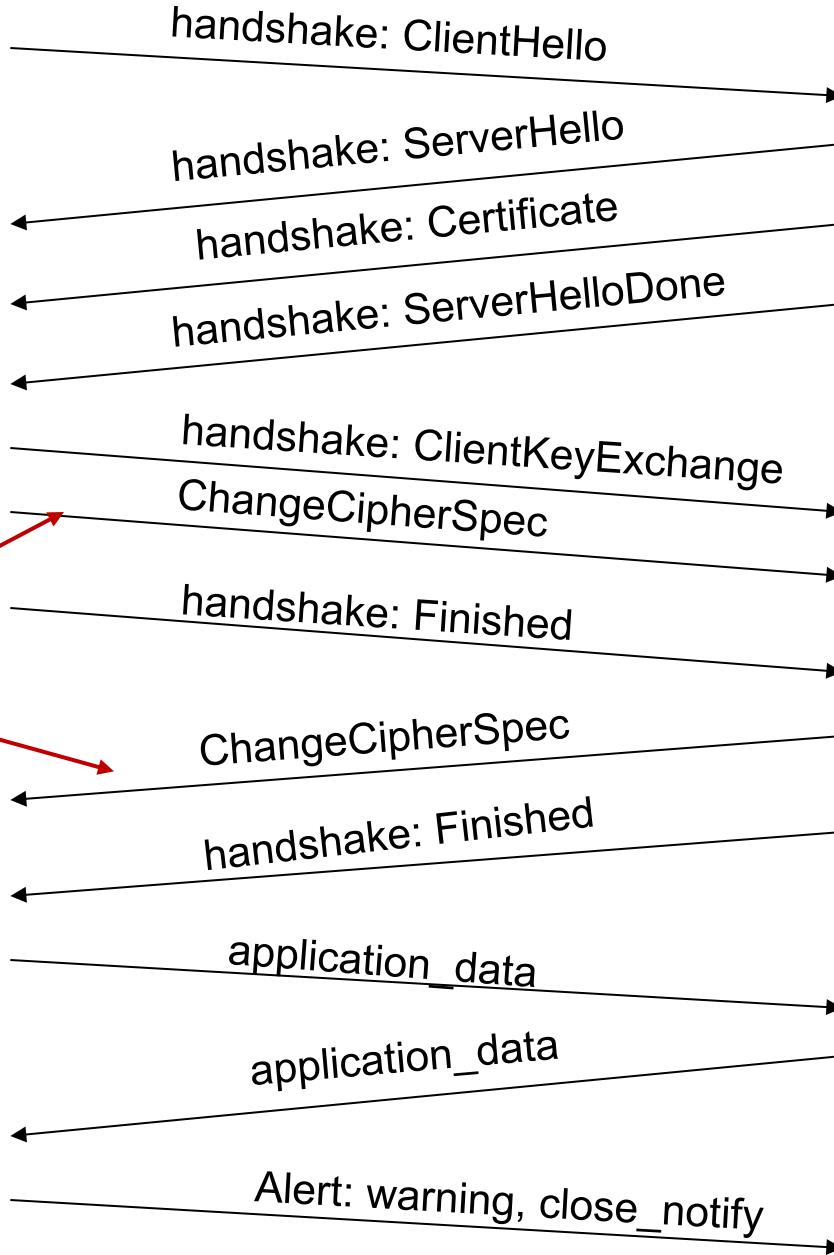


data and MAC encrypted (symmetric algorithm)

Real SSL connection

*everything
henceforth
is encrypted*

TCP FIN follows



Key derivation

- client nonce, server nonce, and pre-master secret input into pseudo random-number generator.
 - produces master secret
- master secret and new nonces input into another random-number generator: “key block”
 - because of resumption: TBD
- key block sliced and diced:
 - client MAC key
 - server MAC key
 - client encryption key
 - server encryption key
 - client initialization vector (IV)
 - server initialization vector (IV)

Chapter 8 roadmap

- 8.1 What is network security?
- 8.2 Principles of cryptography
- 8.3 Message integrity
- 8.4 Securing e-mail
- 8.5 Securing TCP connections: SSL
- 8.6 *Network layer security: IPsec*
- 8.7 Securing wireless LANs
- 8.8 Operational security: firewalls and IDS

What is network-layer confidentiality ?

between two network entities:

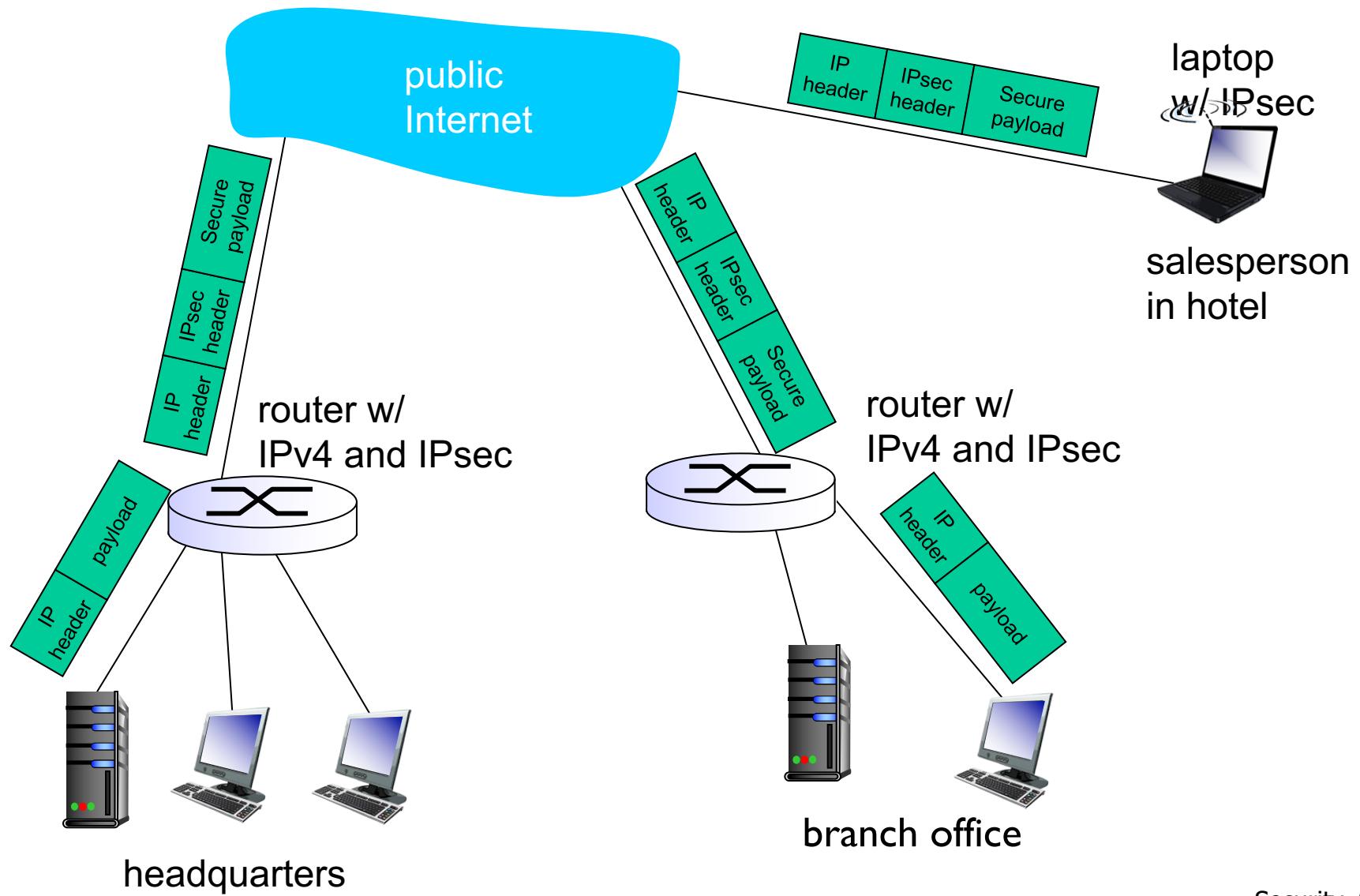
- sending entity encrypts datagram payload, payload could be:
 - TCP or UDP segment, ICMP message, OSPF message
- all data sent from one entity to other would be hidden:
 - web pages, e-mail, P2P file transfers, TCP SYN packets
 - ...
- “**blanket coverage**”

Virtual Private Networks (VPNs)

motivation:

- institutions often want private networks for security.
 - costly: separate routers, links, DNS infrastructure.
- VPN: institution's inter-office traffic is sent over public Internet instead
 - encrypted before entering public Internet
 - logically separate from other traffic

Virtual Private Networks (VPNs)

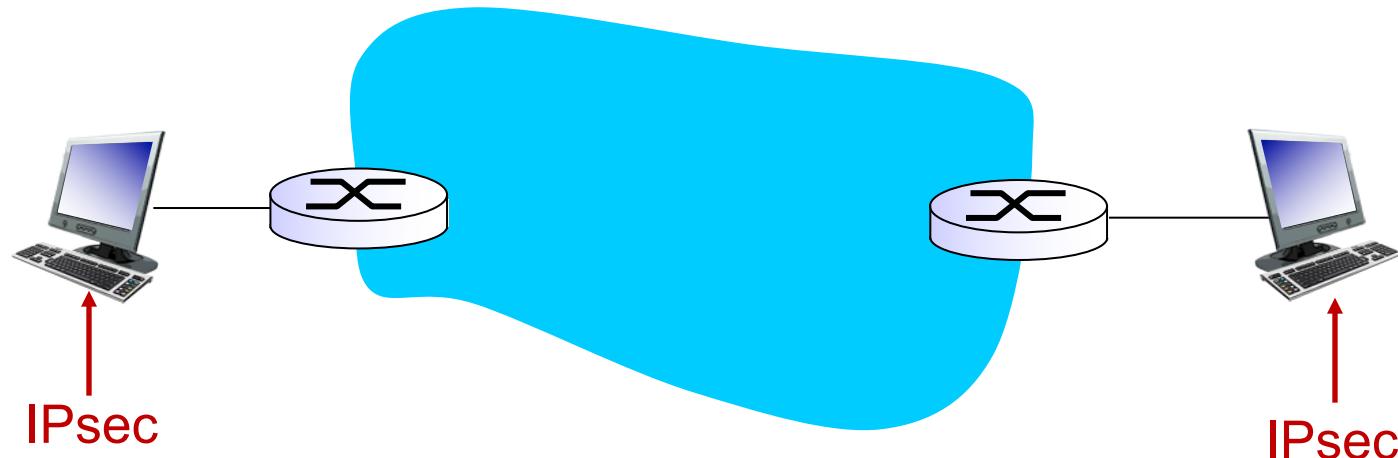


IPsec services

- data integrity
- origin authentication
- replay attack prevention
- confidentiality

- two protocols providing different service models:
 - AH
 - ESP

IPsec transport mode



- IPsec datagram emitted and received by end-system
- protects upper level protocols

IPsec – tunneling mode



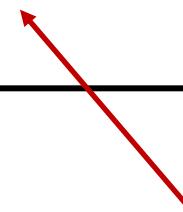
- edge routers IPsec-aware
- hosts IPsec-aware

Two IPsec protocols

- Authentication Header (AH) protocol
 - provides source authentication & data integrity but *not* confidentiality
- Encapsulation Security Protocol (ESP)
 - provides source authentication, data integrity, *and* confidentiality
 - more widely used than AH

Four combinations are possible!

Host mode with AH	Host mode with ESP
Tunnel mode with AH	Tunnel mode with ESP

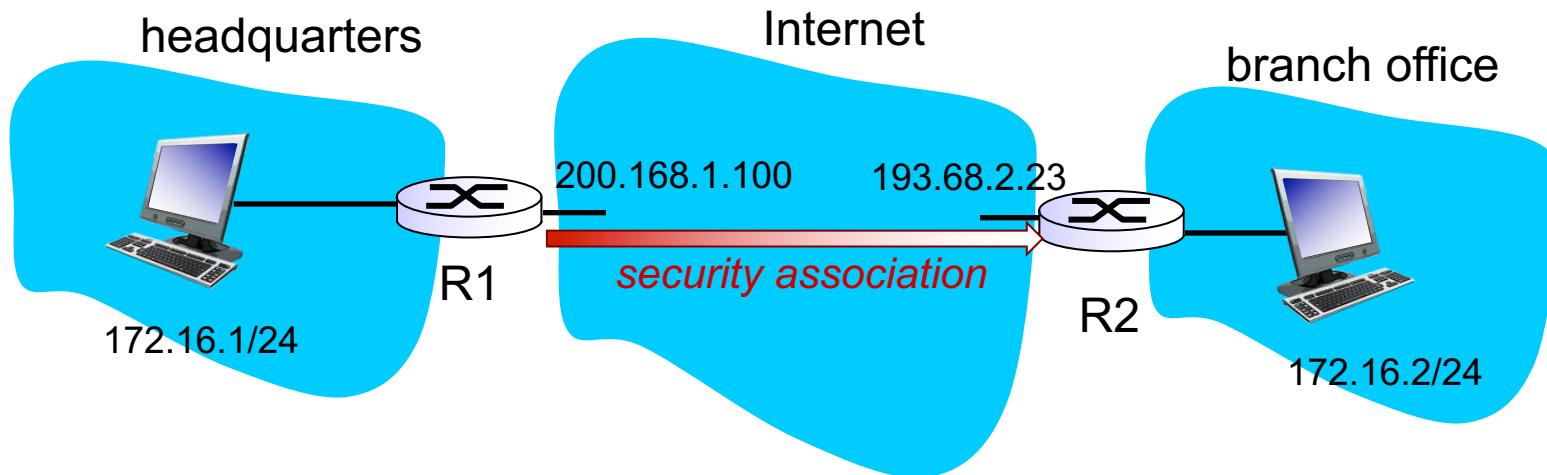


most common and
most important

Security associations (SAs)

- before sending data, “**security association (SA)**” established from sending to receiving entity
 - SAs are simplex: for only one direction
- ending, receiving entities maintain *state information* about SA
 - recall: TCP endpoints also maintain state info
 - IP is connectionless; IPsec is connection-oriented!
- how many SAs in VPN w/ headquarters, branch office, and n traveling salespeople?

Example SA from R1 to R2



R1 stores for SA:

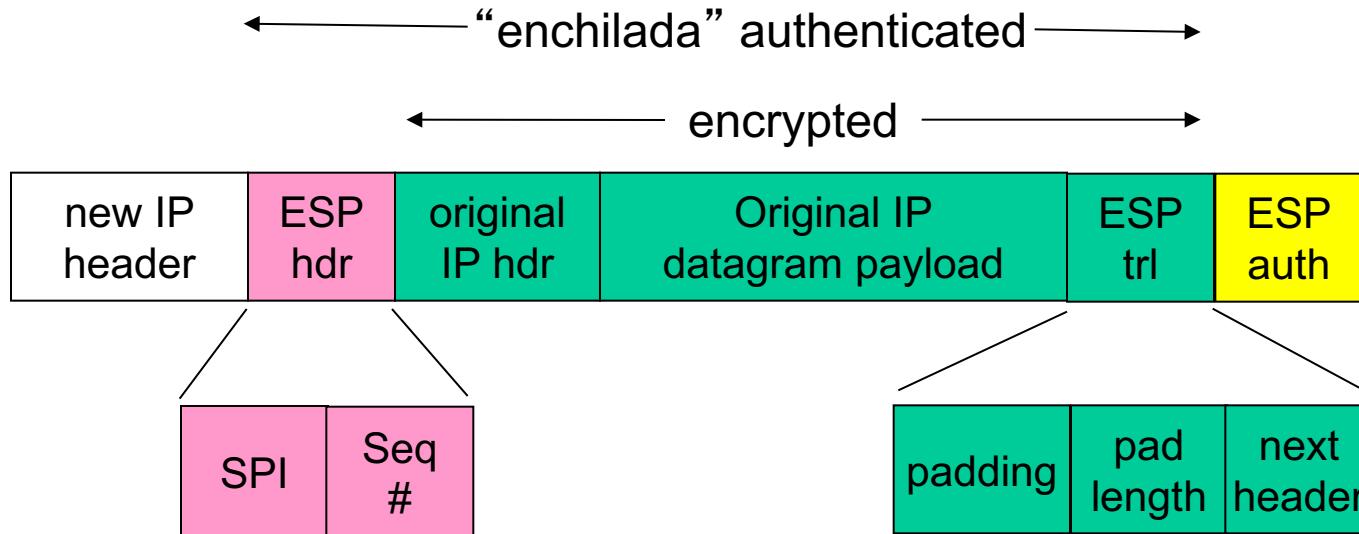
- 32-bit SA identifier: *Security Parameter Index (SPI)*
- origin SA interface (**200.168.1.100**)
- destination SA interface (**193.68.2.23**)
- type of encryption used (e.g., 3DES with CBC)
- encryption key
- type of integrity check used (e.g., HMAC with MD5)
- authentication key

Security Association Database (SAD)

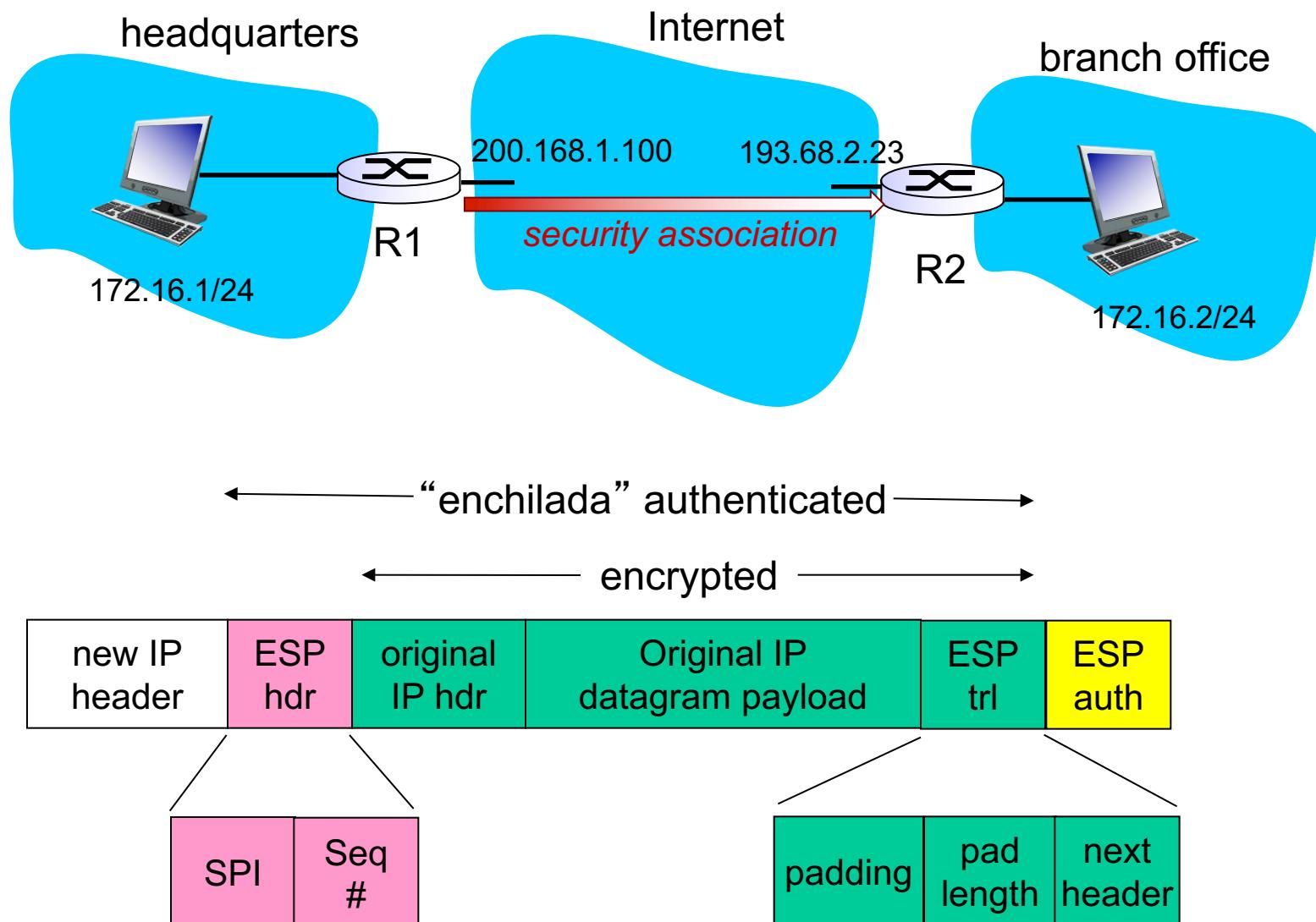
- endpoint holds SA state in *security association database (SAD)*, where it can locate them during processing.
- with n salespersons, $2 + 2n$ SAs in RI's SAD
- when sending IPsec datagram, RI accesses SAD to determine how to process datagram.
- when IPsec datagram arrives to R2, R2 examines SPI in IPsec datagram, indexes SAD with SPI, and processes datagram accordingly.

IPsec datagram

focus for now on tunnel mode with ESP



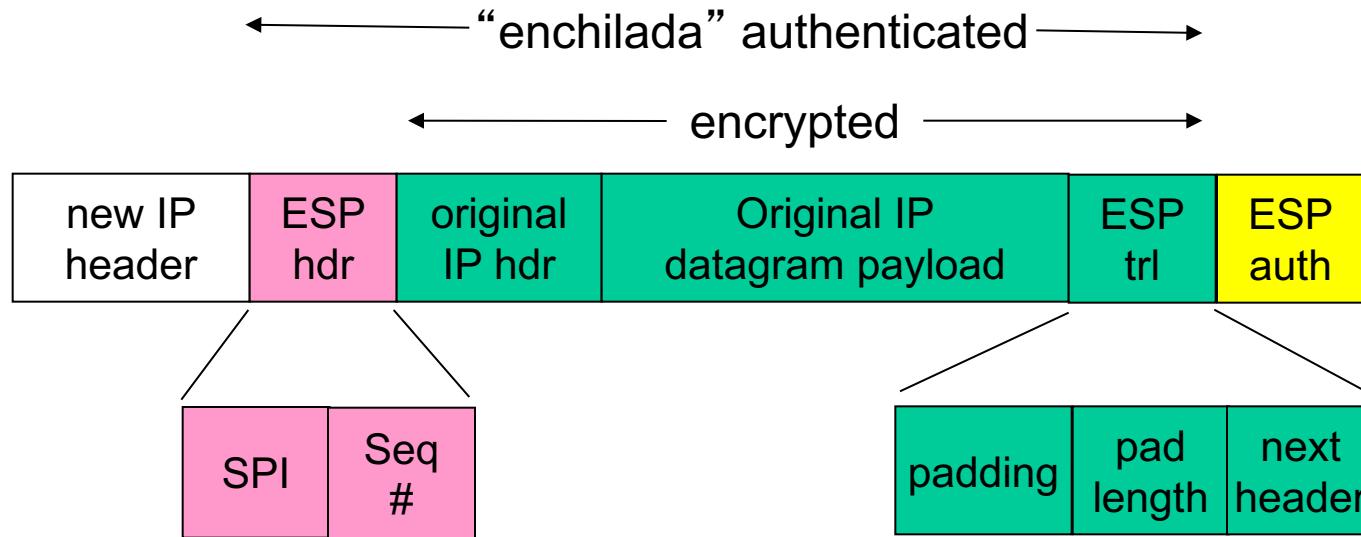
What happens?



R1: convert original datagram to IPsec datagram

- appends to back of original datagram (which includes original header fields!) an “ESP trailer” field.
- encrypts result using algorithm & key specified by SA.
- appends to front of this encrypted quantity the “ESP header, creating “enchilada”.
- creates authentication MAC over the *whole enchilada*, using algorithm and key specified in SA;
- appends MAC to back of *enchilada*, forming *payload*;
- creates brand new IP header, with all the classic IPv4 header fields, which it appends before *payload*

Inside the enchilada:



- ESP trailer: Padding for block ciphers
- ESP header:
 - SPI, so receiving entity knows what to do
 - Sequence number, to thwart replay attacks
- MAC in ESP auth field is created with shared secret key

IPsec sequence numbers

- for new SA, sender initializes seq. # to 0
- each time datagram is sent on SA:
 - sender increments seq # counter
 - places value in seq # field
- goal:
 - prevent attacker from sniffing and replaying a packet
 - receipt of duplicate, authenticated IP packets may disrupt service
- method:
 - destination checks for duplicates
 - doesn't keep track of *all* received packets; instead uses a window

Security Policy Database (SPD)

- policy: For a given datagram, sending entity needs to know if it should use IPsec
- needs also to know which SA to use
 - may use: source and destination IP address; protocol number
- info in SPD indicates “what” to do with arriving datagram
- info in SAD indicates “how” to do it

Summary: IPsec services



- suppose Trudy sits somewhere between R1 and R2. she doesn't know the keys.
 - will Trudy be able to see original contents of datagram? How about source, dest IP address, transport protocol, application port?
 - flip bits without detection?
 - masquerade as R1 using R1's IP address?
 - replay a datagram?

IKE: Internet Key Exchange

- *previous examples:* manual establishment of IPsec SAs in IPsec endpoints:

Example SA

SPI: 12345

Source IP: 200.168.1.100

Dest IP: 193.68.2.23

Protocol: ESP

Encryption algorithm: 3DES-cbc

HMAC algorithm: MD5

Encryption key: 0x7aeaca...

HMAC key: 0xc0291f...

- manual keying is impractical for VPN with 100s of endpoints
- instead use *IPsec IKE (Internet Key Exchange)*

IKE: PSK and PKI

- authentication (prove who you are) with either
 - pre-shared secret (PSK) or
 - with PKI (public/private keys and certificates).
- PSK: both sides start with secret
 - run IKE to authenticate each other and to generate IPsec SAs (one in each direction), including encryption, authentication keys
- PKI: both sides start with public/private key pair, certificate
 - run IKE to authenticate each other, obtain IPsec SAs (one in each direction).
 - similar with handshake in SSL.

IKE phases

- IKE has two phases
 - *phase 1*: establish bi-directional IKE SA
 - note: IKE SA different from IPsec SA
 - aka ISAKMP security association
 - *phase 2*: ISAKMP is used to securely negotiate IPsec pair of SAs
- phase 1 has two modes: aggressive mode and main mode
 - aggressive mode uses fewer messages
 - main mode provides identity protection and is more flexible

IPsec summary

- IKE message exchange for algorithms, secret keys, SPI numbers
- either AH or ESP protocol (or both)
 - AH provides integrity, source authentication
 - ESP protocol (with AH) additionally provides encryption
- IPsec peers can be two end systems, two routers/firewalls, or a router/firewall and an end system

Chapter 8 roadmap

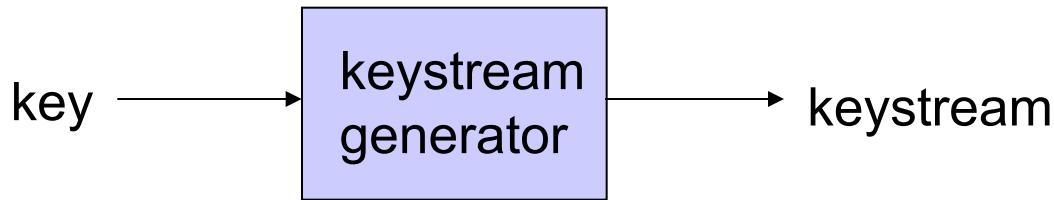
- 8.1 What is network security?
- 8.2 Principles of cryptography
- 8.3 Message integrity
- 8.4 Securing e-mail
- 8.5 Securing TCP connections: SSL
- 8.6 Network layer security: IPsec
- 8.7 *Securing wireless LANs*
- 8.8 Operational security: firewalls and IDS

WEP design goals



- symmetric key crypto
 - confidentiality
 - end host authorization
 - data integrity
- self-synchronizing: each packet separately encrypted
 - given encrypted packet and key, can decrypt; can continue to decrypt packets when preceding packet was lost (unlike Cipher Block Chaining (CBC) in block ciphers)
- Efficient
 - implementable in hardware or software

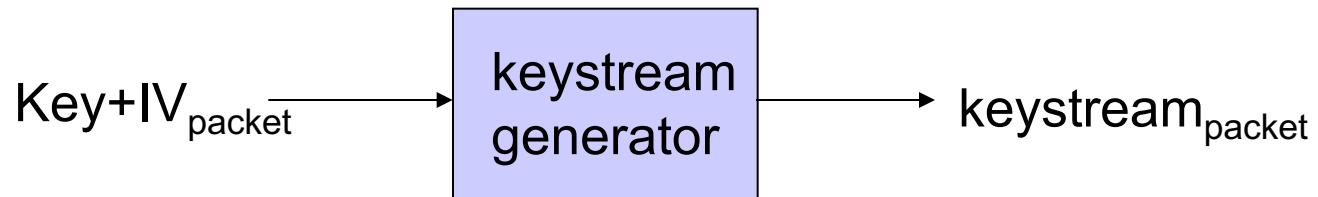
Review: symmetric stream ciphers



- *combine each byte of keystream with byte of plaintext to get ciphertext:*
 - $m(i)$ = ith unit of message
 - $ks(i)$ = ith unit of keystream
 - $c(i)$ = ith unit of ciphertext
 - $c(i) = ks(i) \oplus m(i)$ (\oplus = exclusive or)
 - $m(i) = ks(i) \oplus c(i)$
- WEP uses RC4

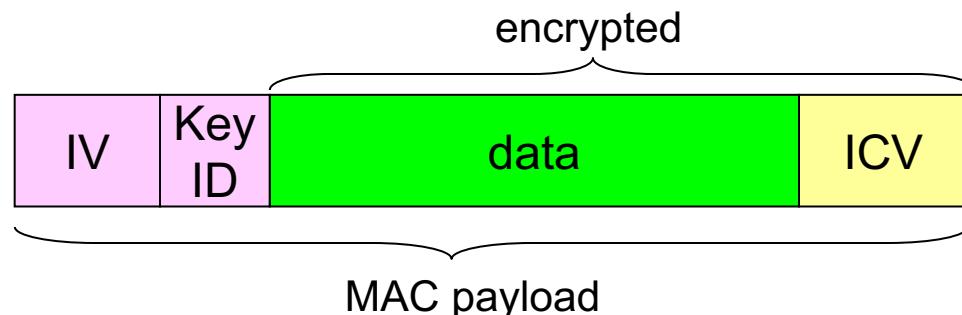
Stream cipher and packet independence

- recall design goal: each packet separately encrypted
- if for frame $n+1$, use keystream from where we left off for frame n , then each frame is not separately encrypted
 - need to know where we left off for packet n
- WEP approach: initialize keystream with key + new IV for each packet:

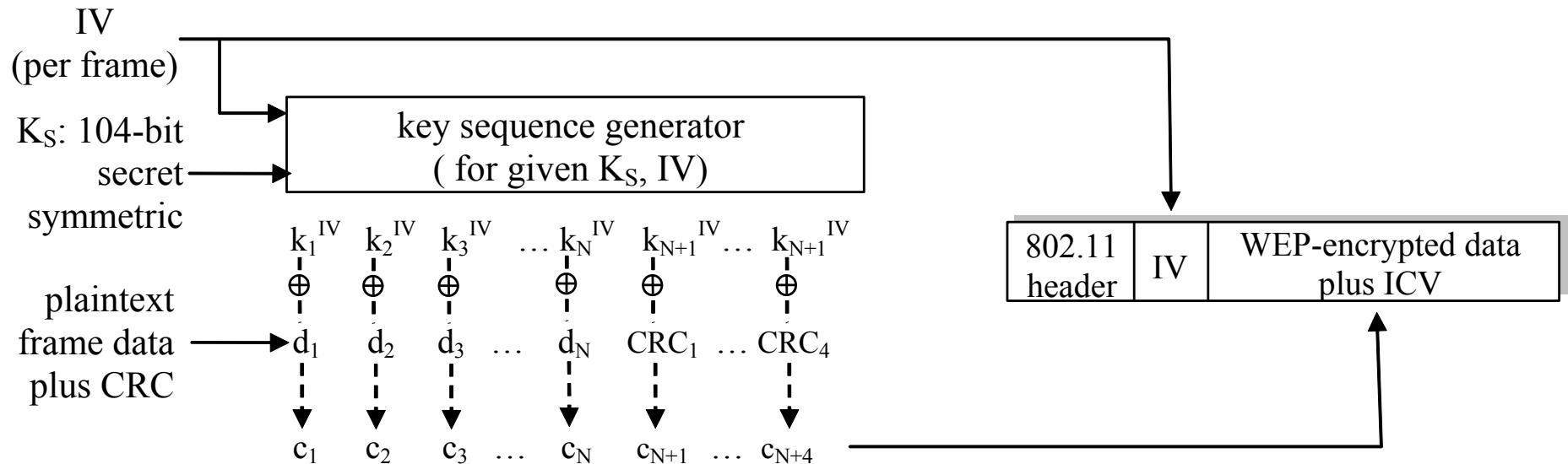


WEP encryption (I)

- sender calculates Integrity Check Value (ICV, four-byte hash/CRC over data)
- each side has 104-bit shared key
- sender creates 24-bit initialization vector (IV), appends to key: gives 128-bit key
- sender also appends keyID (in 8-bit field)
- 128-bit key inputted into pseudo random number generator to get keystream
- data in frame + ICV is encrypted with RC4:
 - bytes of keystream are XORed with bytes of data & ICV
 - IV & keyID are appended to encrypted data to create payload
 - payload inserted into 802.11 frame

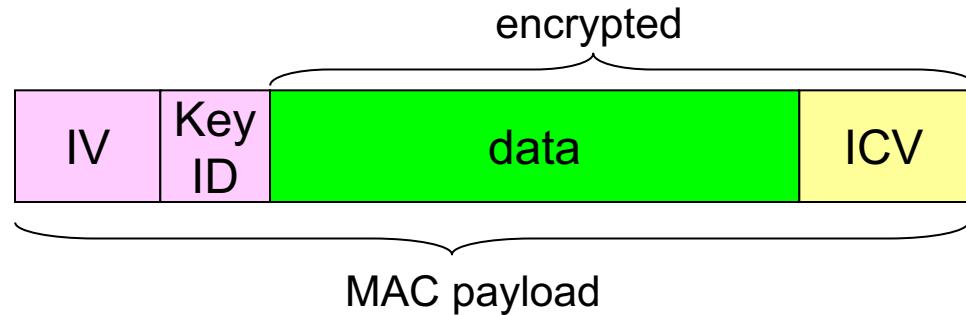


WEP encryption (2)



new IV for each frame

WEP decryption overview

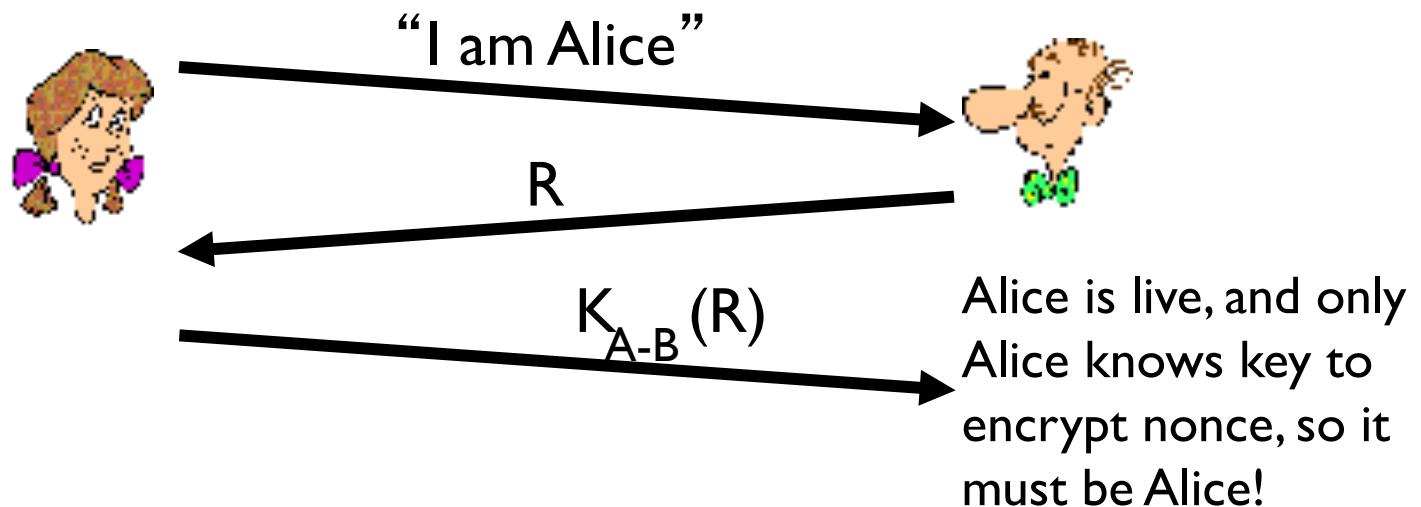


- receiver extracts IV
- inputs IV, shared secret key into pseudo random generator, gets keystream
- XORs keystream with encrypted data to decrypt data + ICV
- verifies integrity of data with ICV
 - note: message integrity approach used here is different from MAC (message authentication code) and signatures (using PKI).

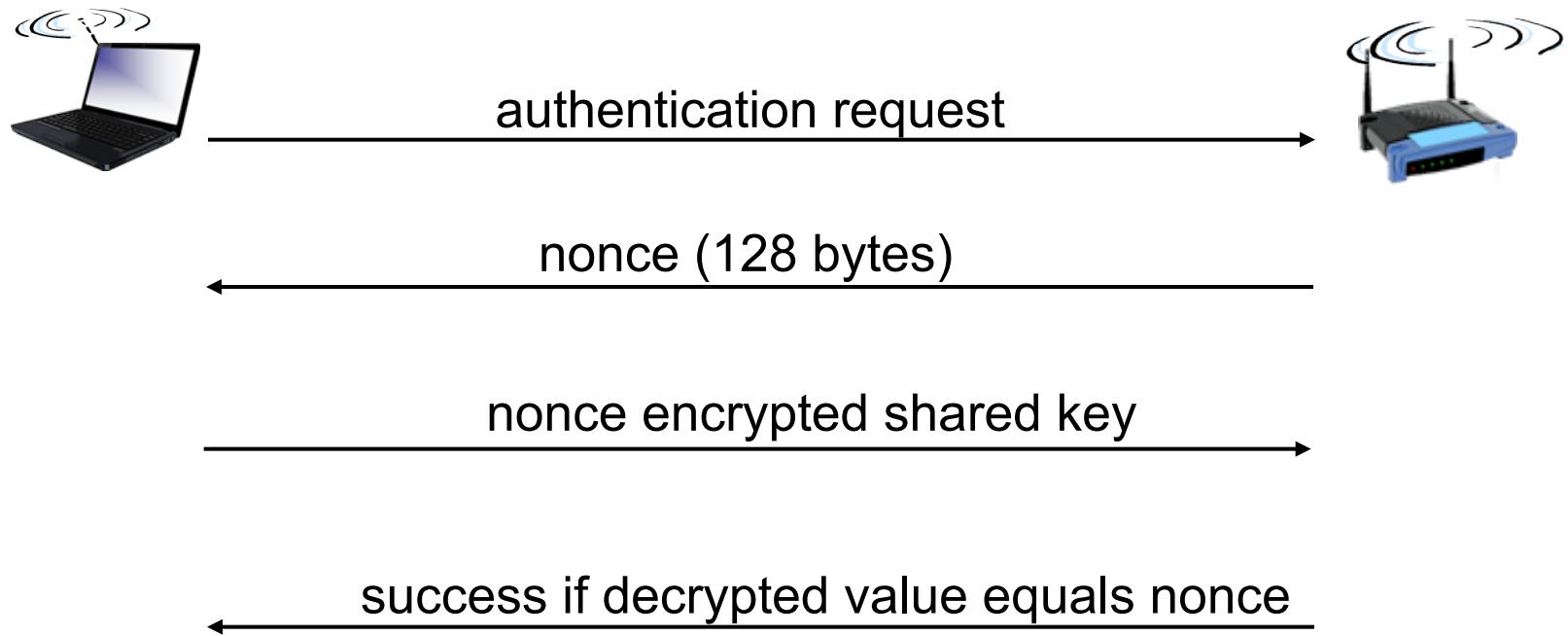
End-point authentication w/ nonce

Nonce: number (R) used only *once –in-a-lifetime*

How to prove Alice “live”: Bob sends Alice **nonce**, R . Alice must return R , encrypted with shared secret key



WEP authentication



Notes:

- not all APs do it, even if WEP is being used
- AP indicates if authentication is necessary in beacon frame
- done before association

Breaking 802.11 WEP encryption

security hole:

- 24-bit IV, one IV per frame, -> IV's eventually reused
- IV transmitted in plaintext -> IV reuse detected

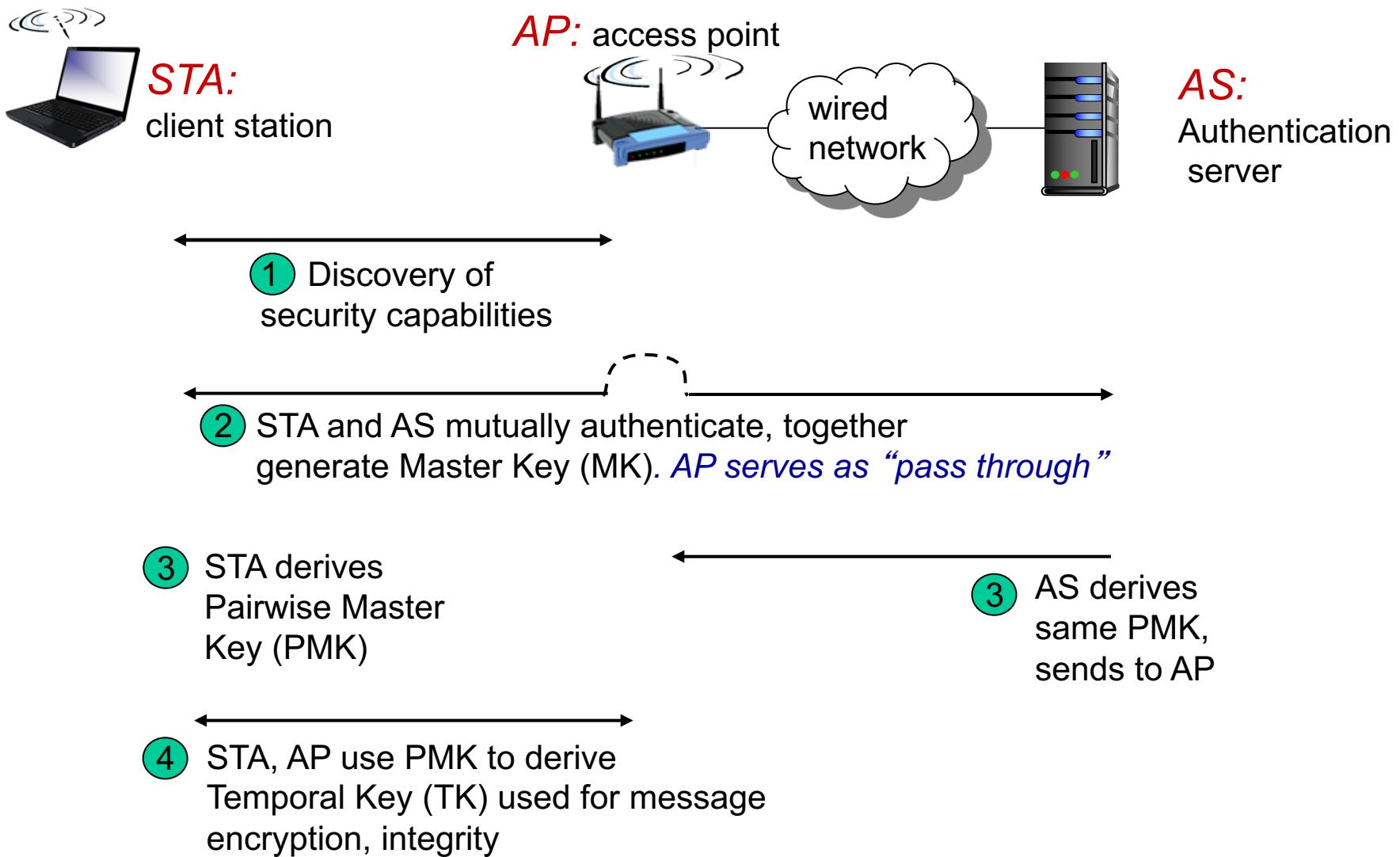
attack:

- Trudy causes Alice to encrypt known plaintext $d_1 \ d_2 \ d_3 \ d_4$
...
- Trudy sees: $c_i = d_i \text{ XOR } k_i^{\text{IV}}$
- Trudy knows $c_i \ d_i$, so can compute k_i^{IV}
- Trudy knows encrypting key sequence $k_1^{\text{IV}} \ k_2^{\text{IV}} \ k_3^{\text{IV}} \dots$
- Next time IV is used, Trudy can decrypt!

802.11i: improved security

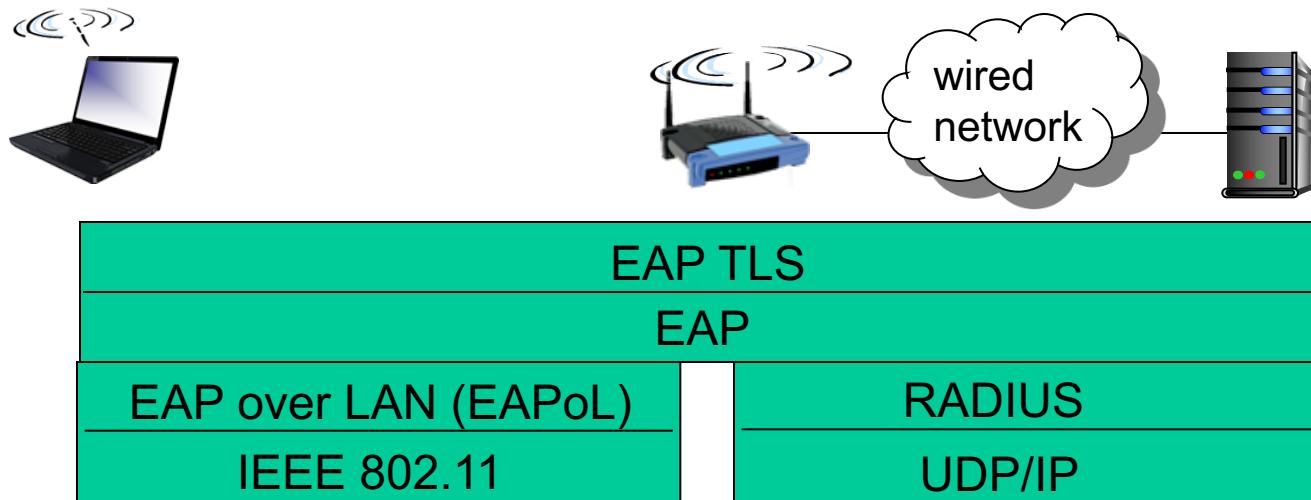
- numerous (stronger) forms of encryption possible
- provides key distribution
- uses authentication server separate from access point

802.11i: four phases of operation



EAP: extensible authentication protocol

- EAP: end-end client (mobile) to authentication server protocol
- EAP sent over separate “links”
 - mobile-to-AP (EAP over LAN)
 - AP to authentication server (RADIUS over UDP)



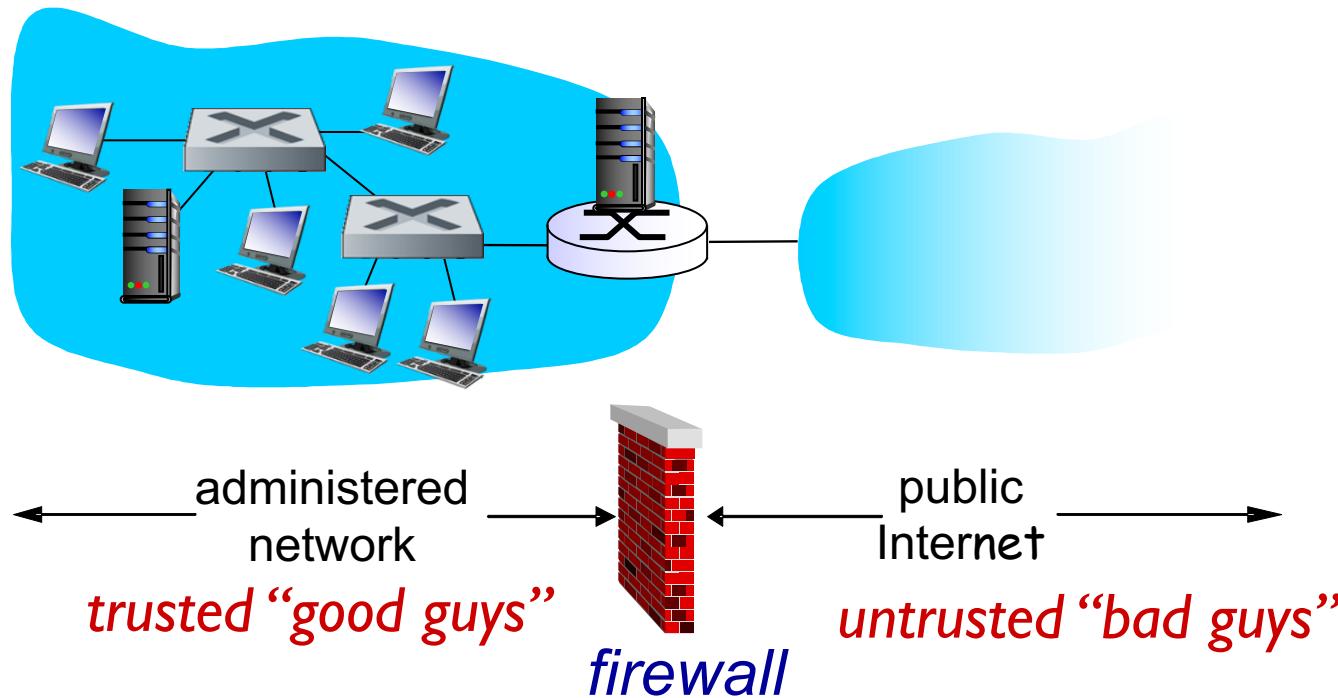
Chapter 8 roadmap

- 8.1 What is network security?
- 8.2 Principles of cryptography
- 8.3 Message integrity
- 8.4 Securing e-mail
- 8.5 Securing TCP connections: SSL
- 8.6 Network layer security: IPsec
- 8.7 Securing wireless LANs
- 8.8 *Operational security: firewalls and IDS*

Firewalls

firewall

isolates organization's internal net from larger Internet,
allowing some packets to pass, blocking others



Firewalls: why

prevent denial of service attacks:

- SYN flooding: attacker establishes many bogus TCP connections, no resources left for “real” connections

prevent illegal modification/access of internal data

- e.g., attacker replaces CIA’s homepage with something else

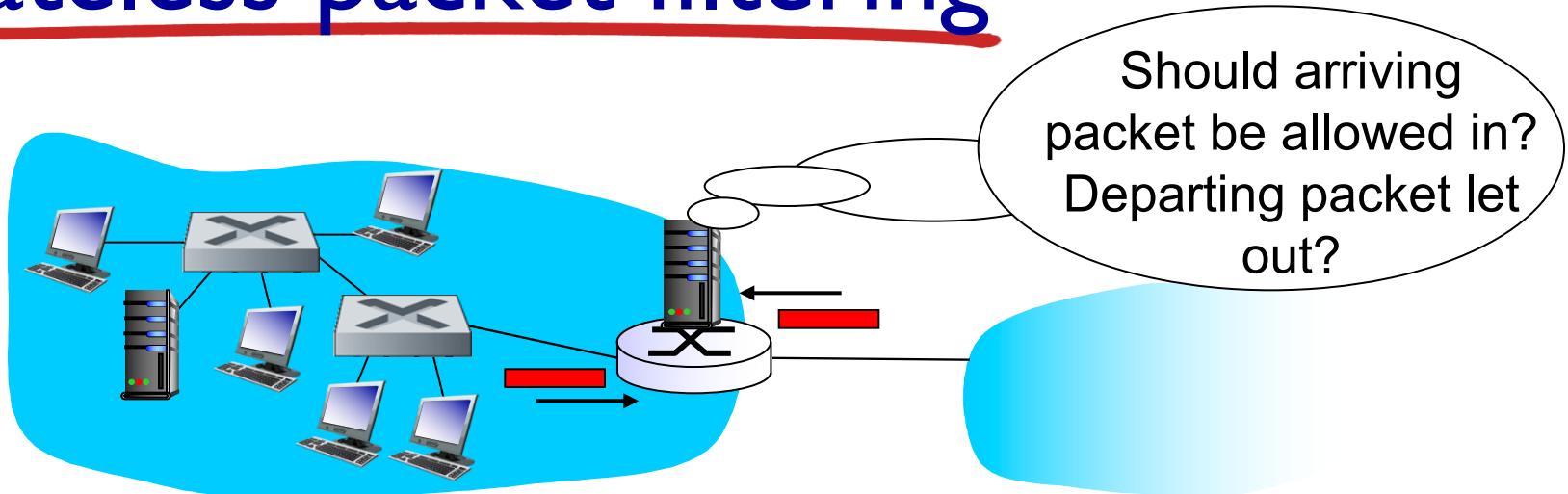
allow only authorized access to inside network

- set of authenticated users/hosts

three types of firewalls:

- stateless packet filters
- stateful packet filters
- application gateways

Stateless packet filtering



- internal network connected to Internet via *router firewall*
- router *filters packet-by-packet*, decision to forward/drop packet based on:
 - source IP address, destination IP address
 - TCP/UDP source and destination port numbers
 - ICMP message type
 - TCP SYN and ACK bits

Stateless packet filtering: example

- *example 1:* block incoming and outgoing datagrams with IP protocol field = 17 and with either source or dest port = 23
 - *result:* all incoming, outgoing UDP flows and telnet connections are blocked
- *example 2:* block inbound TCP segments with ACK=0.
 - *result:* prevents external clients from making TCP connections with internal clients, but allows internal clients to connect to outside.

Stateless packet filtering: more examples

<i>Policy</i>	<i>Firewall Setting</i>
No outside Web access.	Drop all outgoing packets to any IP address, port 80
No incoming TCP connections, except those for institution's public Web server only.	Drop all incoming TCP SYN packets to any IP except 130.207.244.203, port 80
Prevent Web-radios from eating up the available bandwidth.	Drop all incoming UDP packets - except DNS and router broadcasts.
Prevent your network from being used for a smurf DoS attack.	Drop all ICMP packets going to a "broadcast" address (e.g. 130.207.255.255).
Prevent your network from being tracerouted	Drop all outgoing ICMP TTL expired traffic

Access Control Lists

ACL: table of rules, applied top to bottom to incoming packets:
(action, condition) pairs: looks like OpenFlow forwarding (Ch. 4)!

action	source address	dest address	protocol	source port	dest port	flag bit
allow	222.22/16	outside of 222.22/16	TCP	> 1023	80	any
allow	outside of 222.22/16	222.22/16	TCP	80	> 1023	ACK
allow	222.22/16	outside of 222.22/16	UDP	> 1023	53	---
allow	outside of 222.22/16	222.22/16	UDP	53	> 1023	----
deny	all	all	all	all	all	all

Stateful packet filtering

- *stateless packet filter*: heavy handed tool
 - admits packets that “make no sense,” e.g., dest port = 80, ACK bit set, even though no TCP connection established:

action	source address	dest address	protocol	source port	dest port	flag bit
allow	outside of 222.22/16	222.22/16	TCP	80	> 1023	ACK

- *stateful packet filter*: track status of every TCP connection
 - track connection setup (SYN), teardown (FIN): determine whether incoming, outgoing packets “makes sense”
 - timeout inactive connections at firewall: no longer admit packets

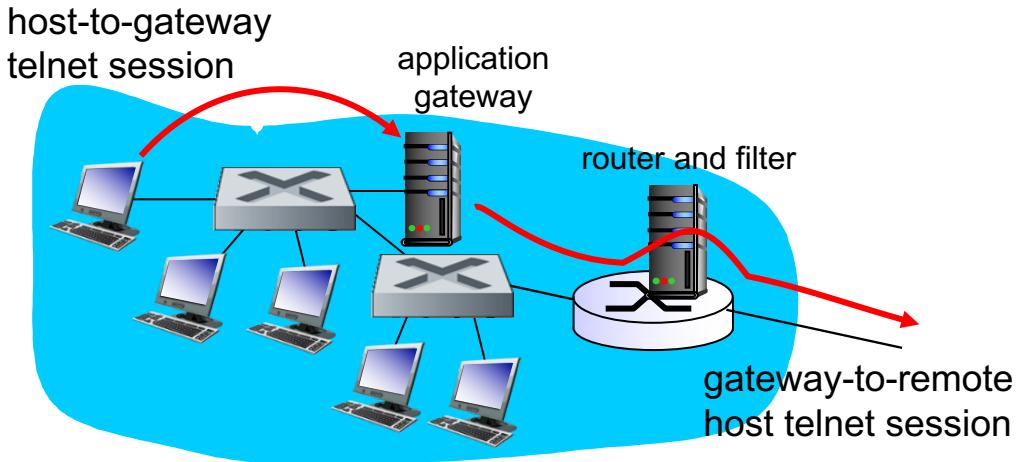
Stateful packet filtering

ACL augmented to indicate need to check connection state table before admitting packet

action	source address	dest address	proto	source port	dest port	flag bit	check connxion
allow	222.22/16	outside of 222.22/16	TCP	> 1023	80	any	
allow	outside of 222.22/16	222.22/16	TCP	80	> 1023	ACK	X
allow	222.22/16	outside of 222.22/16	UDP	> 1023	53	---	
allow	outside of 222.22/16	222.22/16	UDP	53	> 1023	----	X
deny	all	all	all	all	all	all	

Application gateways

- filter packets on application data as well as on IP/TCP/UDP fields.
- *example:* allow select internal users to telnet outside



1. require all telnet users to telnet through gateway.
2. for authorized users, gateway sets up telnet connection to dest host. Gateway relays data between 2 connections
3. router filter blocks all telnet connections not originating from gateway.

Limitations of firewalls, gateways

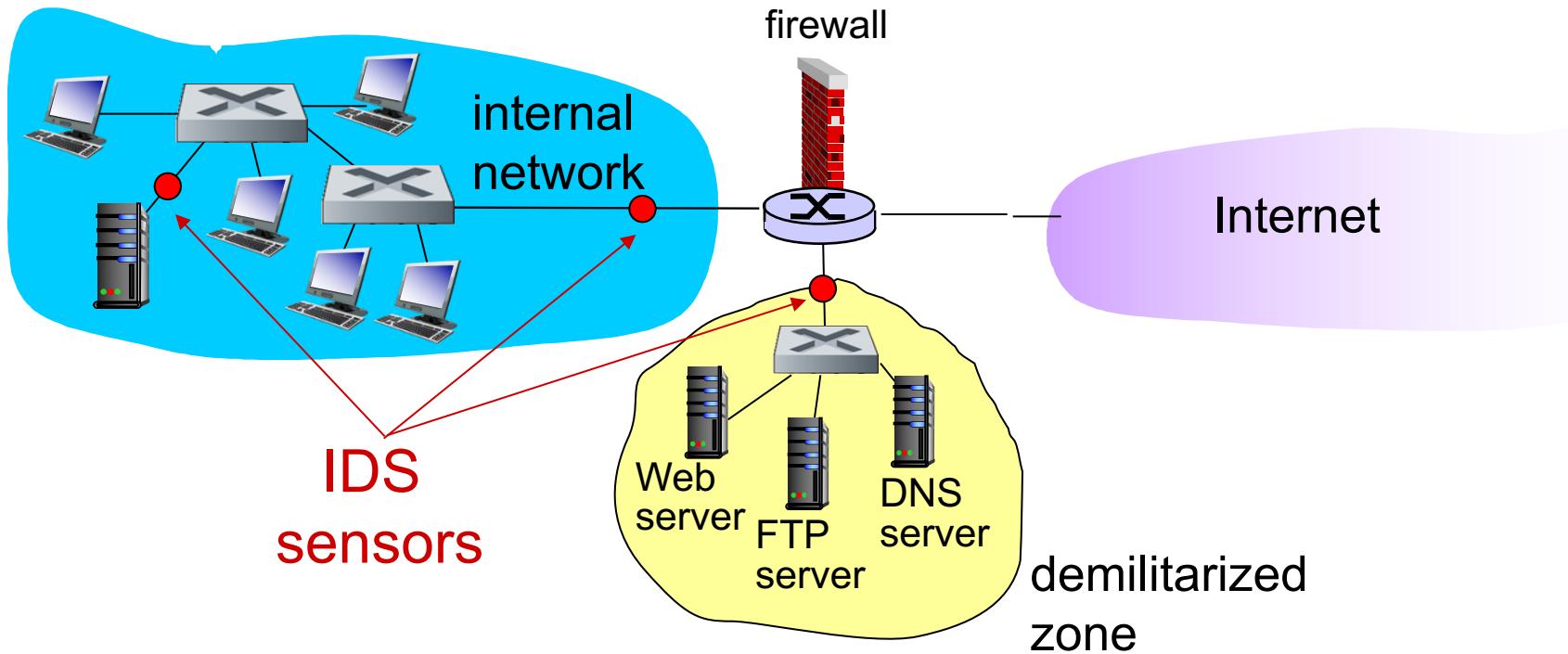
- *IP spoofing*: router can't know if data "really" comes from claimed source
- if multiple app's. need special treatment, each has own app. gateway
- client software must know how to contact gateway.
 - e.g., must set IP address of proxy in Web browser
- filters often use all or nothing policy for UDP
- *tradeoff*: degree of communication with outside world, level of security
- many highly protected sites still suffer from attacks

Intrusion detection systems

- packet filtering:
 - operates on TCP/IP headers only
 - no correlation check among sessions
- *IDS: intrusion detection system*
 - *deep packet inspection*: look at packet contents (e.g., check character strings in packet against database of known virus, attack strings)
 - **examine correlation** among multiple packets
 - port scanning
 - network mapping
 - DoS attack

Intrusion detection systems

multiple IDSs: different types of checking at different locations



Network Security (summary)

basic techniques.....

- cryptography (symmetric and public)
- message integrity
- end-point authentication

.... used in many different security scenarios

- secure email
- secure transport (SSL)
- IP sec
- 802.11

operational security: firewalls and IDS

Chapter I

Introduction

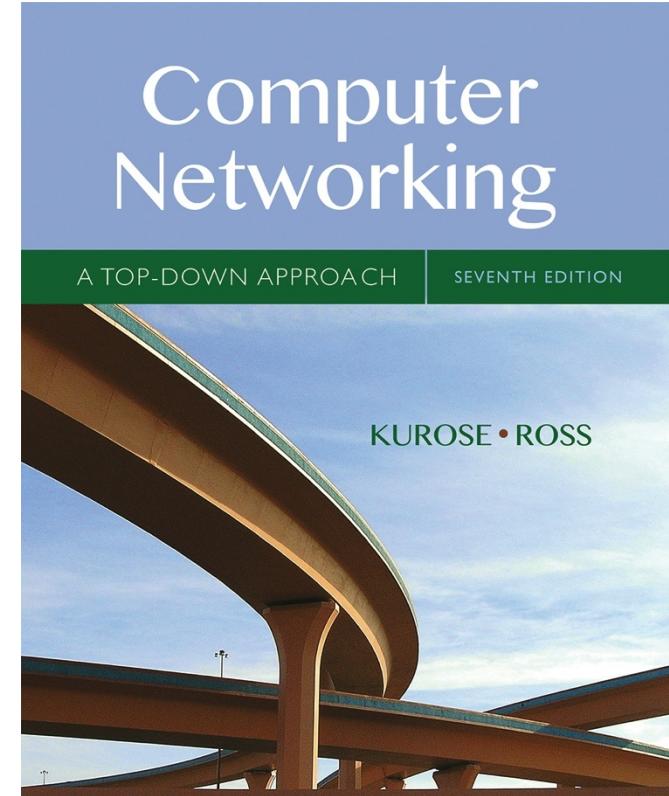
A note on the use of these Powerpoint slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR

©All material copyright 1996-2016
J.F Kurose and K.W. Ross, All Rights Reserved



*Computer
Networking: A Top
Down Approach*

7th edition
Jim Kurose, Keith Ross
Pearson/Addison Wesley
April 2016

Chapter I: introduction

our goal:

- get “feel” and terminology
- more depth, detail
later in course
- approach:
 - use Internet as example

overview:

- what’s the Internet?
- what’s a protocol?
- network edge; hosts, access net, physical media
- network core: packet/circuit switching, Internet structure
- performance: loss, delay, throughput
- security
- protocol layers, service models
- history

Chapter I: roadmap

I.1 what *is* the Internet?

I.2 network edge

- end systems, access networks, links

I.3 network core

- packet switching, circuit switching, network structure

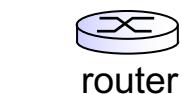
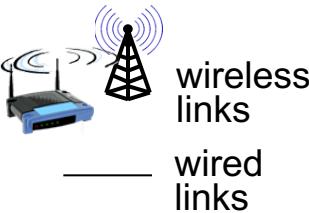
I.4 delay, loss, throughput in networks

I.5 protocol layers, service models

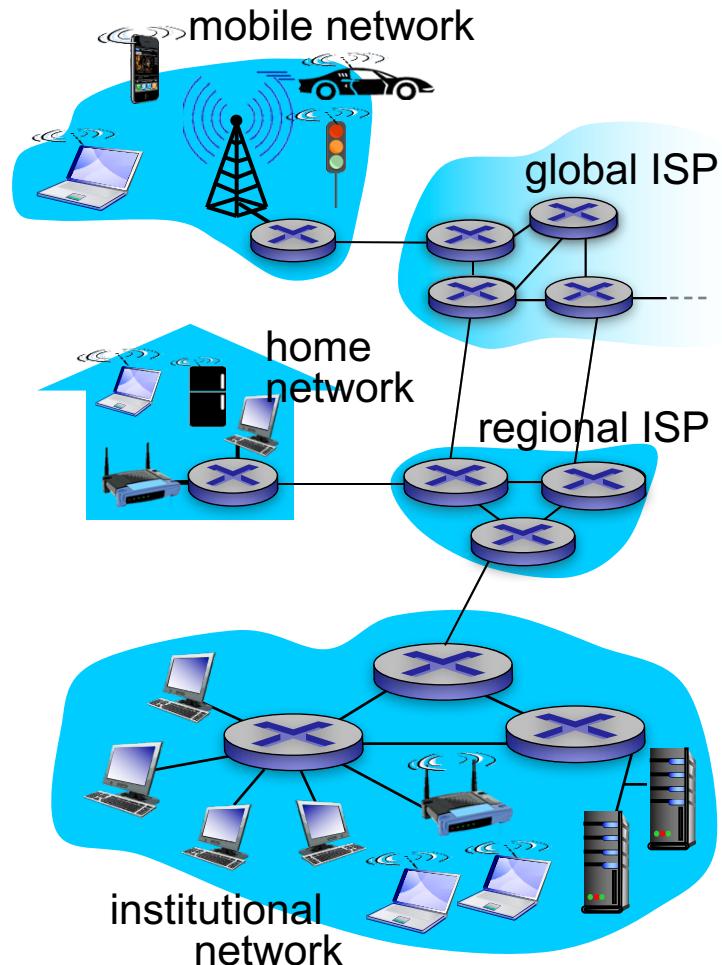
I.6 networks under attack: security

I.7 history

What's the Internet: “nuts and bolts” view



- billions of connected computing devices:
 - *hosts* = *end systems*
 - running *network apps*
- *communication links*
 - fiber, copper, radio, satellite
 - transmission rate: *bandwidth*
- *packet switches*: forward packets (chunks of data)
 - *routers and switches*



“Fun” Internet-connected devices



IP picture frame
<http://www.ceiva.com/>



Web-enabled toaster +
weather forecaster



Internet
refrigerator



Slingbox: watch,
control cable TV remotely



sensorized,
bed
mattress



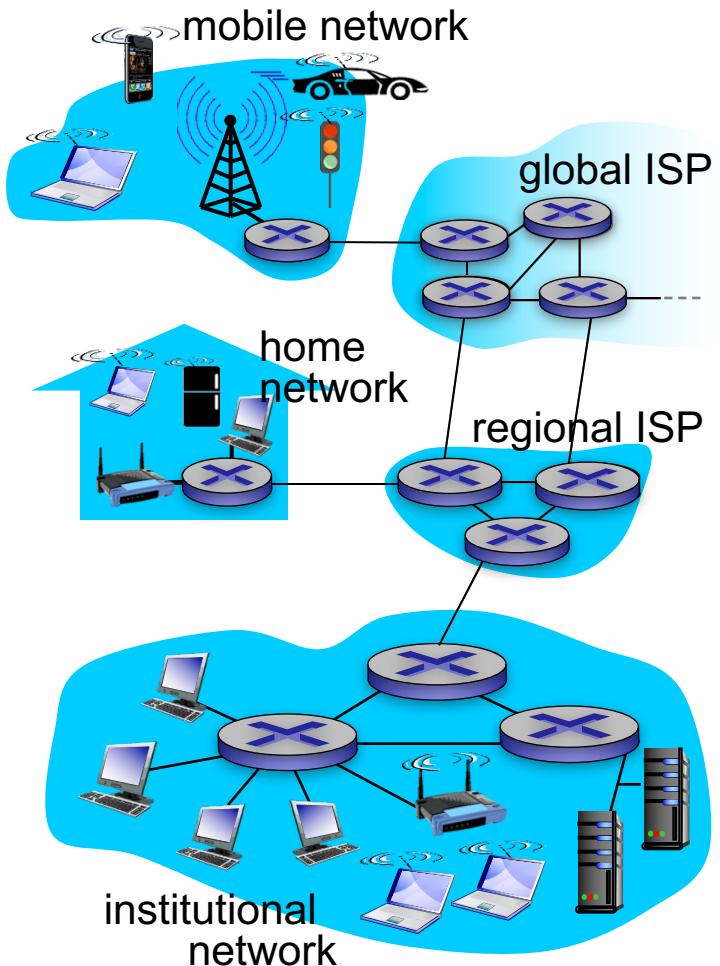
Tweet-a-watt:
monitor energy use



Internet phones

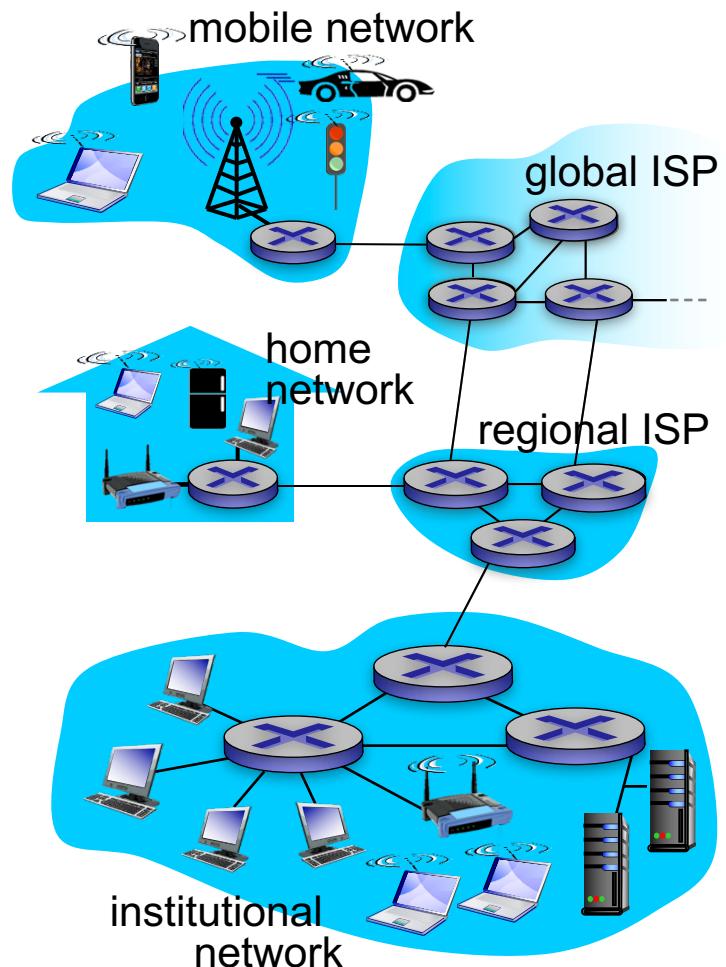
What's the Internet: “nuts and bolts” view

- *Internet: “network of networks”*
 - Interconnected ISPs
- *protocols* control sending, receiving of messages
 - e.g., TCP, IP, HTTP, Skype, 802.11
- *Internet standards*
 - RFC: Request for comments
 - IETF: Internet Engineering Task Force



What's the Internet: a service view

- *infrastructure that provides services to applications:*
 - Web, VoIP, email, games, e-commerce, social nets, ...
- *provides programming interface to apps*
 - hooks that allow sending and receiving app programs to “connect” to Internet
 - provides service options, analogous to postal service



What's a protocol?

human protocols:

- “what’s the time?”
- “I have a question”
- introductions

... specific messages sent

... specific actions taken
when messages
received, or other
events

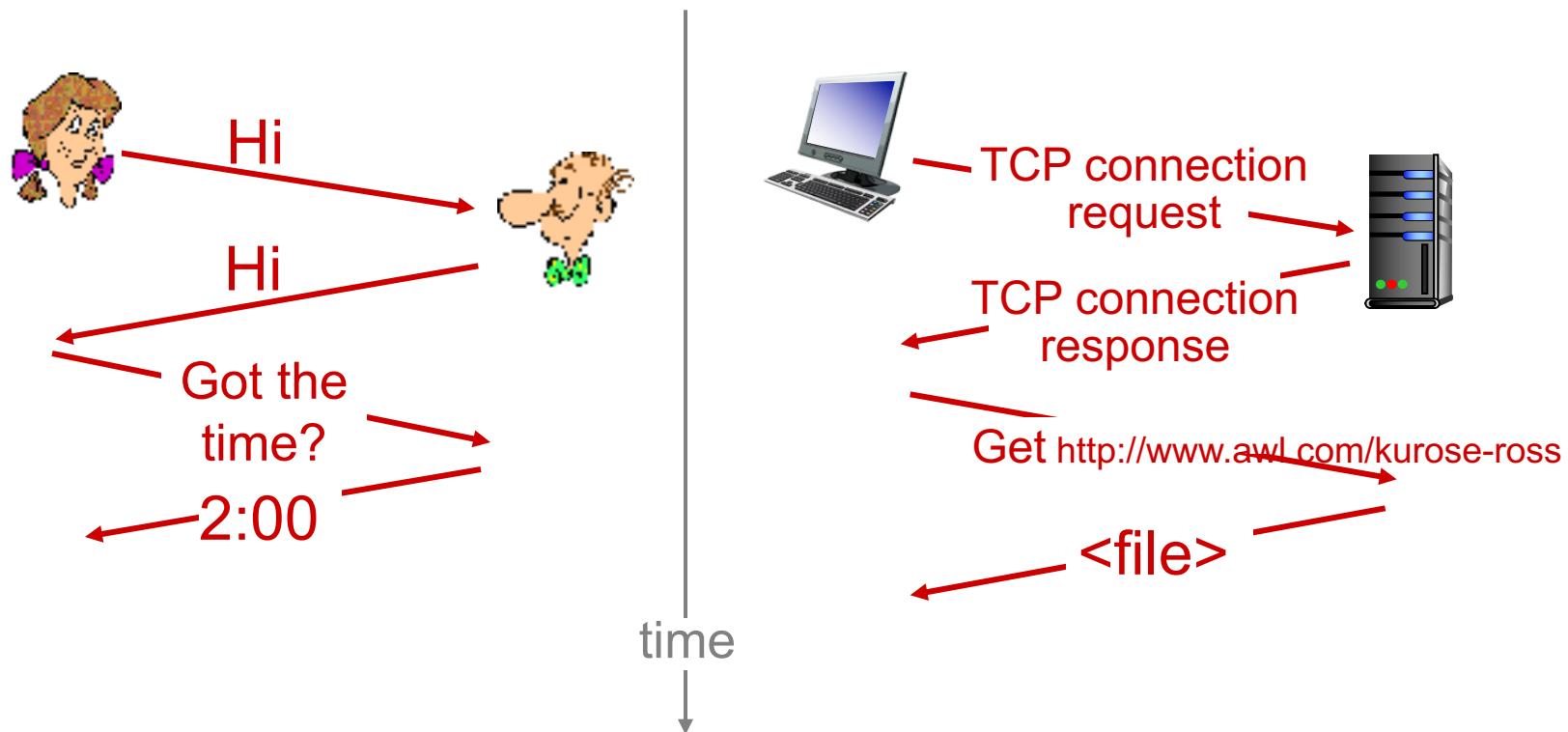
network protocols:

- machines rather than humans
- all communication activity in Internet governed by protocols

protocols define format, order of messages sent and received among network entities, and actions taken on message transmission, receipt

What's a protocol?

a human protocol and a computer network protocol:



Q: other human protocols?

Chapter I: roadmap

I.1 what is the Internet?

I.2 network edge

- end systems, access networks, links

I.3 network core

- packet switching, circuit switching, network structure

I.4 delay, loss, throughput in networks

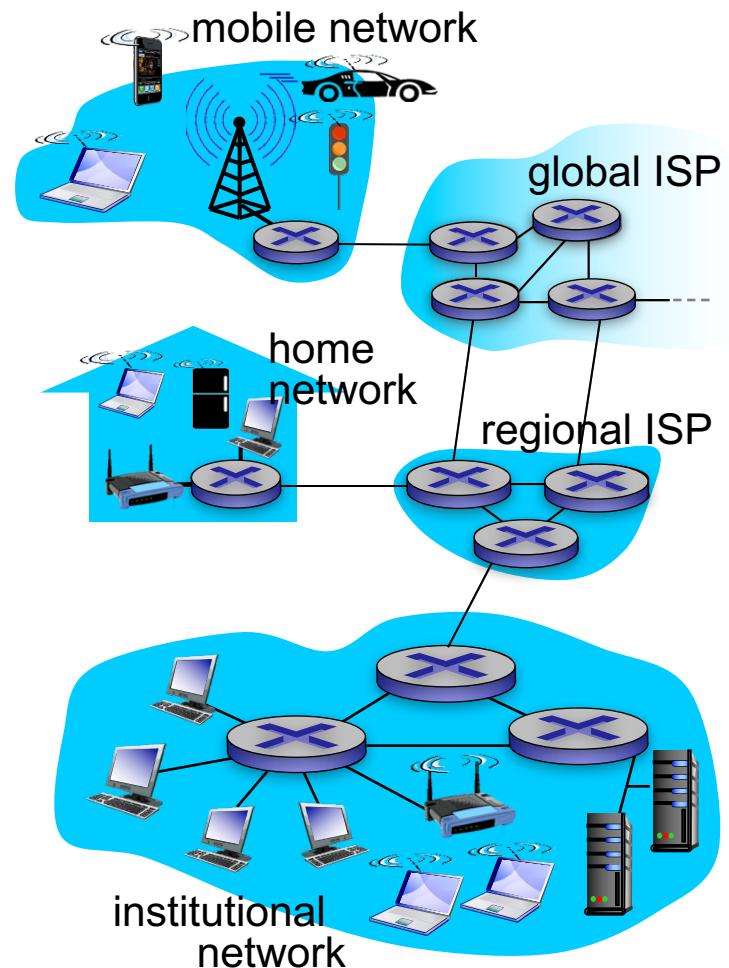
I.5 protocol layers, service models

I.6 networks under attack: security

I.7 history

A closer look at network structure:

- *network edge:*
 - hosts: clients and servers
 - servers often in data centers
- *access networks, physical media:* wired, wireless communication links
- *network core:*
 - interconnected routers
 - network of networks



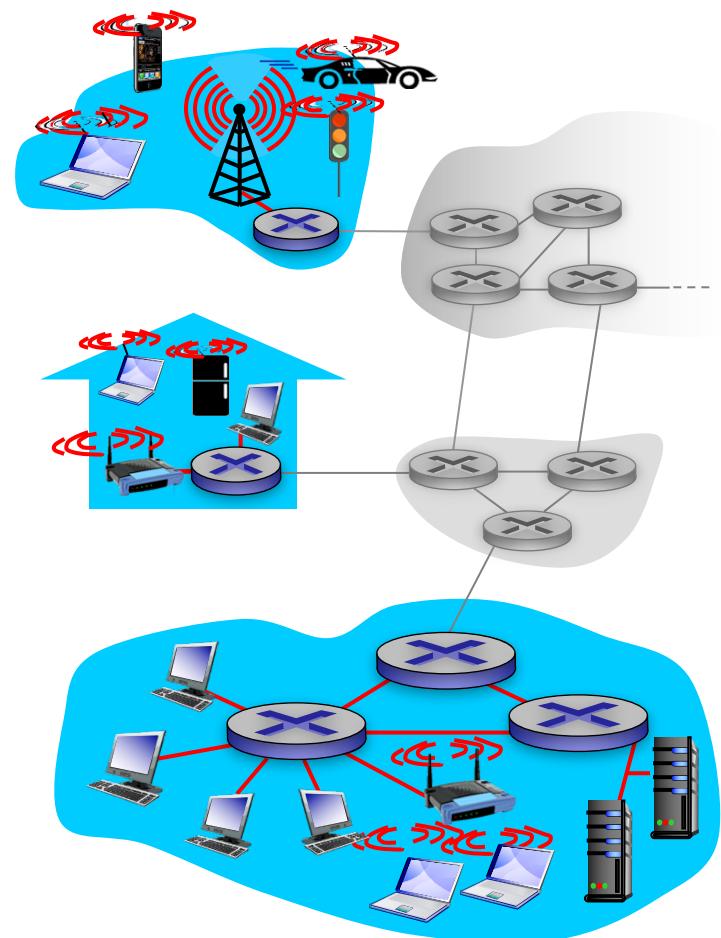
Access networks and physical media

Q: How to connect end systems to edge router?

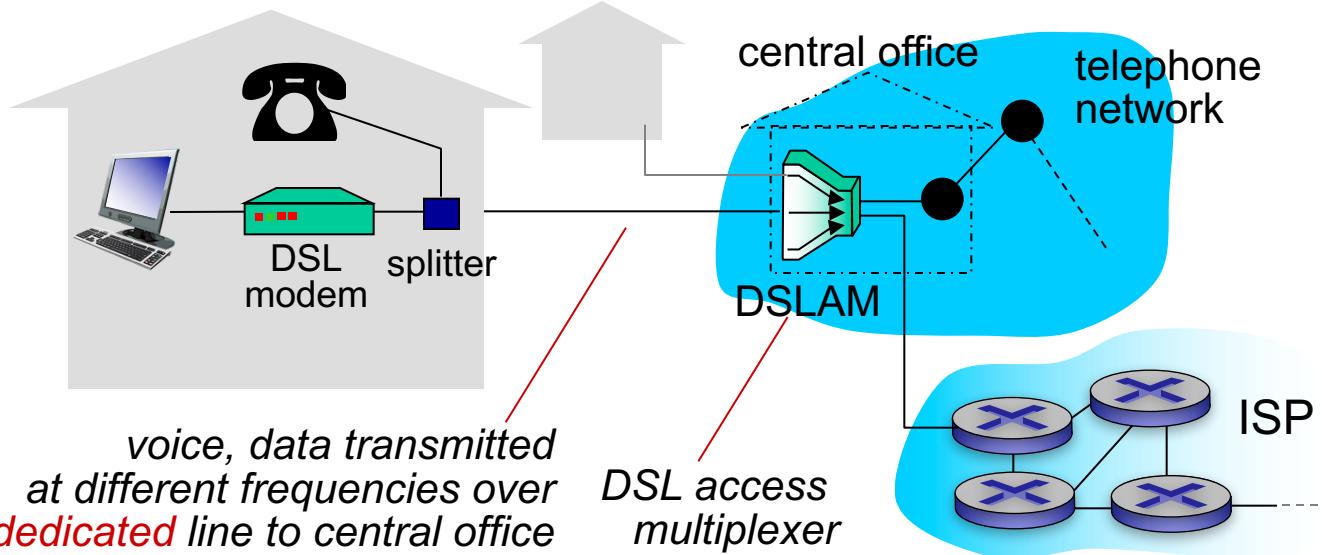
- residential access nets
- institutional access networks (school, company)
- mobile access networks

keep in mind:

- bandwidth (bits per second) of access network?
- shared or dedicated?

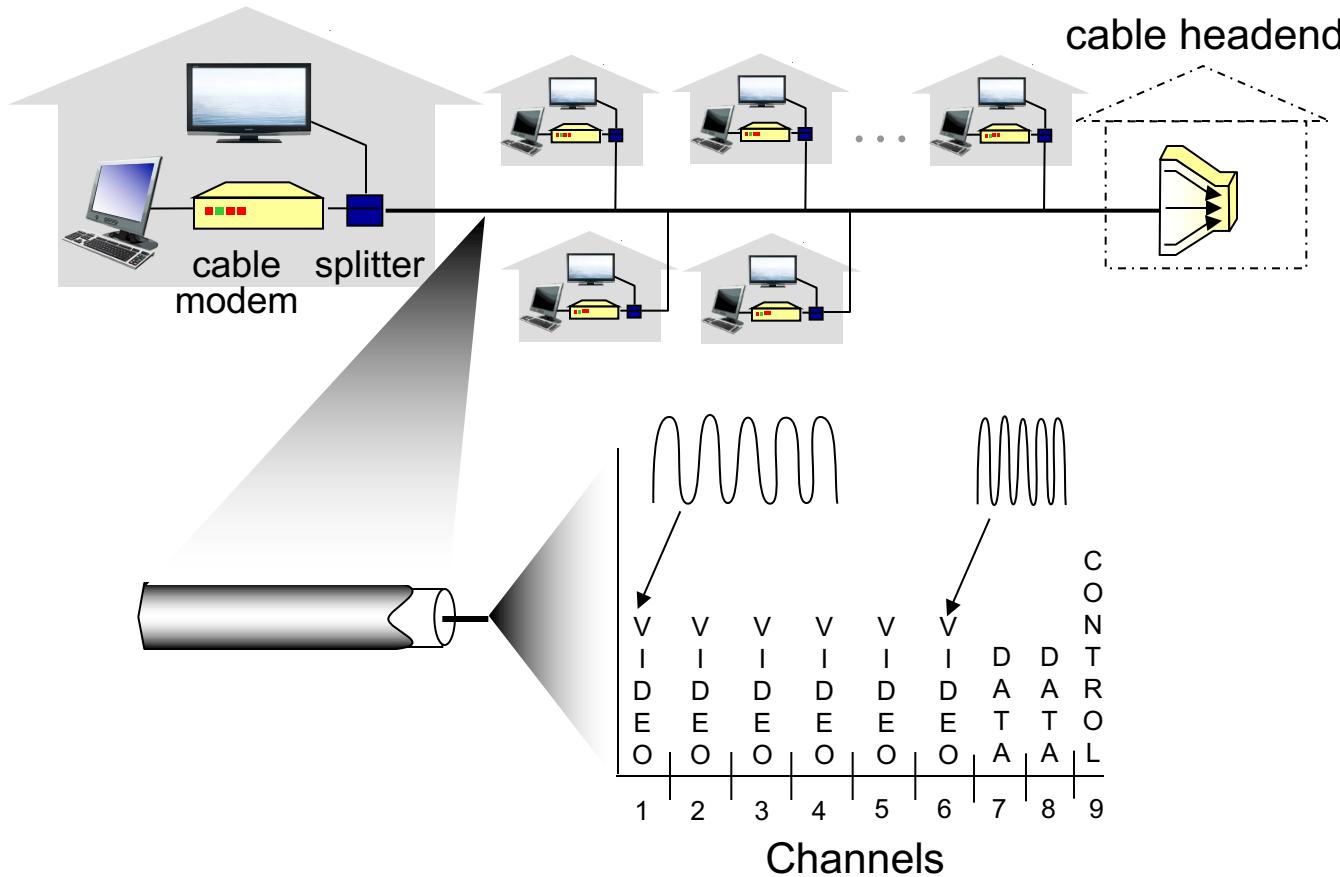


Access network: digital subscriber line (DSL)



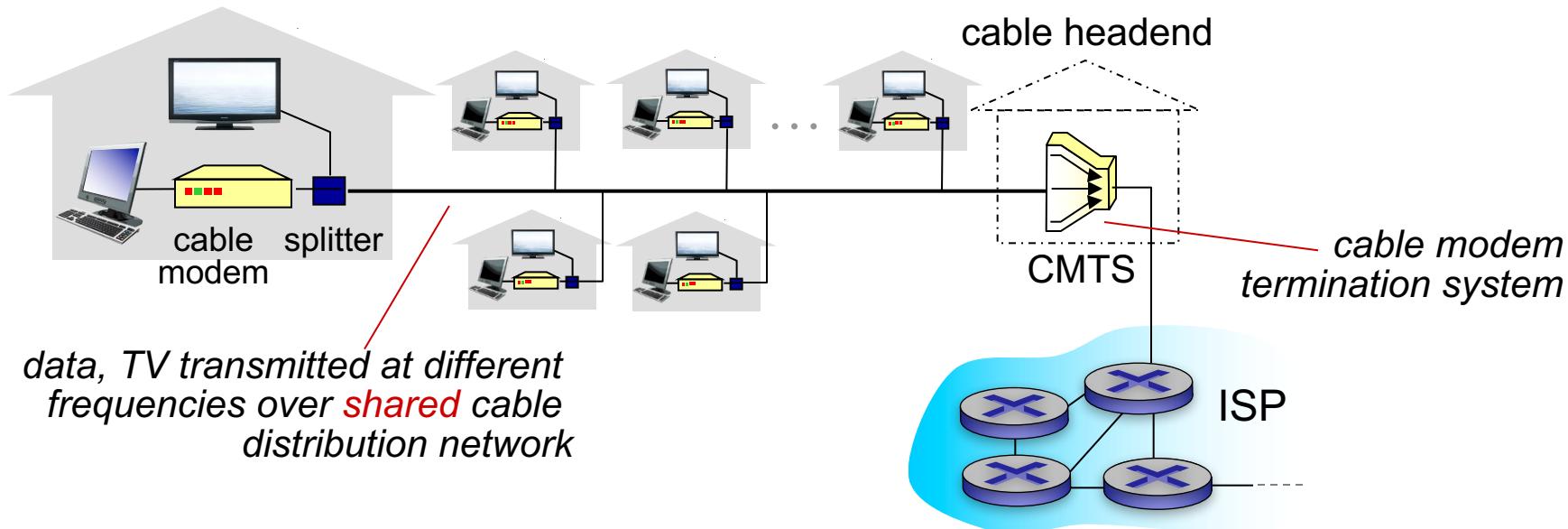
- use **existing** telephone line to central office DSLAM
 - data over DSL phone line goes to Internet
 - voice over DSL phone line goes to telephone net
- < 2.5 Mbps upstream transmission rate (typically < 1 Mbps)
- < 24 Mbps downstream transmission rate (typically < 10 Mbps)

Access network: cable network



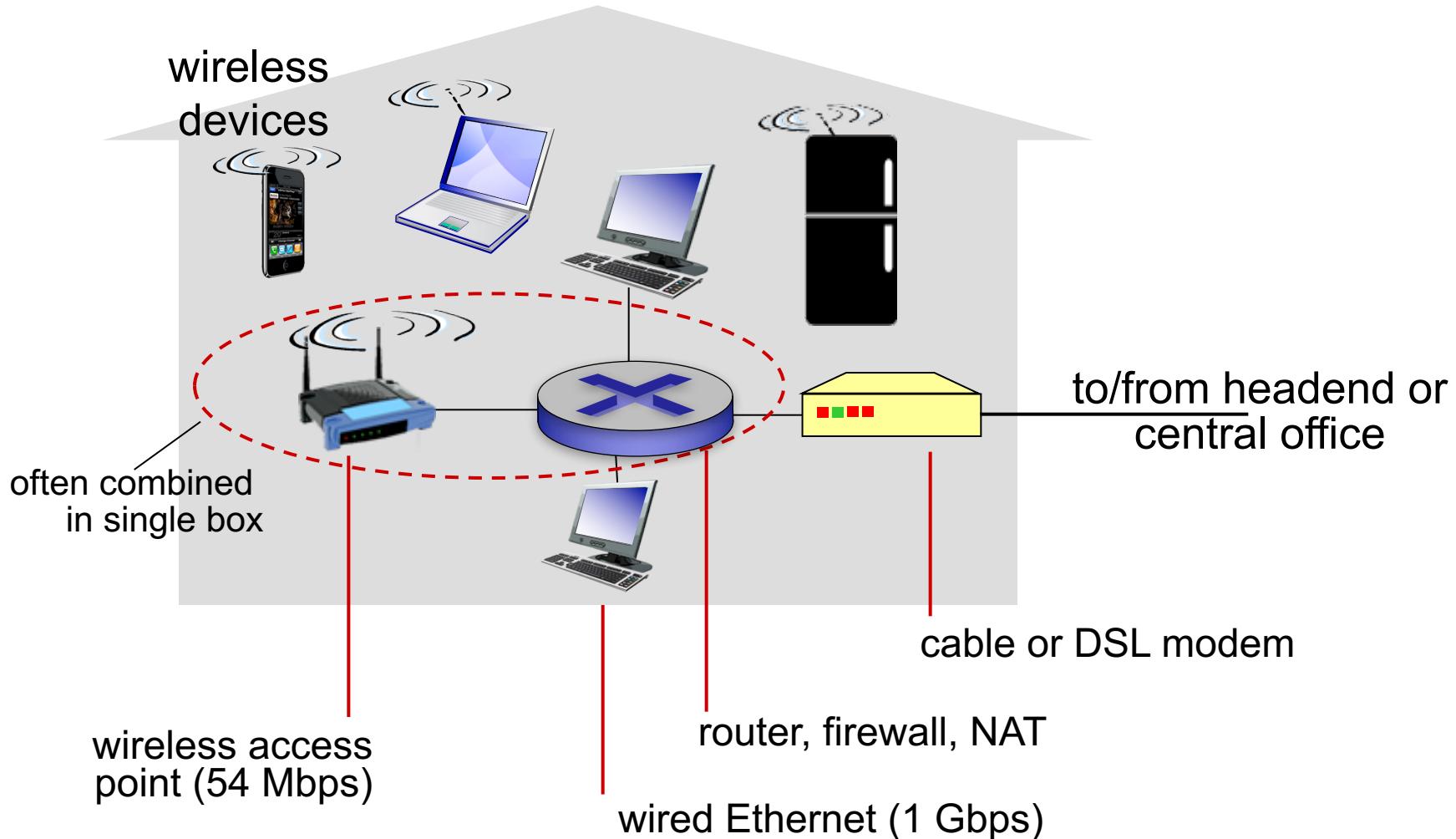
frequency division multiplexing: different channels transmitted in different frequency bands

Access network: cable network

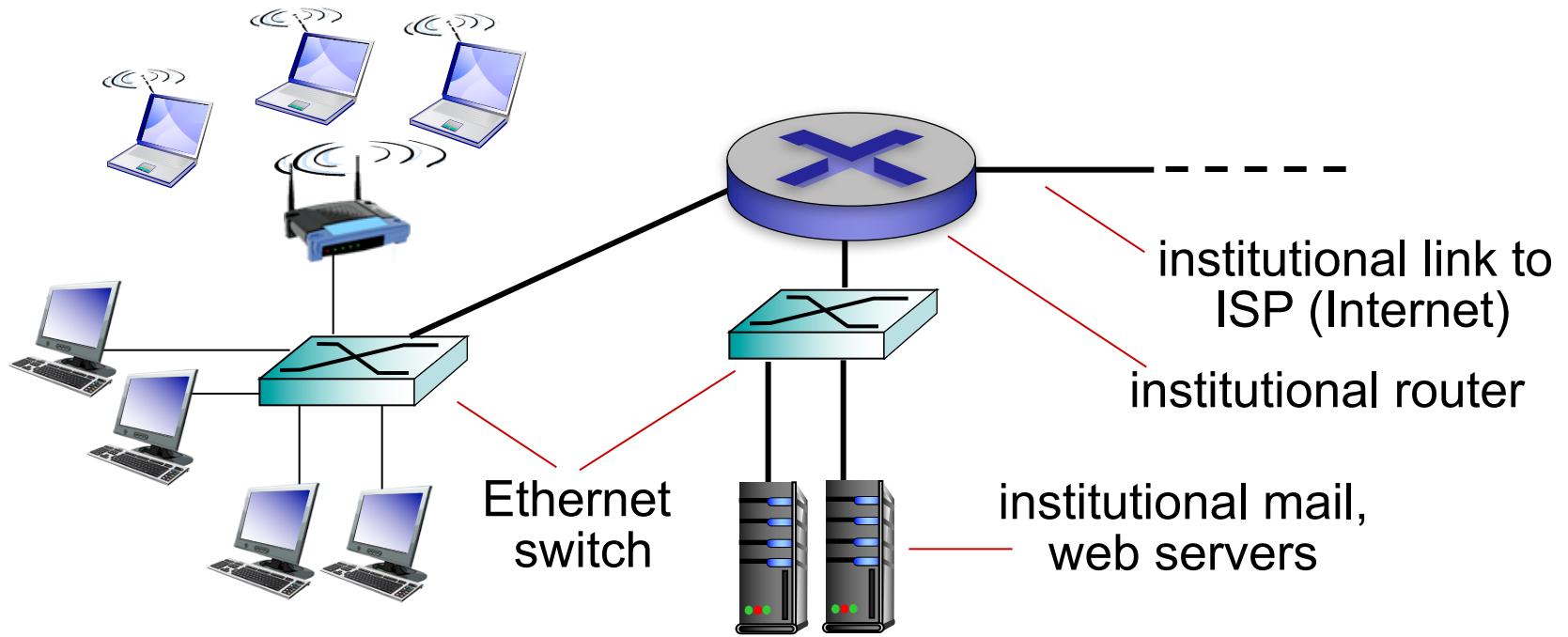


- HFC: hybrid fiber coax
 - asymmetric: up to 30Mbps downstream transmission rate, 2 Mbps upstream transmission rate
- network of cable, fiber attaches homes to ISP router
 - homes **share access network** to cable headend
 - unlike DSL, which has dedicated access to central office

Access network: home network



Enterprise access networks (Ethernet)



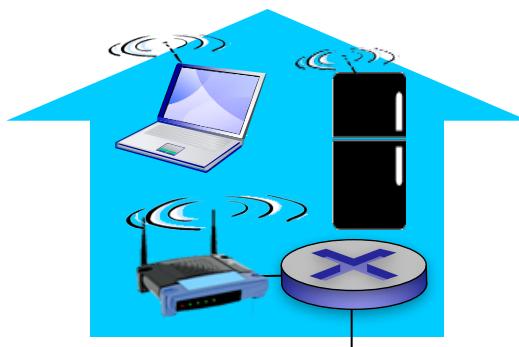
- typically used in companies, universities, etc.
- 10 Mbps, 100Mbps, 1Gbps, 10Gbps transmission rates
- today, end systems typically connect into Ethernet switch

Wireless access networks

- shared wireless access network connects end system to router
 - via base station aka “access point”

wireless LANs:

- within building (100 ft.)
- 802.11b/g/n (WiFi): 11, 54, 450 Mbps transmission rate



to Internet

wide-area wireless access

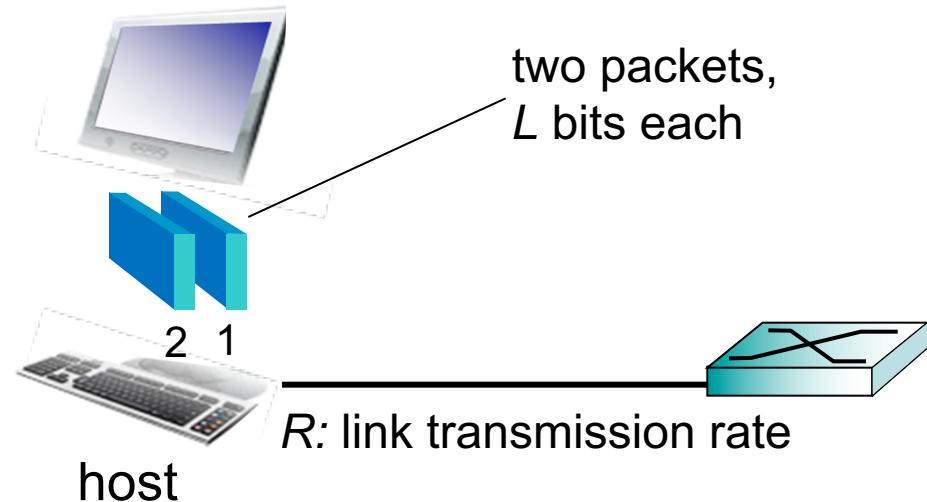
- provided by telco (cellular) operator, 10's km
- between 1 and 10 Mbps
- 3G, 4G: LTE



Host: sends packets of data

host sending function:

- takes application message
- breaks into smaller chunks, known as *packets*, of length L bits
- transmits packet into access network at *transmission rate R*
 - link transmission rate, aka link *capacity*, aka *link bandwidth*



$$\text{packet transmission delay} = \frac{\text{time needed to transmit } L\text{-bit packet into link}}{R \text{ (bits/sec)}}$$

Physical media

- **bit:** propagates between transmitter/receiver pairs
- **physical link:** what lies between transmitter & receiver
- **guided media:**
 - signals propagate in solid media: copper, fiber, coax
- **unguided media:**
 - signals propagate freely, e.g., radio

twisted pair (TP)

- two insulated copper wires
 - Category 5: 100 Mbps, 1 Gbps Ethernet
 - Category 6: 10Gbps



Physical media: coax, fiber

coaxial cable:

- two concentric copper conductors
- bidirectional
- broadband:
 - multiple channels on cable
 - HFC



fiber optic cable:

- glass fiber carrying light pulses, each pulse a bit
- high-speed operation:
 - high-speed point-to-point transmission (e.g., 10' s-100' s Gbps transmission rate)
- low error rate:
 - repeaters spaced far apart
 - immune to electromagnetic noise



Physical media: radio

- signal carried in electromagnetic spectrum
- no physical “wire”
- bidirectional
- propagation environment effects:
 - reflection
 - obstruction by objects
 - interference

radio link types:

- terrestrial microwave
 - e.g. up to 45 Mbps channels
- LAN (e.g., WiFi)
 - 54 Mbps
- wide-area (e.g., cellular)
 - 4G cellular: ~ 10 Mbps
- satellite
 - Kbps to 45Mbps channel (or multiple smaller channels)
 - 270 msec end-end delay
 - geosynchronous versus low altitude

Chapter I: roadmap

I.1 what *is* the Internet?

I.2 network edge

- end systems, access networks, links

I.3 network core

- packet switching, circuit switching, network structure

I.4 delay, loss, throughput in networks

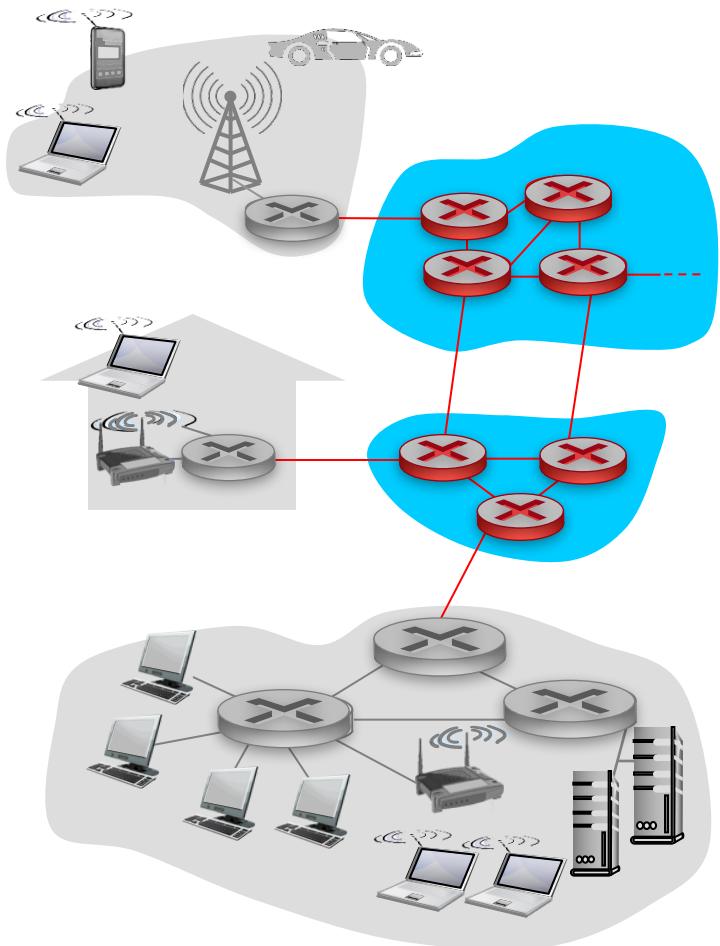
I.5 protocol layers, service models

I.6 networks under attack: security

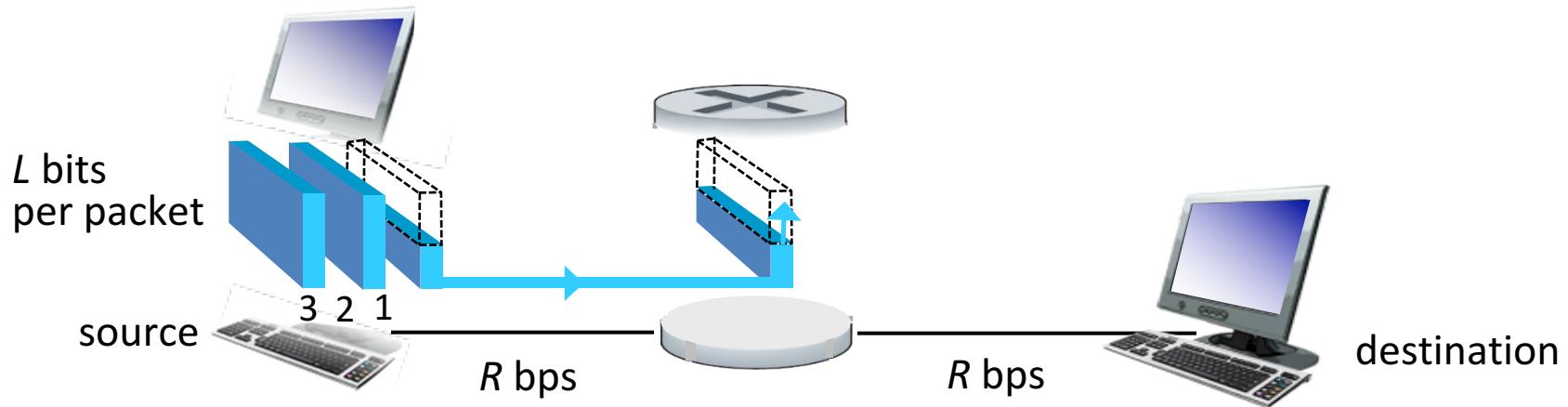
I.7 history

The network core

- mesh of interconnected routers
- **packet-switching:** hosts break application-layer messages into *packets*
 - forward packets from one router to the next, across links on path from source to destination
 - each packet transmitted at full link capacity



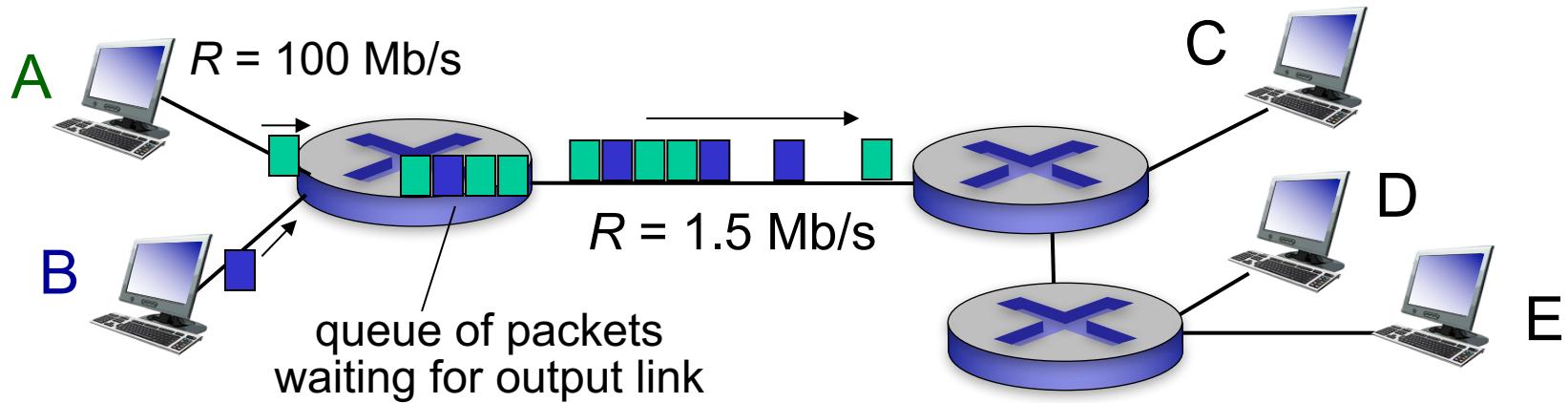
Packet-switching: store-and-forward



- takes L/R seconds to transmit (push out) L -bit packet into link at R bps
- **store and forward:** entire packet must arrive at router before it can be transmitted on next link
- end-end delay = $2L/R$ (assuming zero propagation delay)

- one-hop numerical example:*
- $L = 7.5 \text{ Mbits}$
 - $R = 1.5 \text{ Mbps}$
 - one-hop transmission delay = 5 sec
- } more on delay shortly ...

Packet Switching: queueing delay, loss



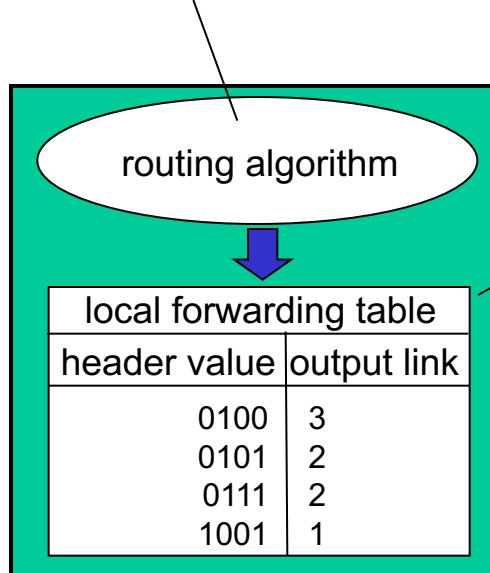
queuing and loss:

- if arrival rate (in bits) to link exceeds transmission rate of link for a period of time:
 - packets will queue, wait to be transmitted on link
 - packets can be dropped (lost) if memory (buffer) fills up

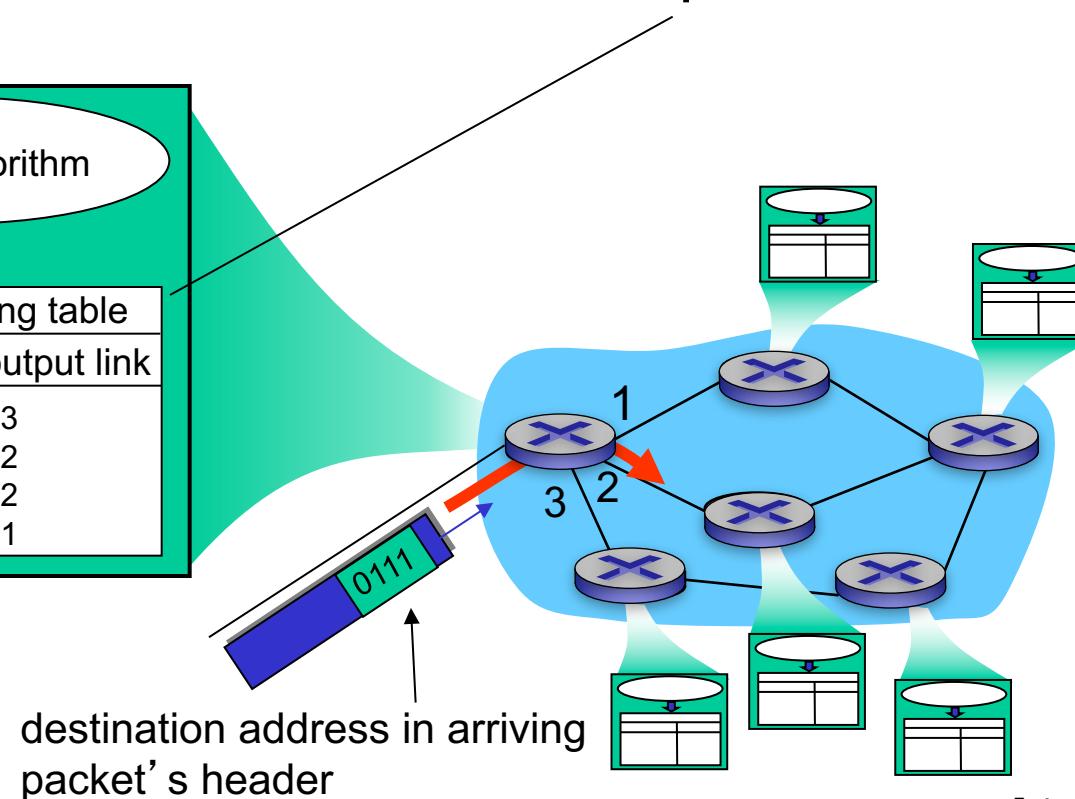
Two key network-core functions

routing: determines source-destination route taken by packets

- *routing algorithms*



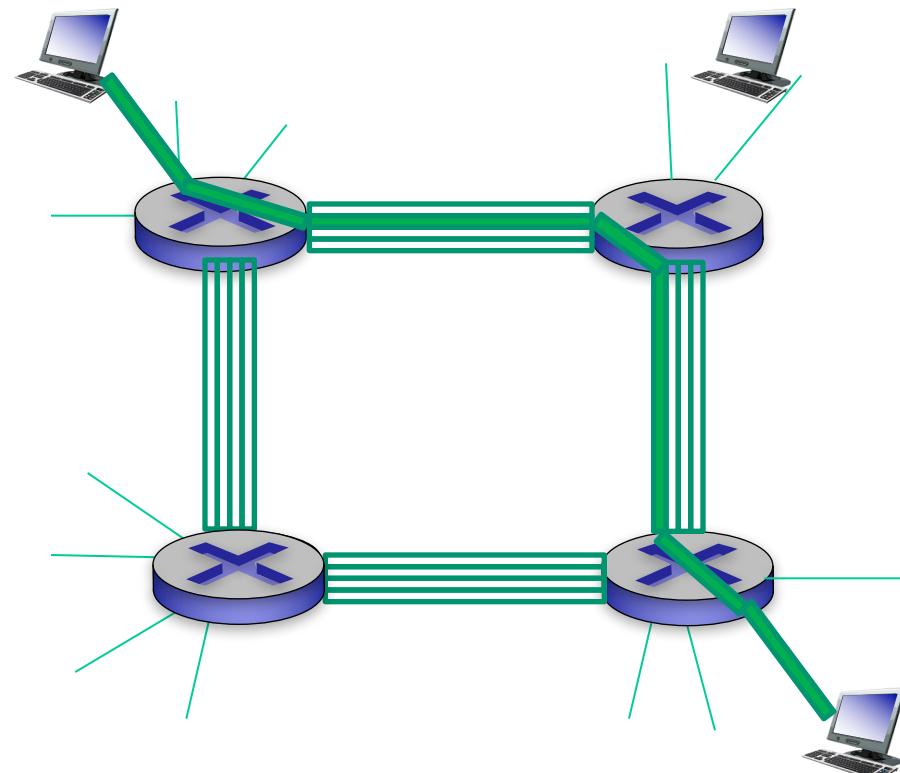
forwarding: move packets from router's input to appropriate router output



Alternative core: circuit switching

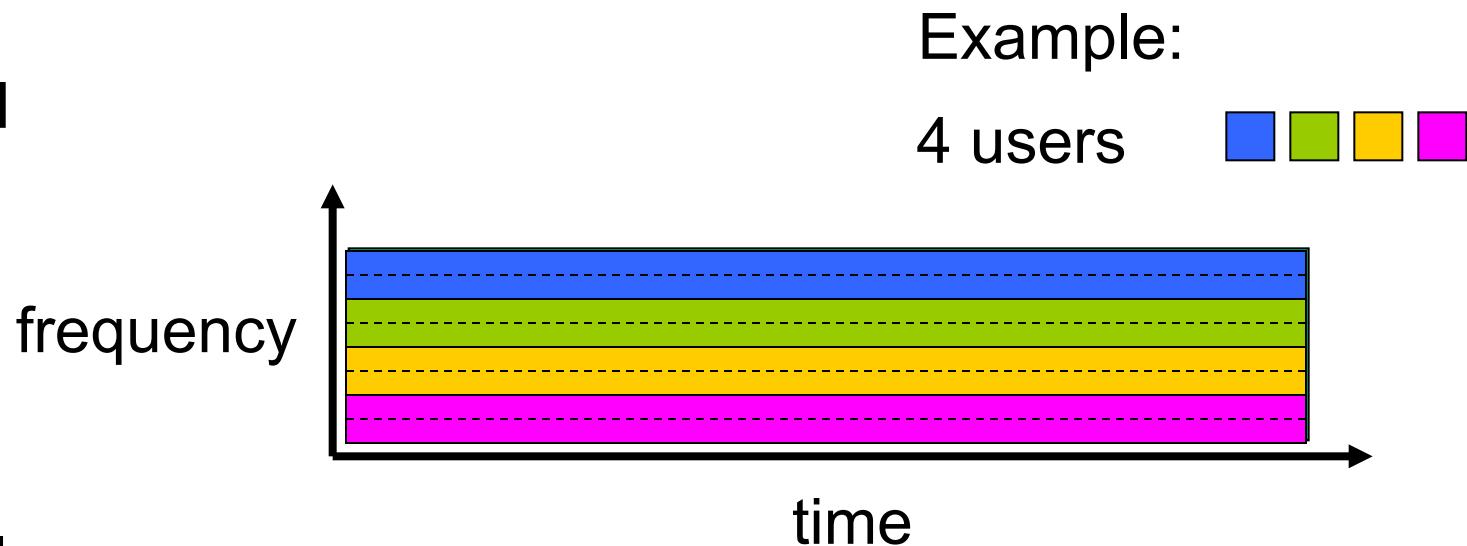
end-end resources allocated to, reserved for “call” between source & dest:

- in diagram, each link has four circuits.
 - call gets 2nd circuit in top link and 1st circuit in right link.
- dedicated resources: no sharing
 - circuit-like (guaranteed) performance
- circuit segment idle if not used by call (*no sharing*)
- commonly used in traditional telephone networks

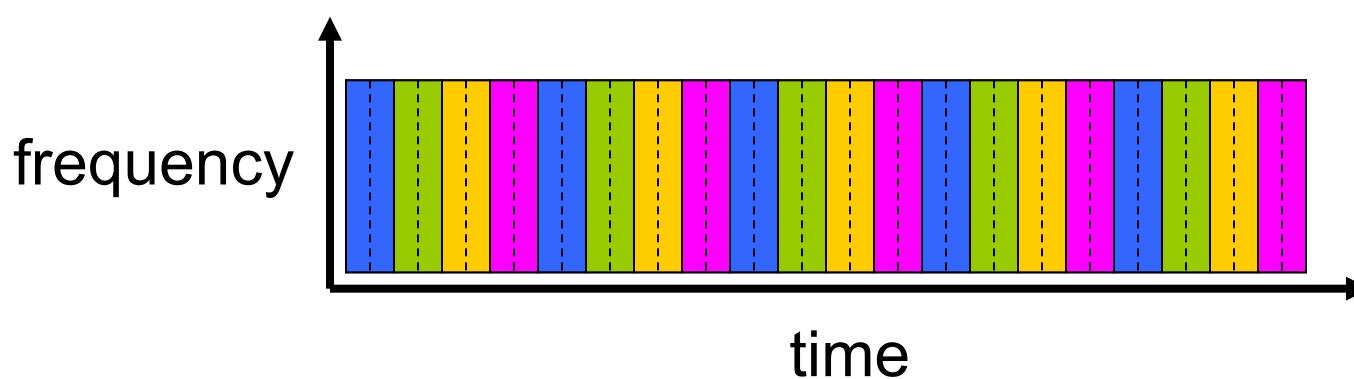


Circuit switching: FDM versus TDM

FDM



TDM

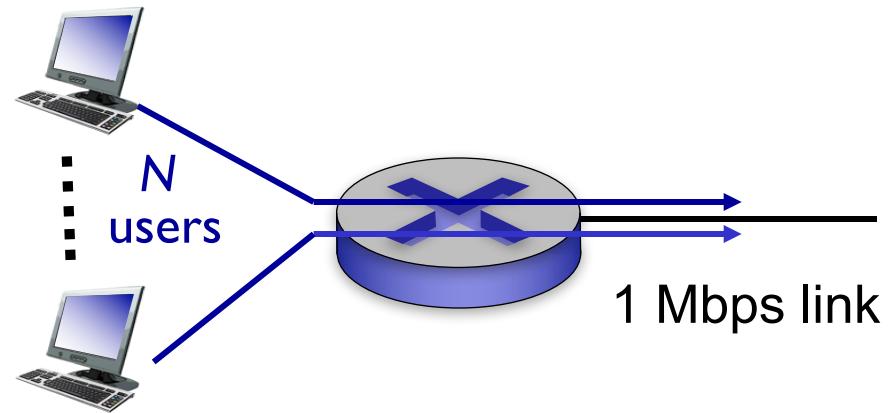


Packet switching versus circuit switching

packet switching allows more users to use network!

example:

- 1 Mb/s link
- each user:
 - 100 kb/s when “active”
 - active 10% of time
- *circuit-switching*:
 - 10 users
- *packet switching*:
 - with 35 users, probability > 10 active at same time is less than .0004 *



Q: how did we get value 0.0004?

Q: what happens if > 35 users ?

* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

Packet switching versus circuit switching

is packet switching a “slam dunk winner?”

- great for bursty data
 - resource sharing
 - simpler, no call setup
- **excessive congestion possible:** packet delay and loss
 - protocols needed for reliable data transfer, congestion control
- **Q: How to provide circuit-like behavior?**
 - bandwidth guarantees needed for audio/video apps
 - still an unsolved problem (chapter 7)

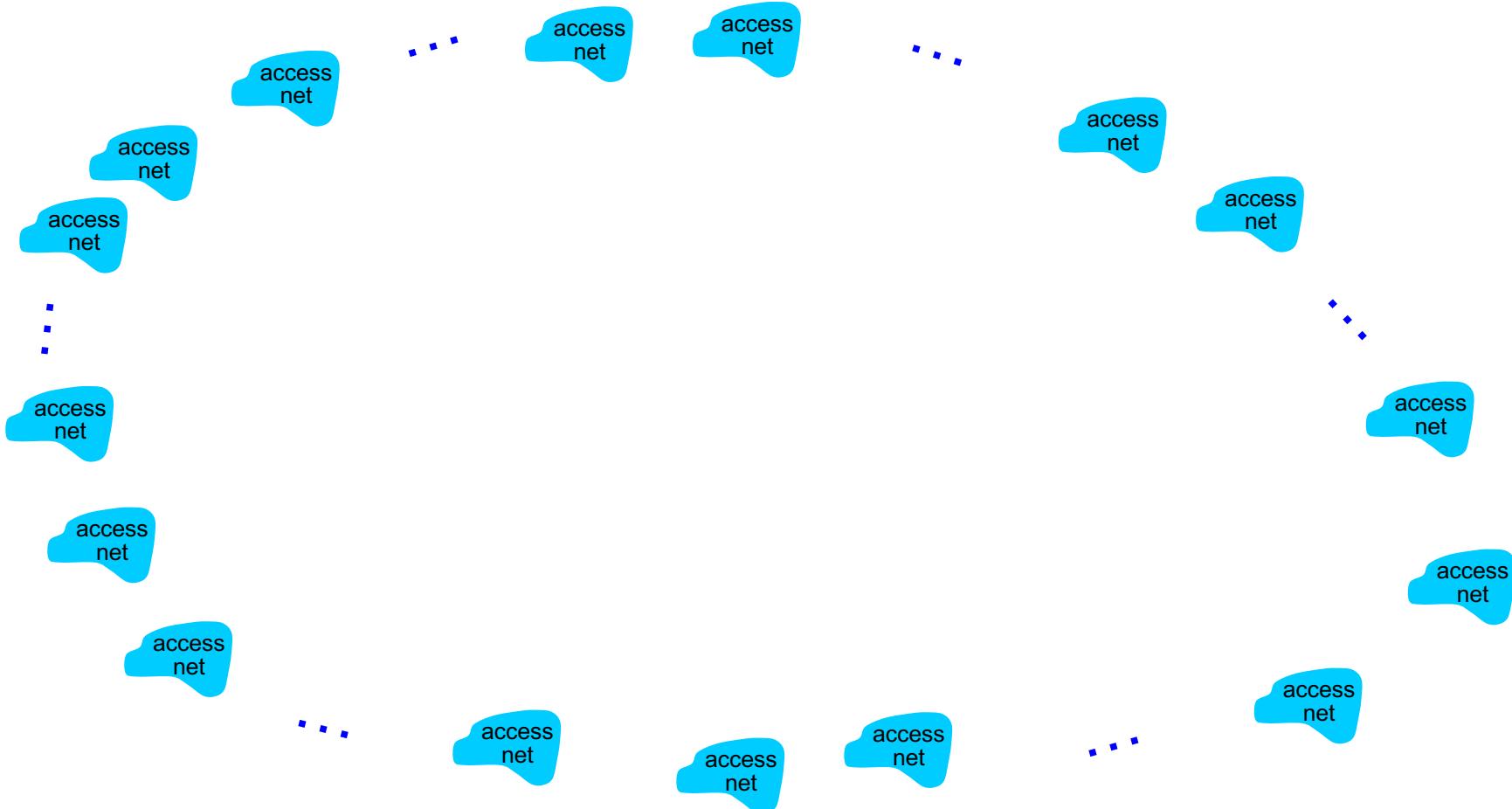
Q: human analogies of reserved resources (circuit switching)
versus on-demand allocation (packet-switching)?

Internet structure: network of networks

- End systems connect to Internet via **access ISPs** (Internet Service Providers)
 - residential, company and university ISPs
- Access ISPs in turn must be interconnected.
 - so that any two hosts can send packets to each other
- Resulting network of networks is very complex
 - evolution was driven by **economics** and **national policies**
- Let's take a stepwise approach to describe current Internet structure

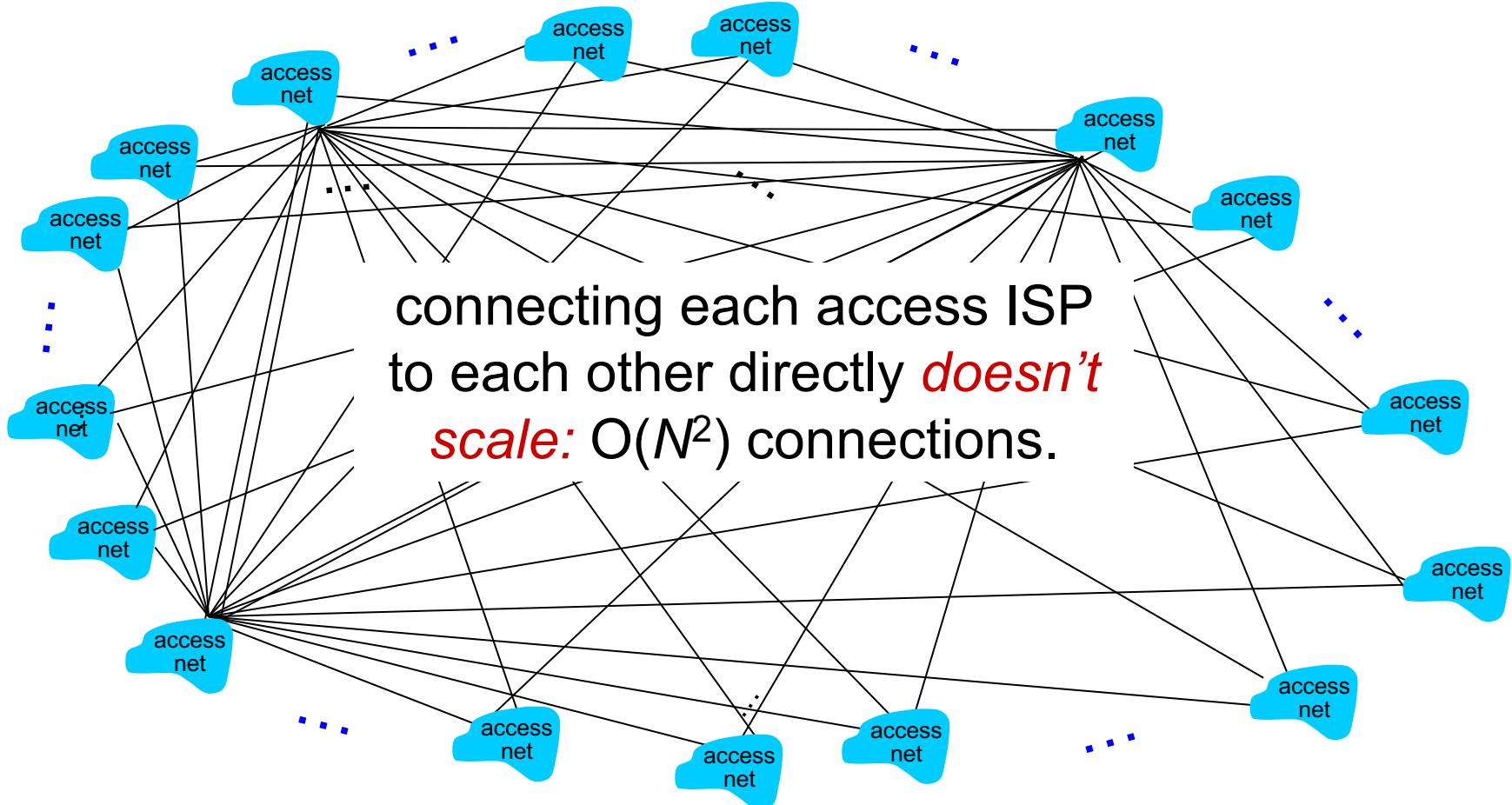
Internet structure: network of networks

Question: given *millions* of access ISPs, how to connect them together?



Internet structure: network of networks

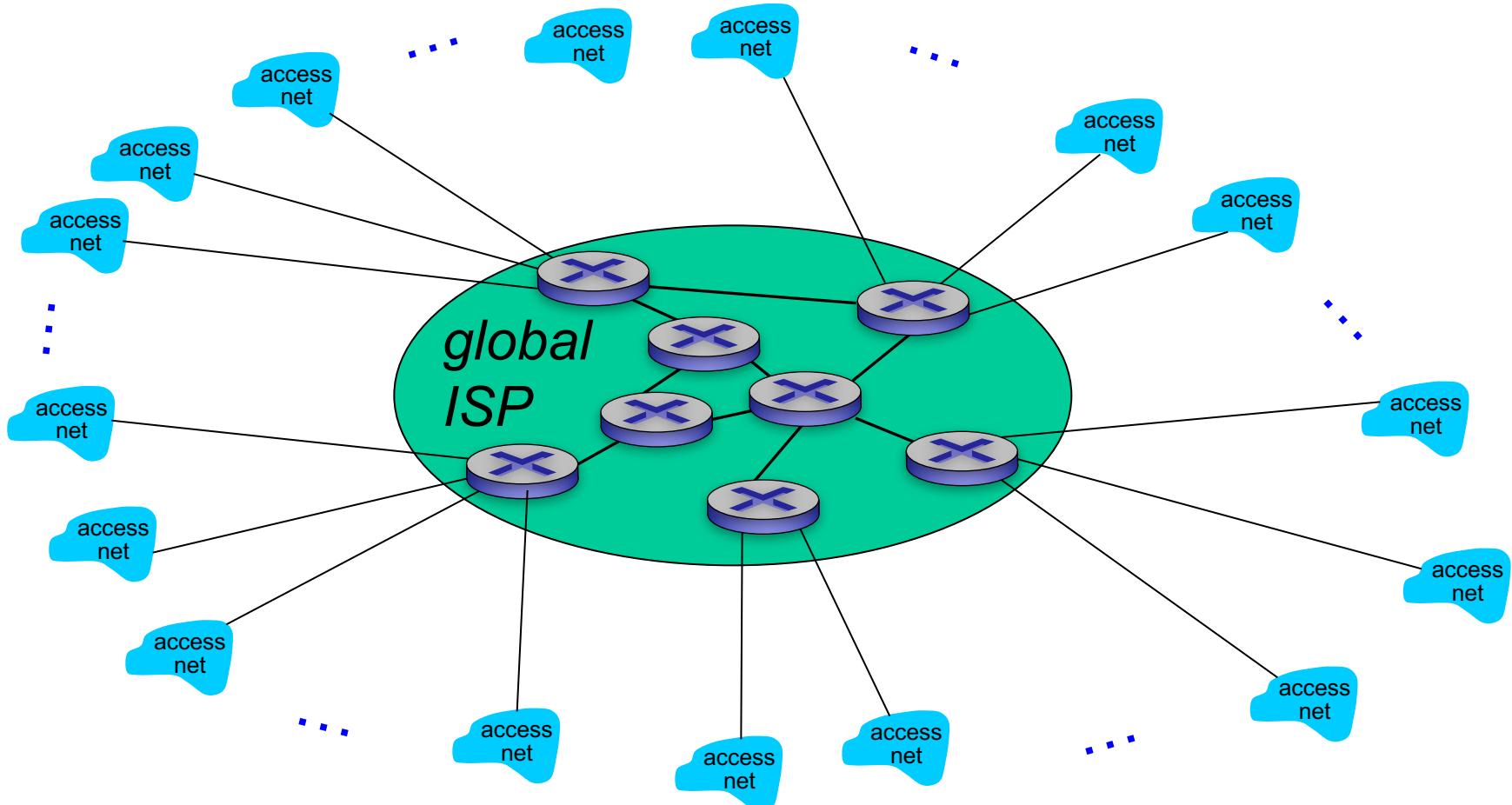
Option: connect each access ISP to every other access ISP?



Internet structure: network of networks

Option: connect each access ISP to one *global transit ISP*?

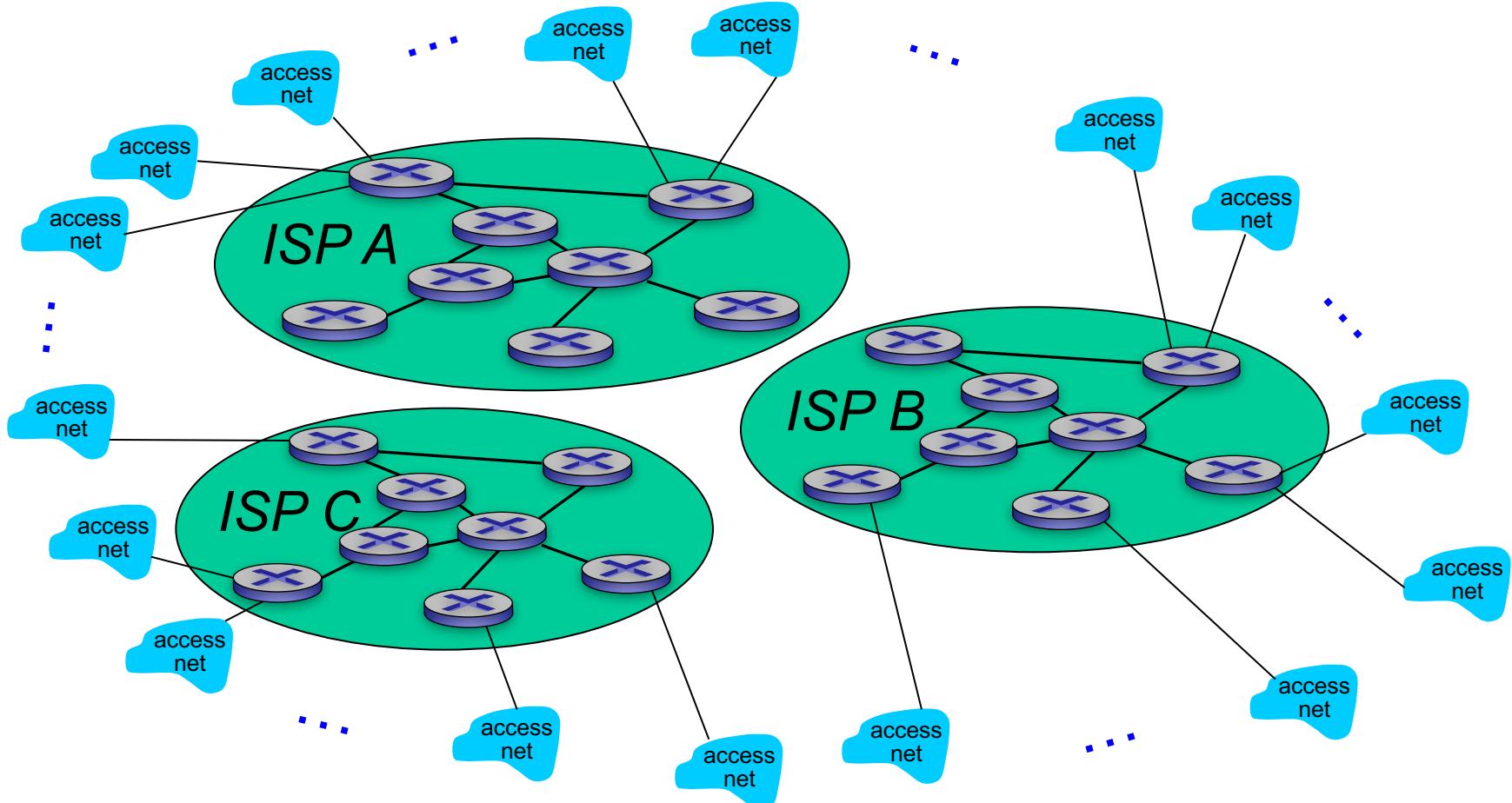
Customer and provider ISPs have economic agreement.



Internet structure: network of networks

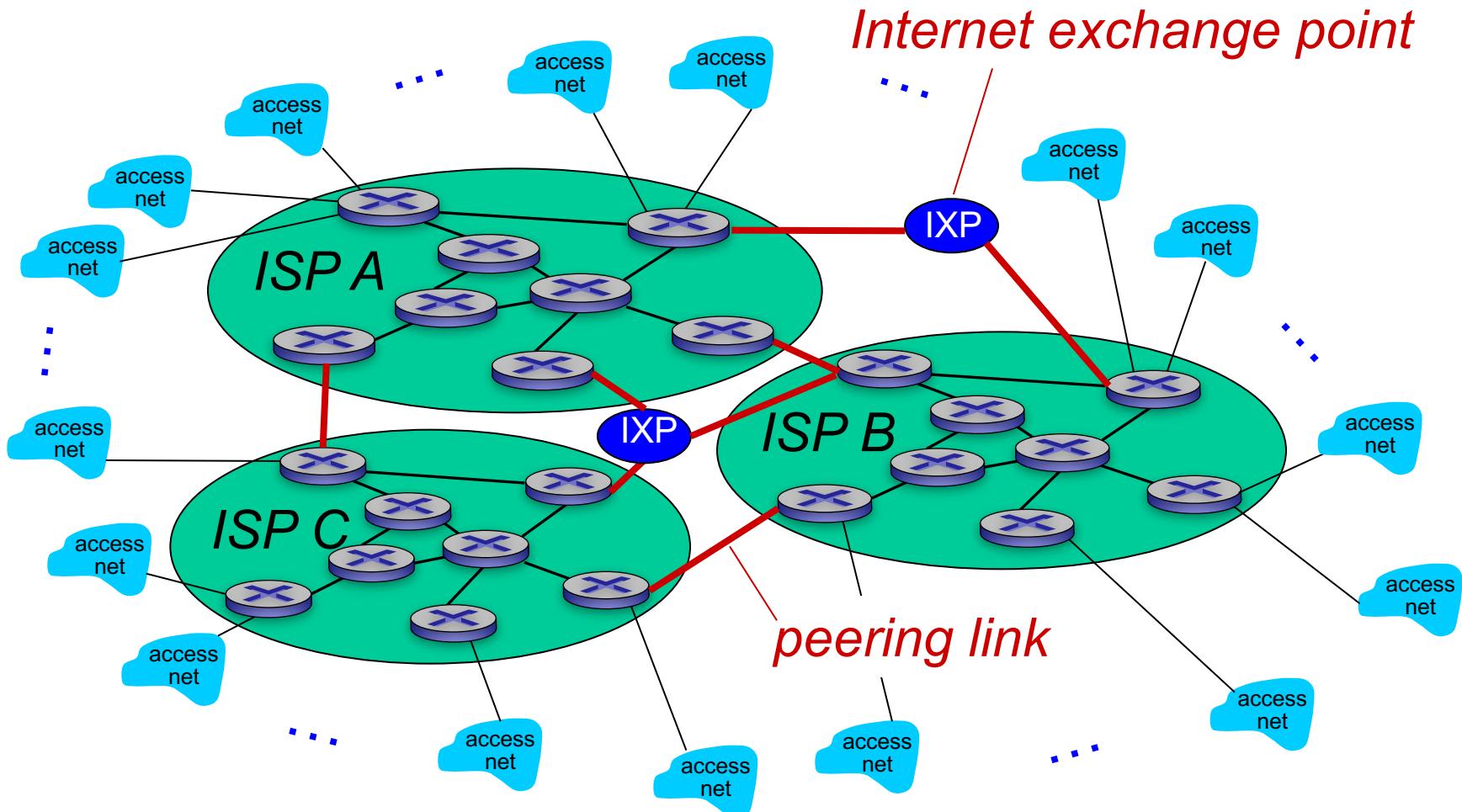
But if one global ISP is viable business, there will be competitors

....



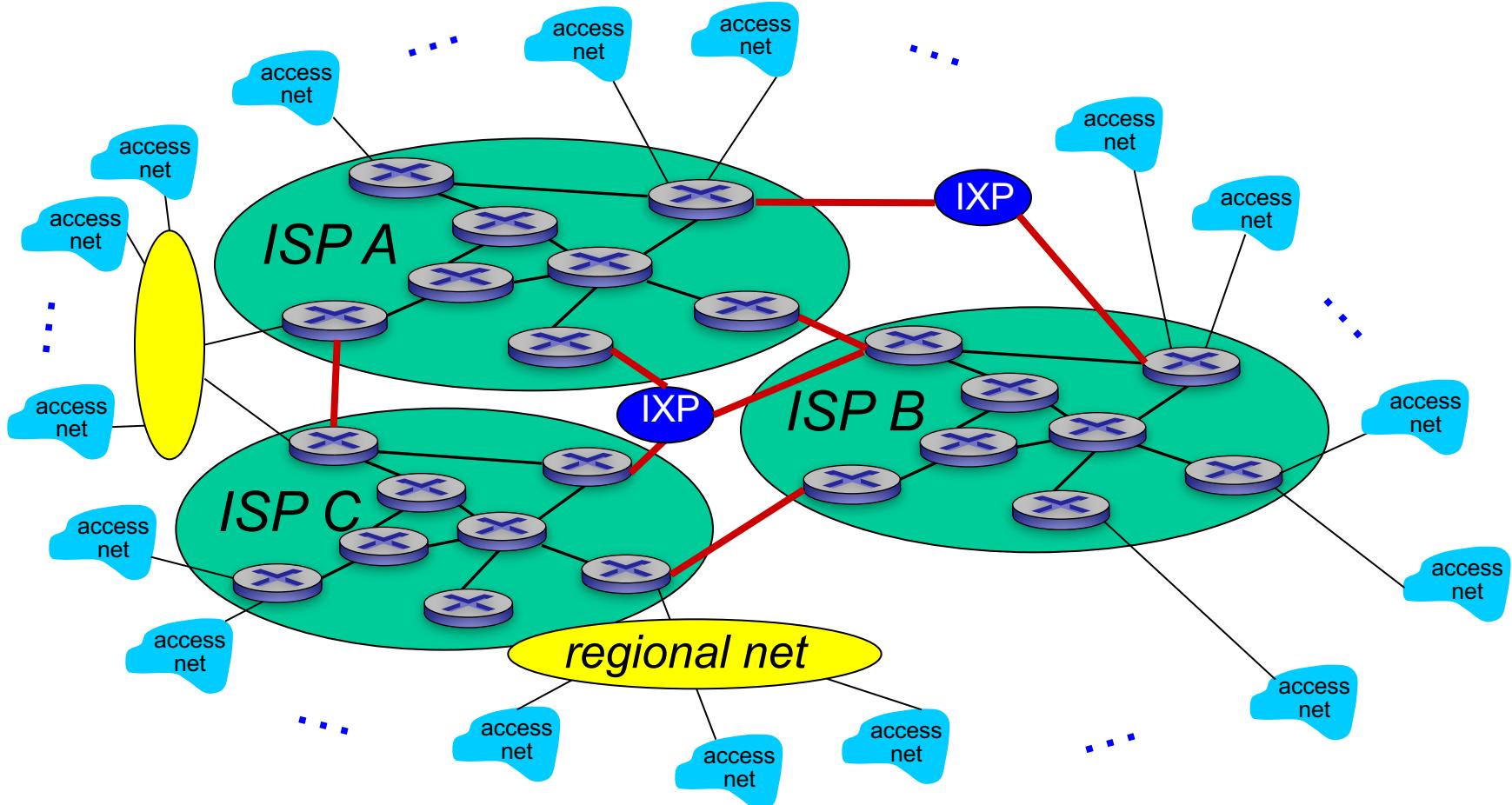
Internet structure: network of networks

But if one global ISP is viable business, there will be competitors
.... which must be interconnected



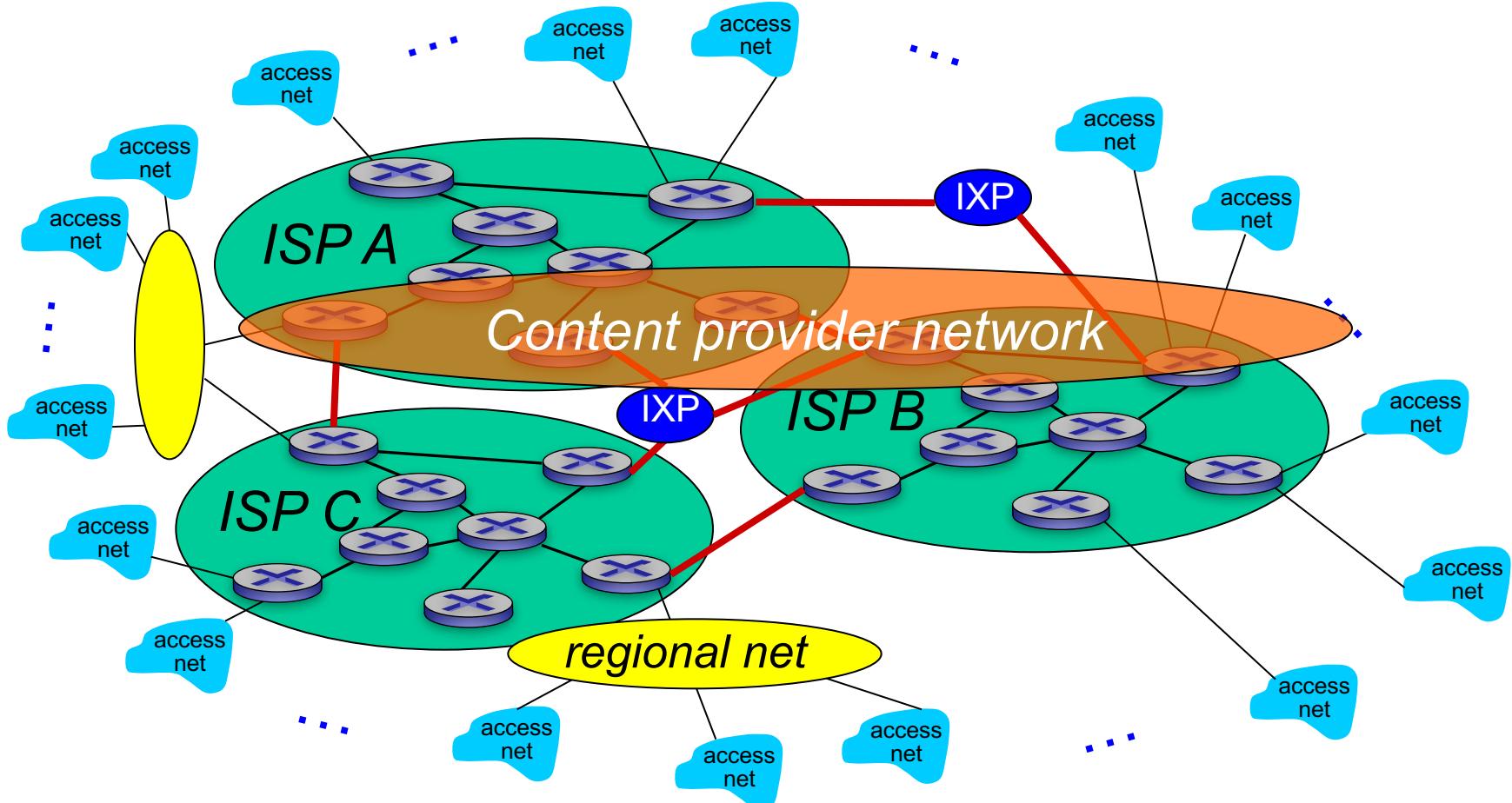
Internet structure: network of networks

... and regional networks may arise to connect access nets to ISPs

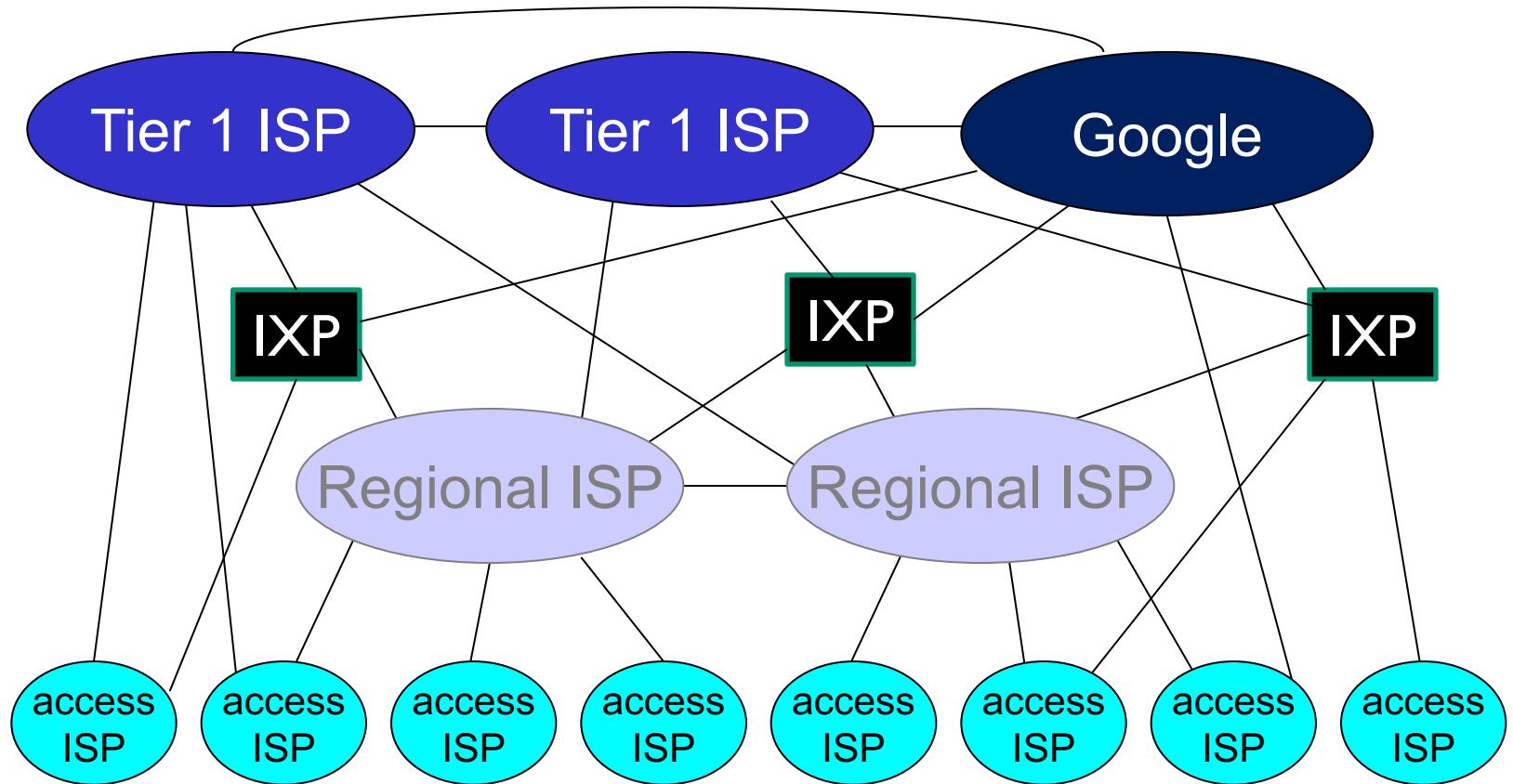


Internet structure: network of networks

... and content provider networks (e.g., Google, Microsoft, Akamai) may run their own network, to bring services, content close to end users

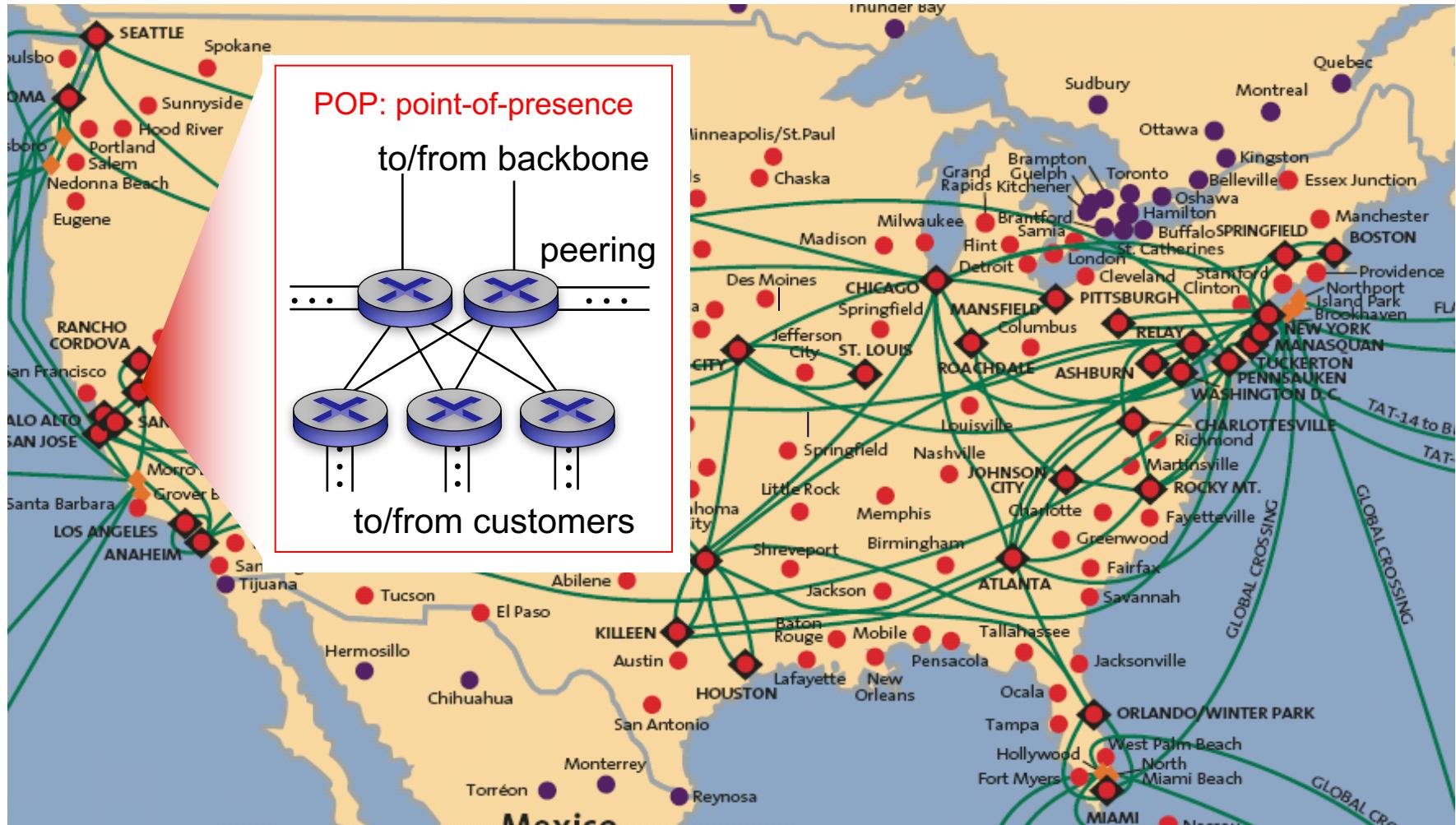


Internet structure: network of networks



- at center: small # of well-connected large networks
 - “tier-1” commercial ISPs (e.g., Level 3, Sprint, AT&T, NTT), national & international coverage
 - content provider network (e.g., Google): private network that connects its data centers to Internet, often bypassing tier-1, regional ISPs

Tier-I ISP: e.g., Sprint



Chapter I: roadmap

I.1 what *is* the Internet?

I.2 network edge

- end systems, access networks, links

I.3 network core

- packet switching, circuit switching, network structure

I.4 delay, loss, throughput in networks

I.5 protocol layers, service models

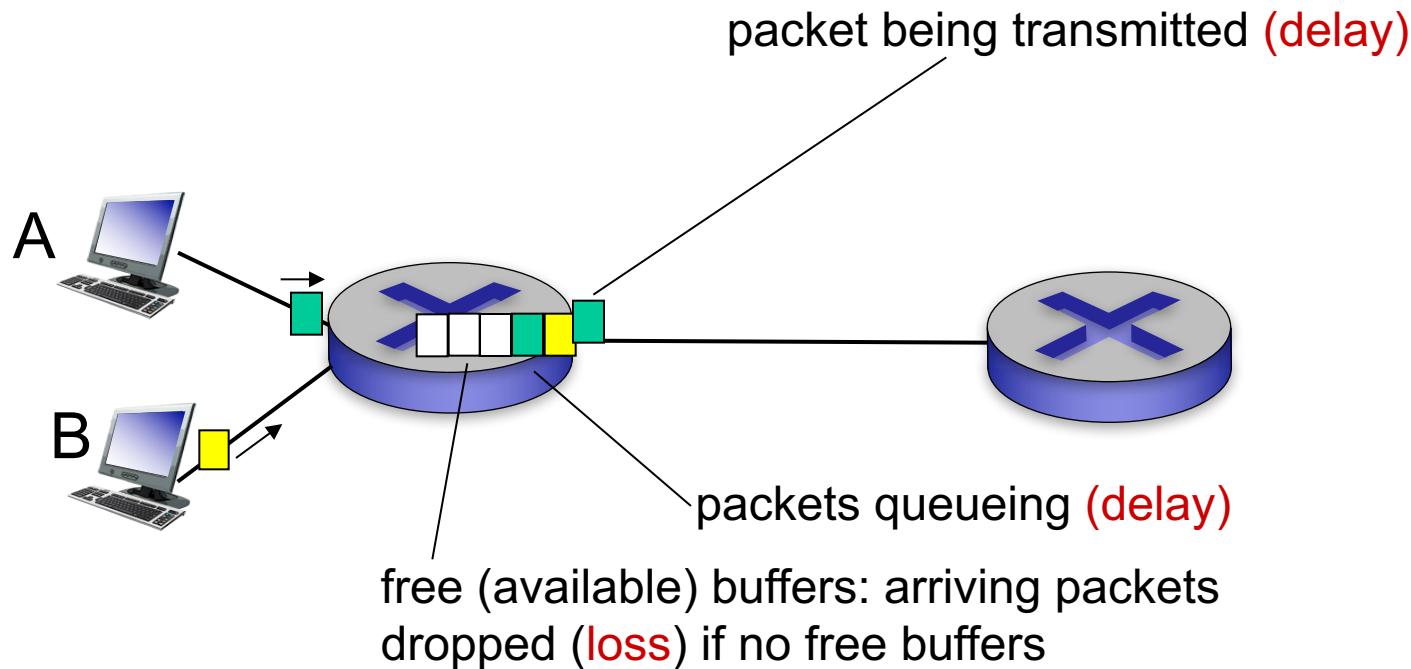
I.6 networks under attack: security

I.7 history

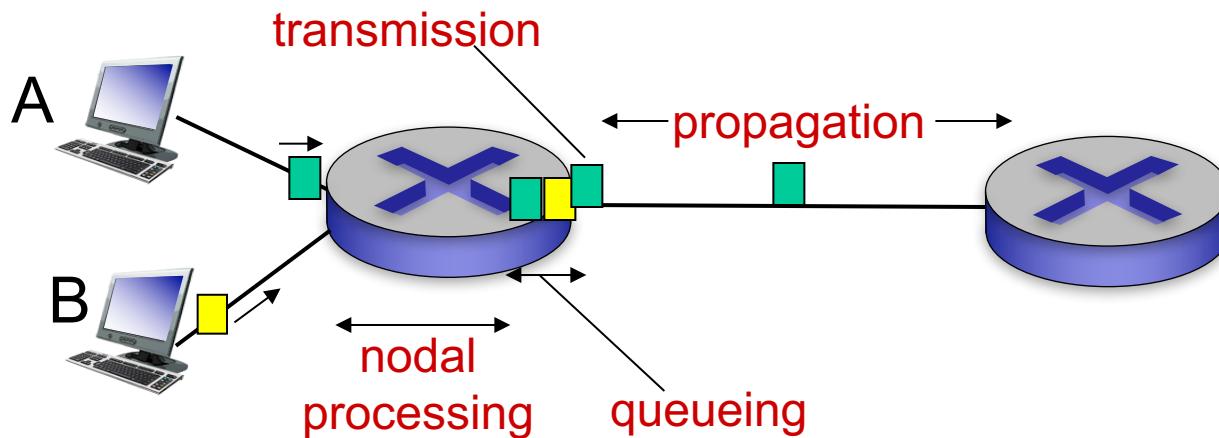
How do loss and delay occur?

packets queue in router buffers

- packet arrival rate to link (temporarily) exceeds output link capacity
- packets queue, wait for turn



Four sources of packet delay



$$d_{\text{nodal}} = d_{\text{proc}} + d_{\text{queue}} + d_{\text{trans}} + d_{\text{prop}}$$

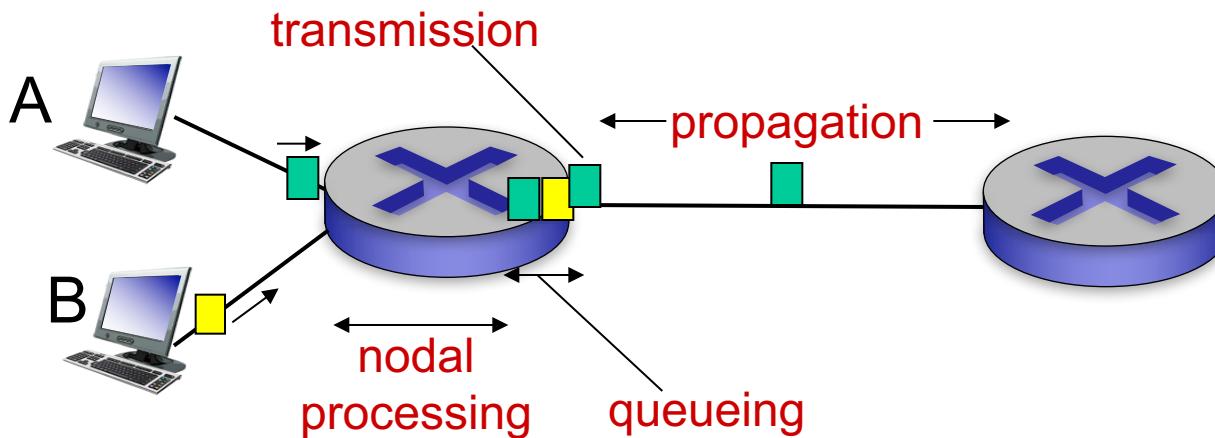
d_{proc} : nodal processing

- check bit errors
- determine output link
- typically < msec

d_{queue} : queueing delay

- time waiting at output link for transmission
- depends on congestion level of router

Four sources of packet delay



$$d_{\text{nodal}} = d_{\text{proc}} + d_{\text{queue}} + d_{\text{trans}} + d_{\text{prop}}$$

d_{trans} : transmission delay:

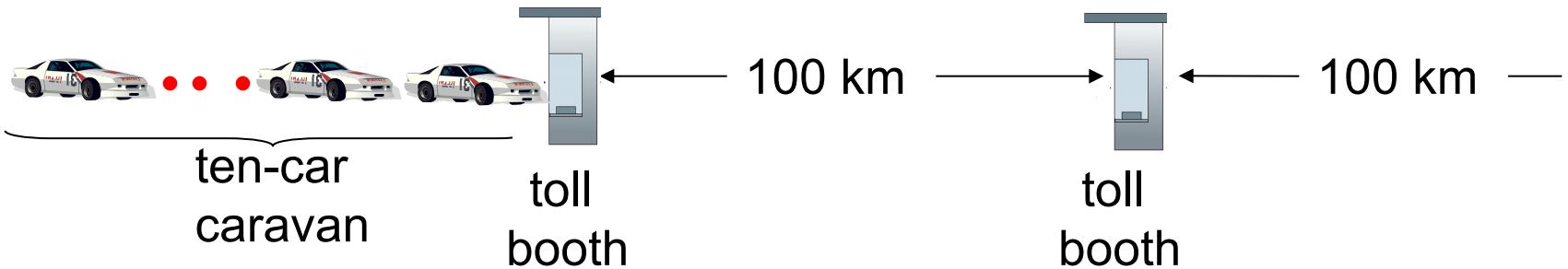
- L : packet length (bits)
- R : link bandwidth (bps)
- $d_{\text{trans}} = L/R$ ← d_{trans} and d_{prop} →
very different

d_{prop} : propagation delay:

- d : length of physical link
- s : propagation speed ($\sim 2 \times 10^8 \text{ m/sec}$)
- $d_{\text{prop}} = d/s$

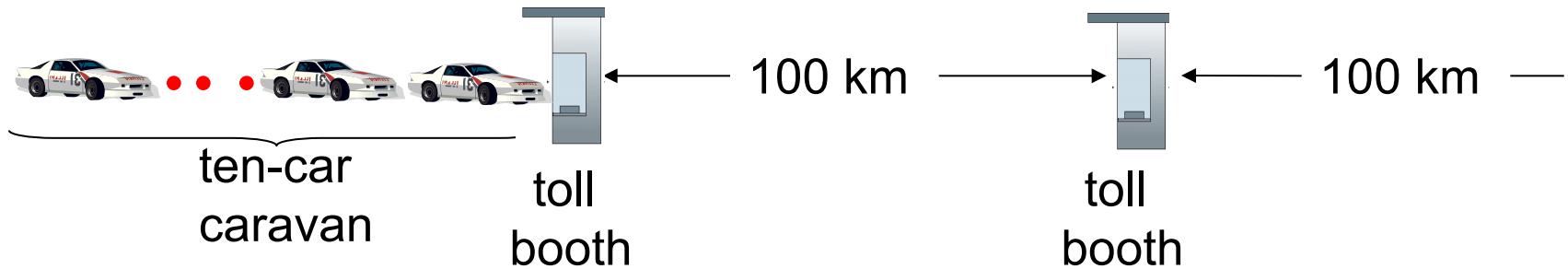
- * Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/
- * Check out the Java applet for an interactive animation on trans vs. prop delay

Caravan analogy



- cars “propagate” at 100 km/hr
- toll booth takes 12 sec to service car (bit transmission time)
- car ~ bit; caravan ~ packet
- Q: How long until caravan is lined up before 2nd toll booth?
A: 62 minutes
- time to “push” entire caravan through toll booth onto highway = $12*10 = 120$ sec
- time for last car to propagate from 1st to 2nd toll both:
 $100\text{km}/(100\text{km/hr})= 1\text{ hr}$

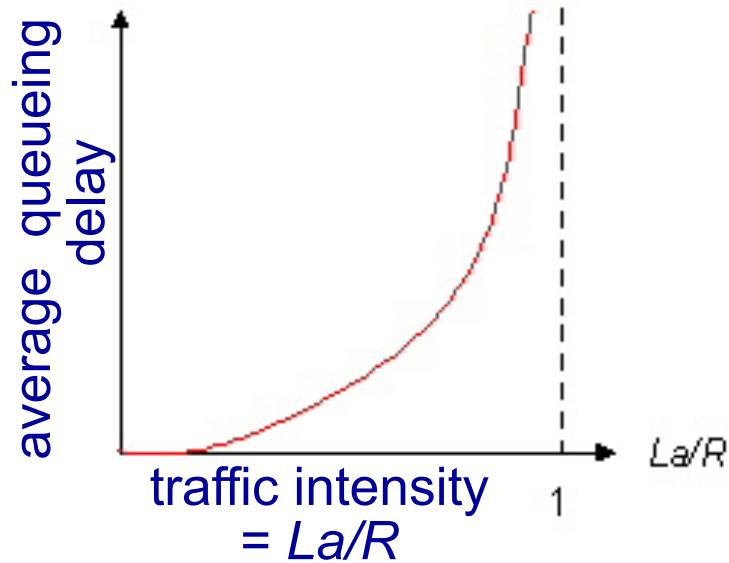
Caravan analogy (more)



- suppose cars now “propagate” at 1000 km/hr
- and suppose toll booth now takes one min to service a car
- **Q: Will cars arrive to 2nd booth before all cars serviced at first booth?**
 - **A: Yes!** after 7 min, first car arrives at second booth, three cars still at first booth; after 16 minutes all cars arrived.
- **Q: Will completed caravan arrive to 2nd booth before caravan in previous example (12 sec booth, 100 km/hr)?**
 - **A: Yes!** Previous whole caravan arrived after 62 minutes, last whole caravan arrived after 16 minutes

Queueing delay (revisited)

- R : link bandwidth (bps)
- L : packet length (bits)
- a : average packet arrival rate



- $La/R \sim 0$: avg. queueing delay small
- $La/R \rightarrow 1$: avg. queueing delay large
- $La/R > 1$: more “work” arriving than can be serviced, average delay infinite!



$La/R \sim 0$

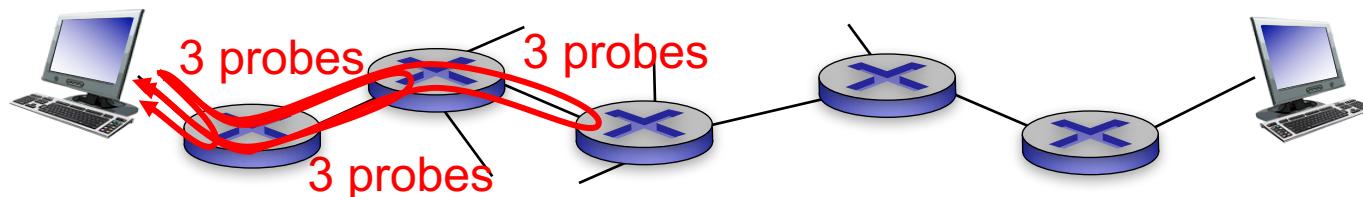


$La/R \rightarrow 1$

* Check online interactive animation on queuing and loss

“Real” Internet delays and routes

- what do “real” Internet delay & loss look like?
- **traceroute** program: provides delay measurement from source to router along end-end Internet path towards destination. For all i :
 - sends three packets that will reach router i on path towards destination
 - router i will return packets to sender
 - sender times interval between transmission and reply.



“Real” Internet delays, routes

traceroute: gaia.cs.umass.edu to www.eurecom.fr

3 delay measurements from
gaia.cs.umass.edu to cs-gw.cs.umass.edu

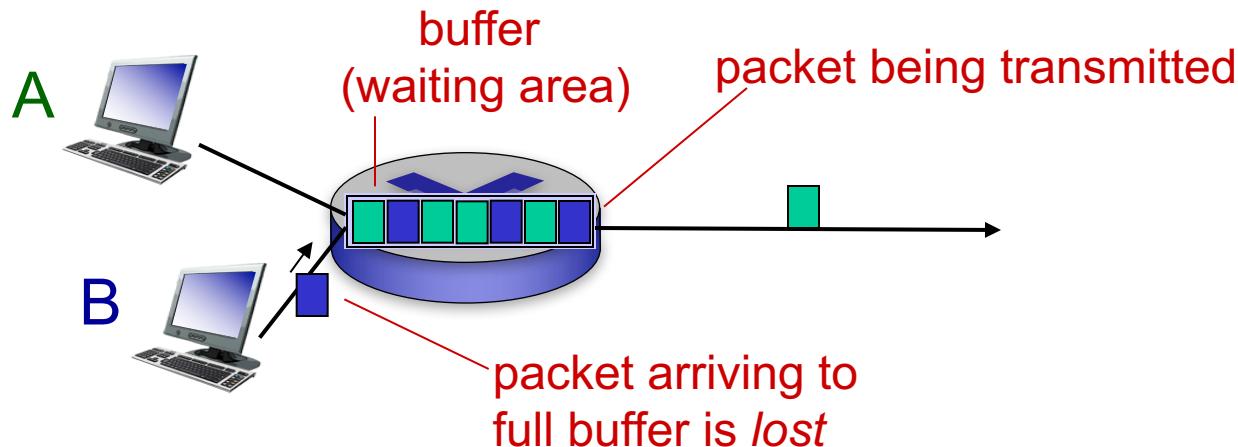
1	cs-gw (128.119.240.254)	1 ms	1 ms	2 ms
2	border1-rt-fa5-1-0.gw.umass.edu (128.119.3.145)	1 ms	1 ms	2 ms
3	cht-vbns.gw.umass.edu (128.119.3.130)	6 ms	5 ms	5 ms
4	jn1-at1-0-0-19.wor.vbns.net (204.147.132.129)	16 ms	11 ms	13 ms
5	jn1-so7-0-0-0.wae.vbns.net (204.147.136.136)	21 ms	18 ms	18 ms
6	abilene-vbns.abilene.ucaid.edu (198.32.11.9)	22 ms	18 ms	22 ms
7	nycm-wash.abilene.ucaid.edu (198.32.8.46)	22 ms	22 ms	22 ms
8	62.40.103.253 (62.40.103.253)	104 ms	109 ms	106 ms
9	de2-1.de1.de.geant.net (62.40.96.129)	109 ms	102 ms	104 ms
10	de.fr1.fr.geant.net (62.40.96.50)	113 ms	121 ms	114 ms
11	renater-gw.fr1.fr.geant.net (62.40.103.54)	112 ms	114 ms	112 ms
12	nio-n2.cssi.renater.fr (193.51.206.13)	111 ms	114 ms	116 ms
13	nice.cssi.renater.fr (195.220.98.102)	123 ms	125 ms	124 ms
14	r3t2-nice.cssi.renater.fr (195.220.98.110)	126 ms	126 ms	124 ms
15	eurecom-valbonne.r3t2.ft.net (193.48.50.54)	135 ms	128 ms	133 ms
16	194.214.211.25 (194.214.211.25)	126 ms	128 ms	126 ms
17	***			
18	***	* means no response (probe lost, router not replying)		
19	fantasia.eurecom.fr (193.55.113.142)	132 ms	128 ms	136 ms

trans-oceanic link

* Do some traceroutes from exotic countries at www.traceroute.org

Packet loss

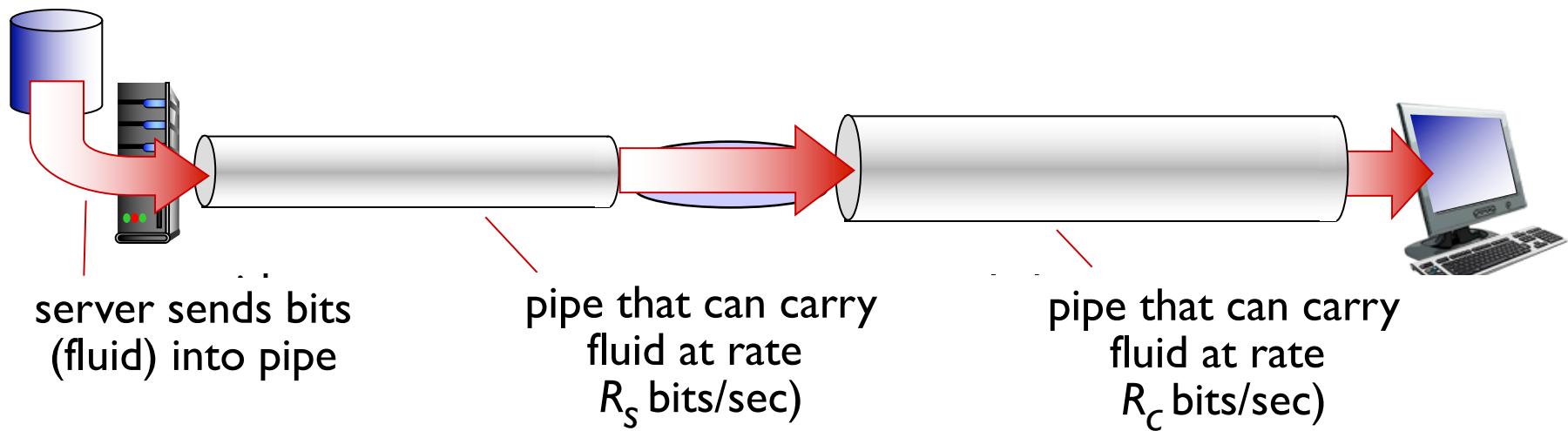
- queue (aka buffer) preceding link in buffer has finite capacity
- packet arriving to full queue dropped (aka lost)
- lost packet may be retransmitted by previous node, by source end system, or not at all



* Check out the Java applet for an interactive animation on queuing and loss

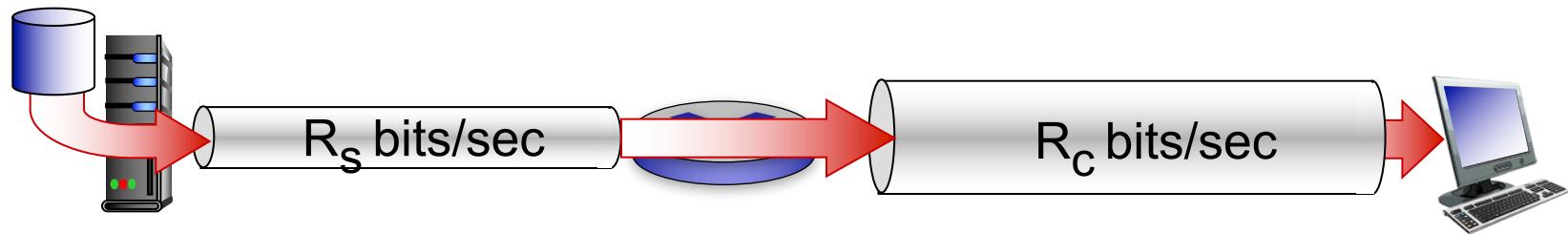
Throughput

- **throughput:** rate (bits/time unit) at which bits transferred between sender/receiver
 - *instantaneous:* rate at given point in time
 - *average:* rate over longer period of time

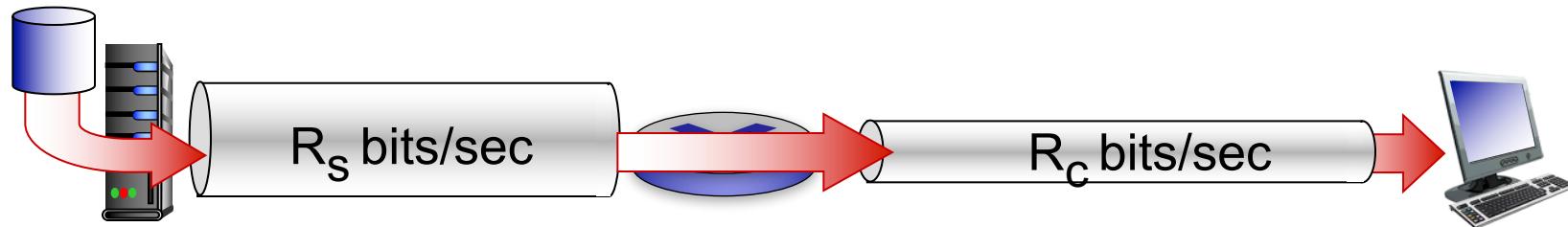


Throughput (more)

- $R_s < R_c$ What is average end-end throughput?



- $R_s > R_c$ What is average end-end throughput?

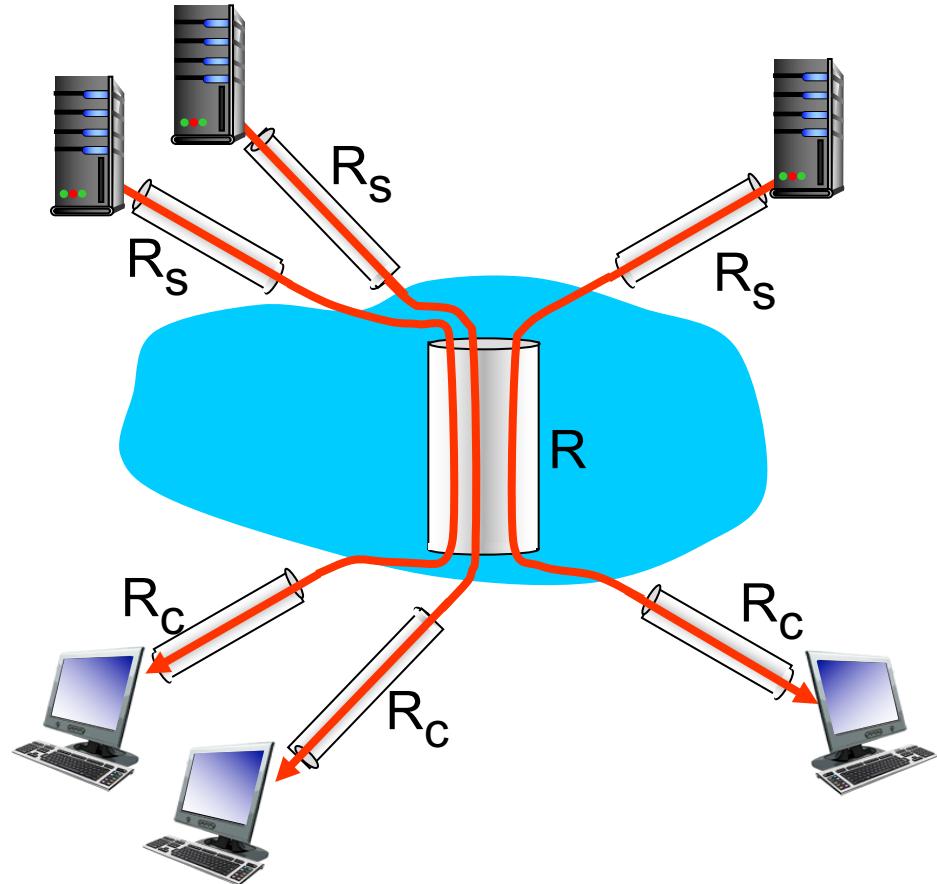


bottleneck link

link on end-end path that constrains end-end throughput

Throughput: Internet scenario

- per-connection end-end throughput:
 $\min(R_c, R_s, R/10)$
- in practice: R_c or R_s is often bottleneck



10 connections (fairly) share backbone bottleneck link R bits/sec

* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

Chapter I: roadmap

I.1 what *is* the Internet?

I.2 network edge

- end systems, access networks, links

I.3 network core

- packet switching, circuit switching, network structure

I.4 delay, loss, throughput in networks

I.5 protocol layers, service models

I.6 networks under attack: security

I.7 history

Protocol “layers”

*Networks are complex,
with many “pieces”:*

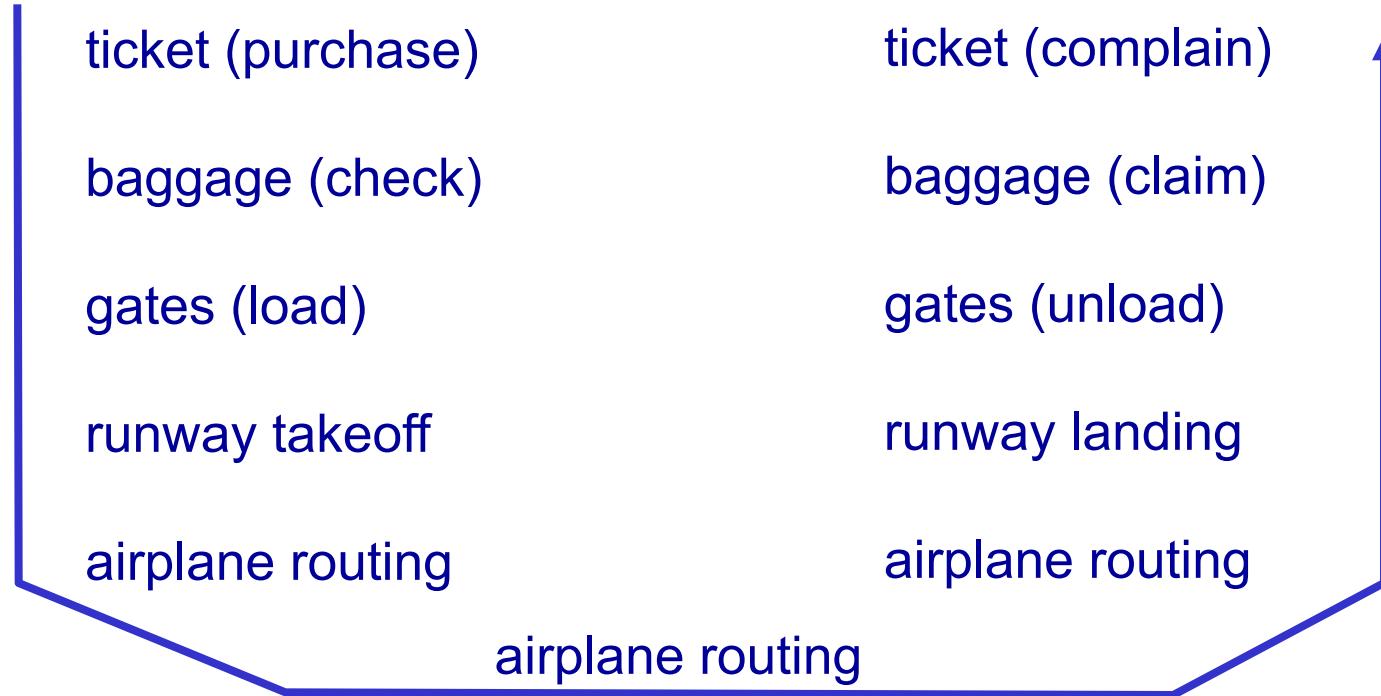
- hosts
- routers
- links of various media
- applications
- protocols
- hardware, software

Question:

is there any hope of
organizing structure of
network?

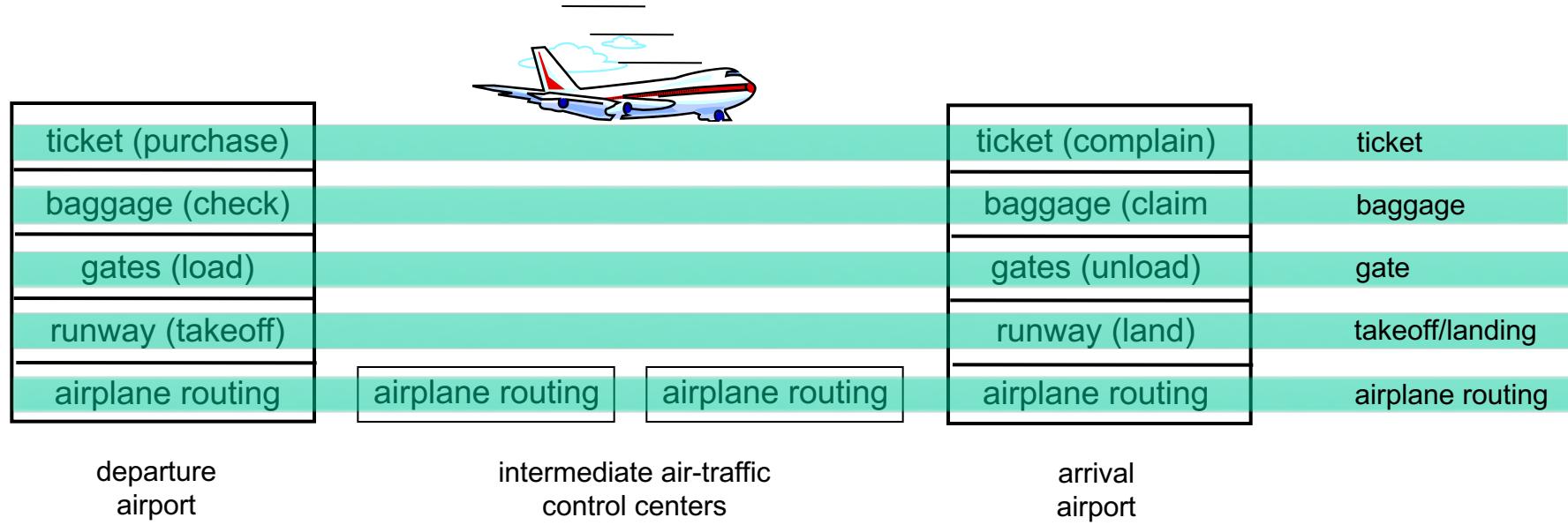
.... or at least our
discussion of networks?

Organization of air travel



- a series of steps

Layering of airline functionality



layers: each layer implements a service

- via its own internal-layer actions
 - relying on services provided by layer below

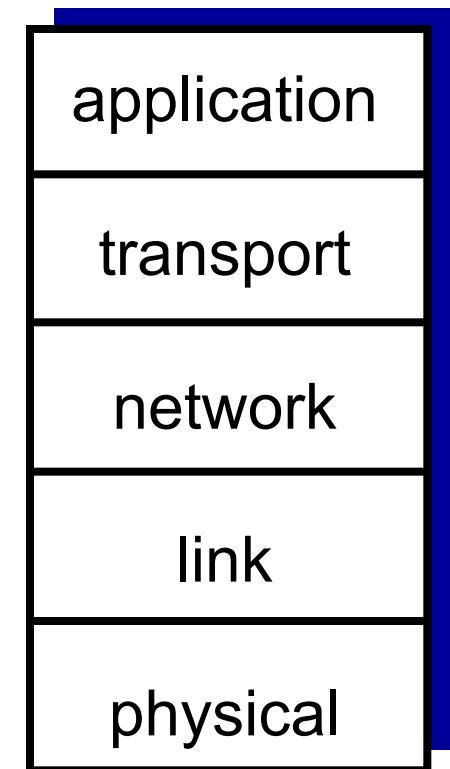
Why layering?

dealing with complex systems:

- explicit structure allows identification, relationship of complex system's pieces
 - layered *reference model* for discussion
- modularization eases maintenance, updating of system
 - change of implementation of layer's service transparent to rest of system
 - e.g., change in gate procedure doesn't affect rest of system
- layering considered harmful?

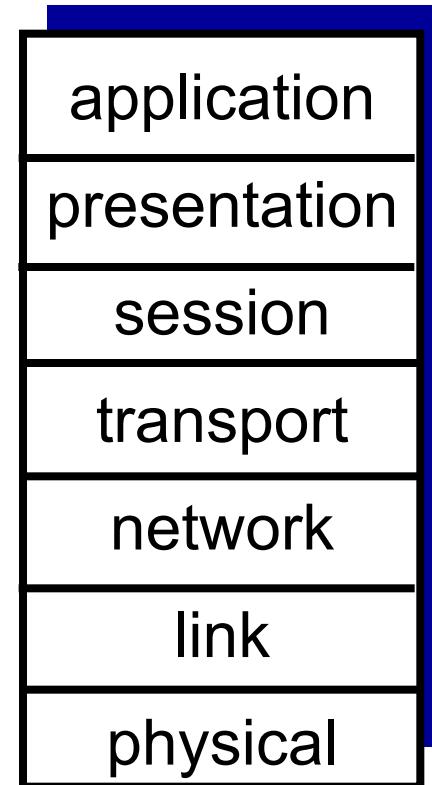
Internet protocol stack

- *application*: supporting network applications
 - FTP, SMTP, HTTP
- *transport*: process-process data transfer
 - TCP, UDP
- *network*: routing of datagrams from source to destination
 - IP, routing protocols
- *link*: data transfer between neighboring network elements
 - Ethernet, 802.111 (WiFi), PPP
- *physical*: bits “on the wire”



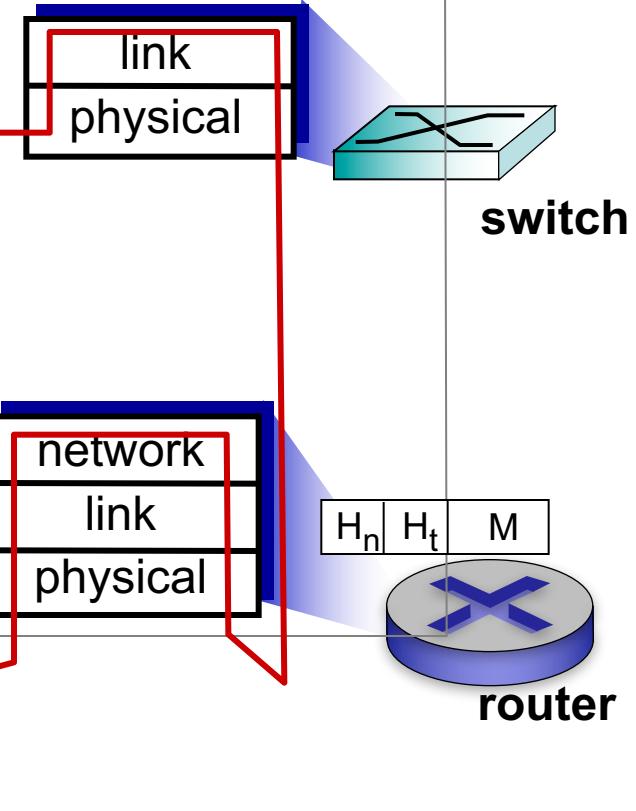
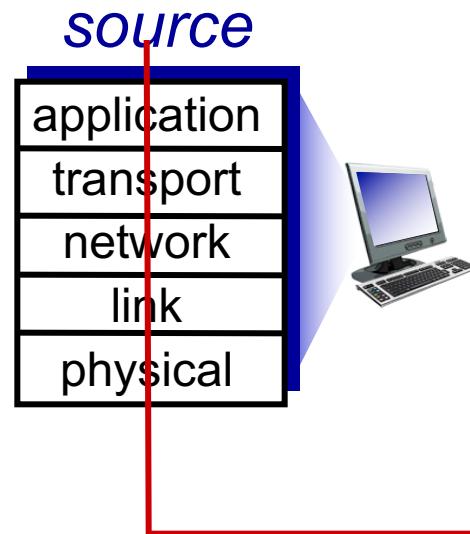
ISO/OSI reference model

- ***presentation:*** allow applications to interpret meaning of data, e.g., encryption, compression, machine-specific conventions
- ***session:*** synchronization, checkpointing, recovery of data exchange
- Internet stack “missing” these layers!
 - these services, *if needed*, must be implemented in application
 - needed?



Encapsulation

message	M
segment	H _t M
datagram	H _n H _t M
frame	H _l H _n H _t M



Chapter I: roadmap

I.1 what *is* the Internet?

I.2 network edge

- end systems, access networks, links

I.3 network core

- packet switching, circuit switching, network structure

I.4 delay, loss, throughput in networks

I.5 protocol layers, service models

I.6 networks under attack: security

I.7 history

Network security

- field of network security:
 - how bad guys can attack computer networks
 - how we can defend networks against attacks
 - how to design architectures that are immune to attacks
- Internet not originally designed with (much) security in mind
 - *original vision:* “a group of mutually trusting users attached to a transparent network” ☺
 - Internet protocol designers playing “catch-up”
 - security considerations in all layers!

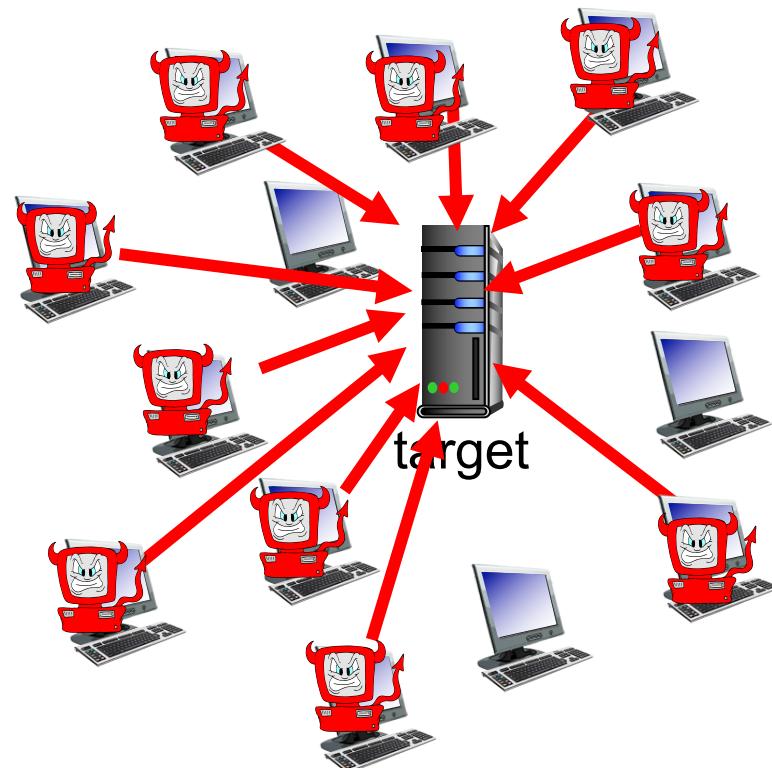
Bad guys: put malware into hosts via Internet

- malware can get in host from:
 - *virus*: self-replicating infection by receiving/executing object (e.g., e-mail attachment)
 - *worm*: self-replicating infection by passively receiving object that gets itself executed
- **spyware malware** can record keystrokes, web sites visited, upload info to collection site
- infected host can be enrolled in **botnet**, used for spam, DDoS attacks

Bad guys: attack server, network infrastructure

Denial of Service (DoS): attackers make resources (server, bandwidth) unavailable to legitimate traffic by overwhelming resource with bogus traffic

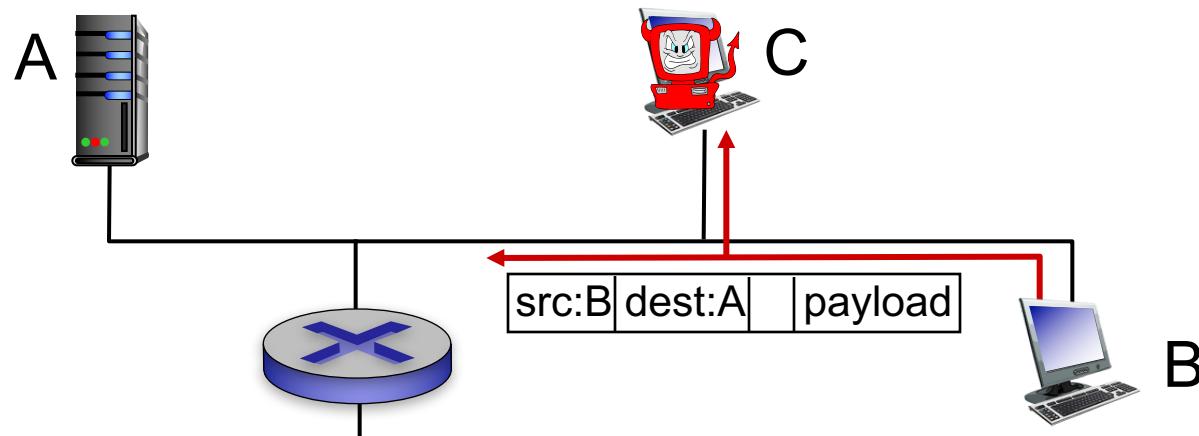
1. select target
2. break into hosts around the network (see botnet)
3. send packets to target from compromised hosts



Bad guys can sniff packets

packet “sniffing”:

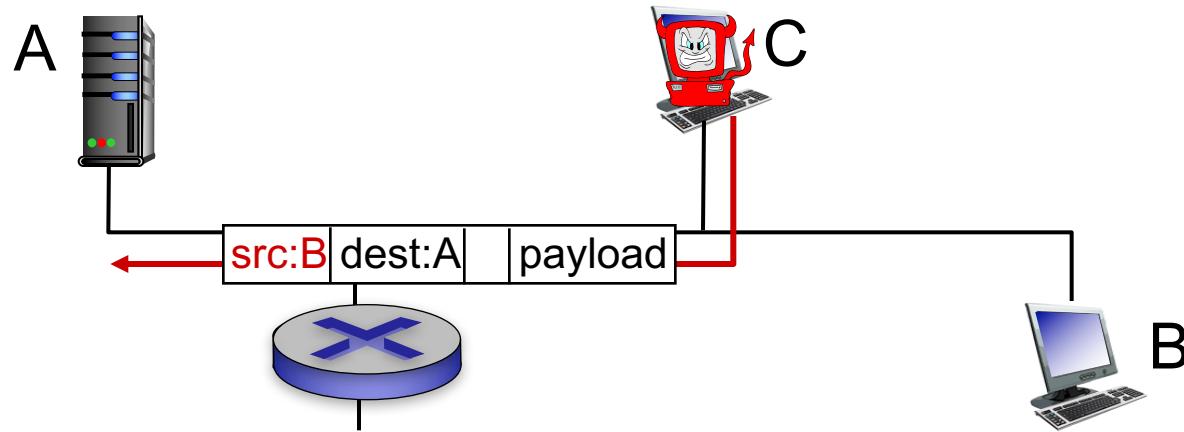
- broadcast media (shared Ethernet, wireless)
- promiscuous network interface reads/records all packets (e.g., including passwords!) passing by



- wireshark software used for end-of-chapter labs is a (free) packet-sniffer

Bad guys can use fake addresses

IP spoofing: send packet with false source address



... lots more on security (throughout, Chapter 8)

Chapter I: roadmap

I.1 what *is* the Internet?

I.2 network edge

- end systems, access networks, links

I.3 network core

- packet switching, circuit switching, network structure

I.4 delay, loss, throughput in networks

I.5 protocol layers, service models

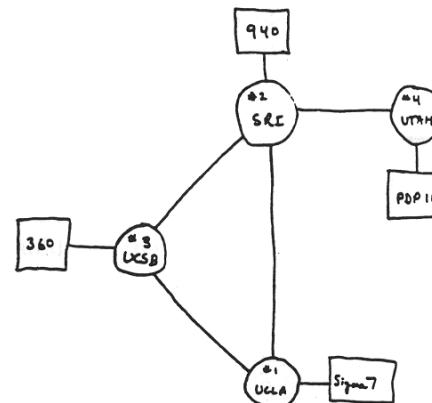
I.6 networks under attack: security

I.7 history

Internet history

1961-1972: Early packet-switching principles

- 1961: Kleinrock - queueing theory shows effectiveness of packet-switching
- 1964: Baran - packet-switching in military nets
- 1967: ARPAnet conceived by Advanced Research Projects Agency
- 1969: first ARPAnet node operational
- 1972:
 - ARPAnet public demo
 - NCP (Network Control Protocol) first host-host protocol
 - first e-mail program
 - ARPAnet has 15 nodes



Internet history

1972-1980: Internetworking, new and proprietary nets

- 1970: ALOHAnet satellite network in Hawaii
- 1974: Cerf and Kahn - architecture for interconnecting networks
- 1976: Ethernet at Xerox PARC
- late 70' s: proprietary architectures: DECnet, SNA, XNA
- late 70' s: switching fixed length packets (ATM precursor)
- 1979: ARPAnet has 200 nodes

Cerf and Kahn's
internetworking principles:

- minimalism, autonomy - no internal changes required to interconnect networks
- best effort service model
- stateless routers
- decentralized control

define today's Internet architecture

Internet history

1980-1990: new protocols, a proliferation of networks

- 1983: deployment of TCP/IP
- 1982: smtp e-mail protocol defined
- 1983: DNS defined for name-to-IP-address translation
- 1985: ftp protocol defined
- 1988: TCP congestion control
- new national networks: CSnet, BITnet, NSFnet, Minitel
- 100,000 hosts connected to confederation of networks

Internet history

1990, 2000's: commercialization, the Web, new apps

- early 1990's: ARPAnet decommissioned
- 1991: NSF lifts restrictions on commercial use of NSFnet (decommissioned, 1995)
- early 1990s: Web
 - hypertext [Bush 1945, Nelson 1960's]
 - HTML, HTTP: Berners-Lee
 - 1994: Mosaic, later Netscape
 - late 1990's:
commercialization of the Web

late 1990's – 2000's:

- more killer apps: instant messaging, P2P file sharing
- network security to forefront
- est. 50 million host, 100 million+ users
- backbone links running at Gbps

Internet history

2005-present

- ~5B devices attached to Internet (2016)
 - smartphones and tablets
- aggressive deployment of broadband access
- increasing ubiquity of high-speed wireless access
- emergence of online social networks:
 - Facebook: ~ one billion users
- service providers (Google, Microsoft) create their own networks
 - bypass Internet, providing “instantaneous” access to search, video content, email, etc.
- e-commerce, universities, enterprises running their services in “cloud” (e.g., Amazon EC2)

Introduction: summary

covered a “ton” of material!

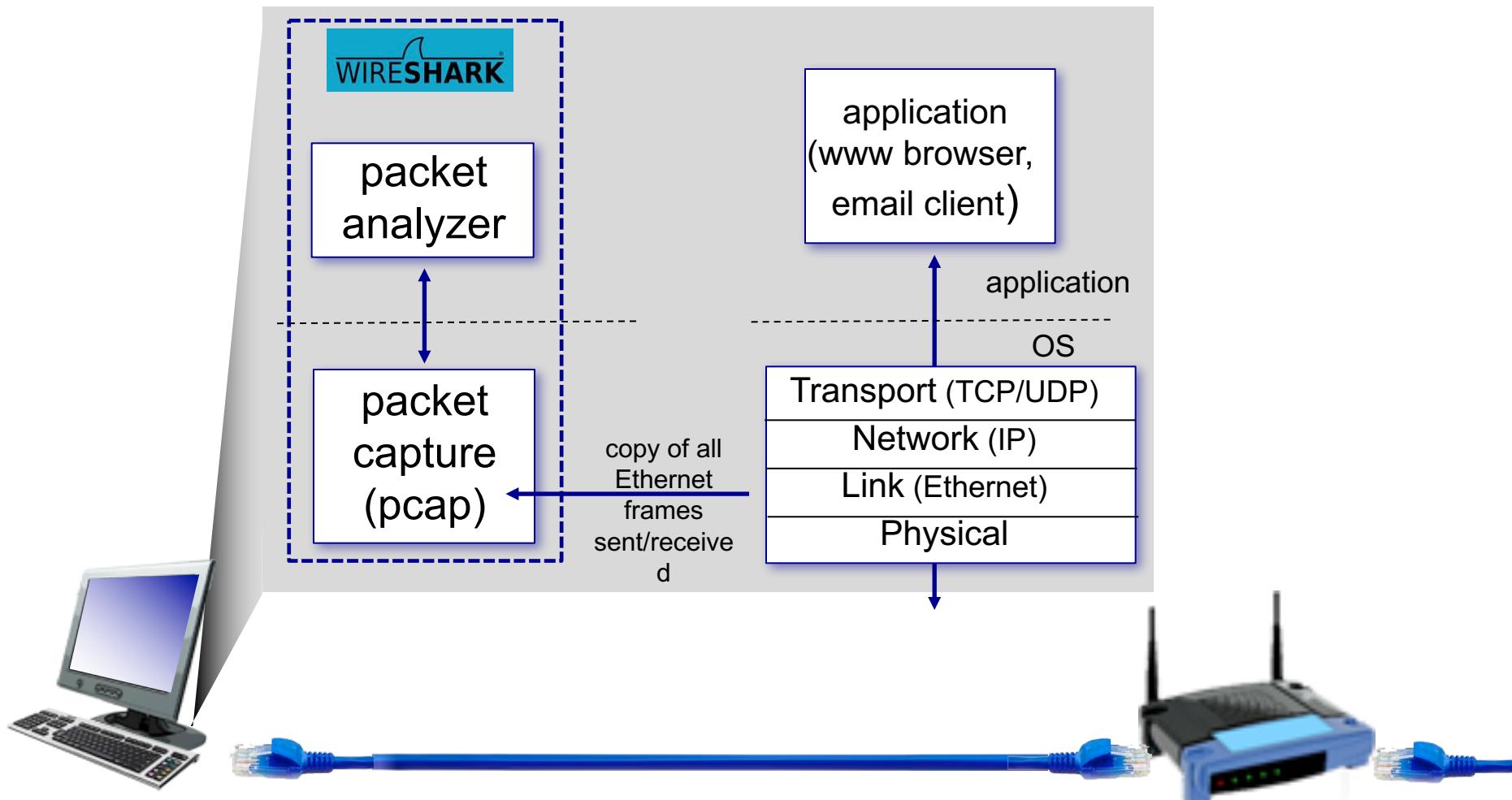
- Internet overview
- what’s a protocol?
- network edge, core, access network
 - packet-switching versus circuit-switching
 - Internet structure
- performance: loss, delay, throughput
- layering, service models
- security
- history

you now have:

- context, overview, “feel” of networking
- more depth, detail to follow!

Chapter I

Additional Slides



Chapter 2

Application Layer

A note on the use of these Powerpoint slides:

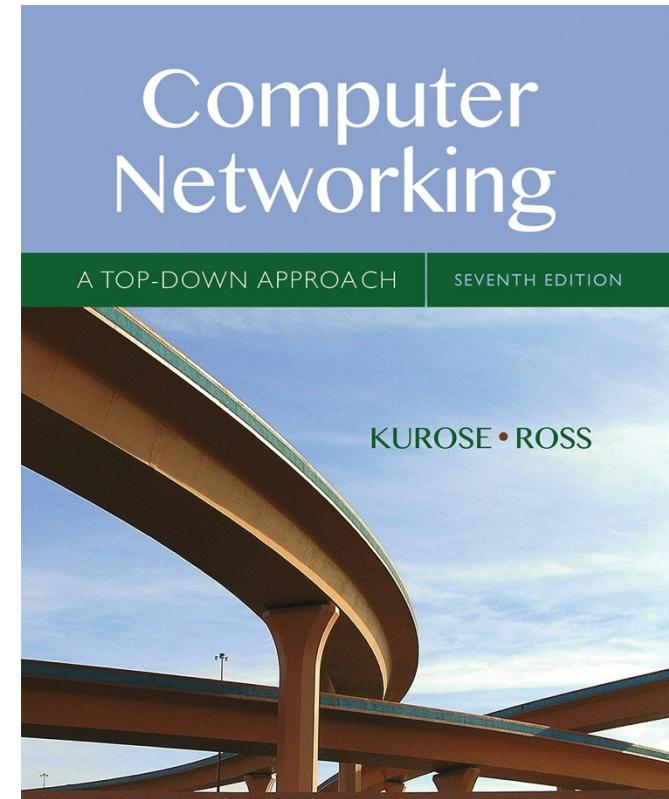
We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR

© All material copyright 1996-2016

J.F Kurose and K.W. Ross, All Rights Reserved



*Computer
Networking: A Top
Down Approach*

7th edition

Jim Kurose, Keith Ross
Pearson/Addison Wesley
April 2016

Chapter 2: outline

2.1 principles of network applications

2.2 Web and HTTP

2.3 electronic mail

- SMTP, POP3, IMAP

2.4 DNS

2.5 P2P applications

2.6 video streaming and content distribution networks

2.7 socket programming with UDP and TCP

Chapter 2: application layer

our goals:

- conceptual, implementation aspects of network application protocols
 - transport-layer service models
 - client-server paradigm
 - peer-to-peer paradigm
 - content distribution networks
- learn about protocols by examining popular application-level protocols
 - HTTP
 - FTP
 - SMTP / POP3 / IMAP
 - DNS
- creating network applications
 - socket API

Some network apps

- e-mail
- web
- text messaging
- remote login
- P2P file sharing
- multi-user network games
- streaming stored video (YouTube, Hulu, Netflix)
- voice over IP (e.g., Skype)
- real-time video conferencing
- social networking
- search
- ...
- ...

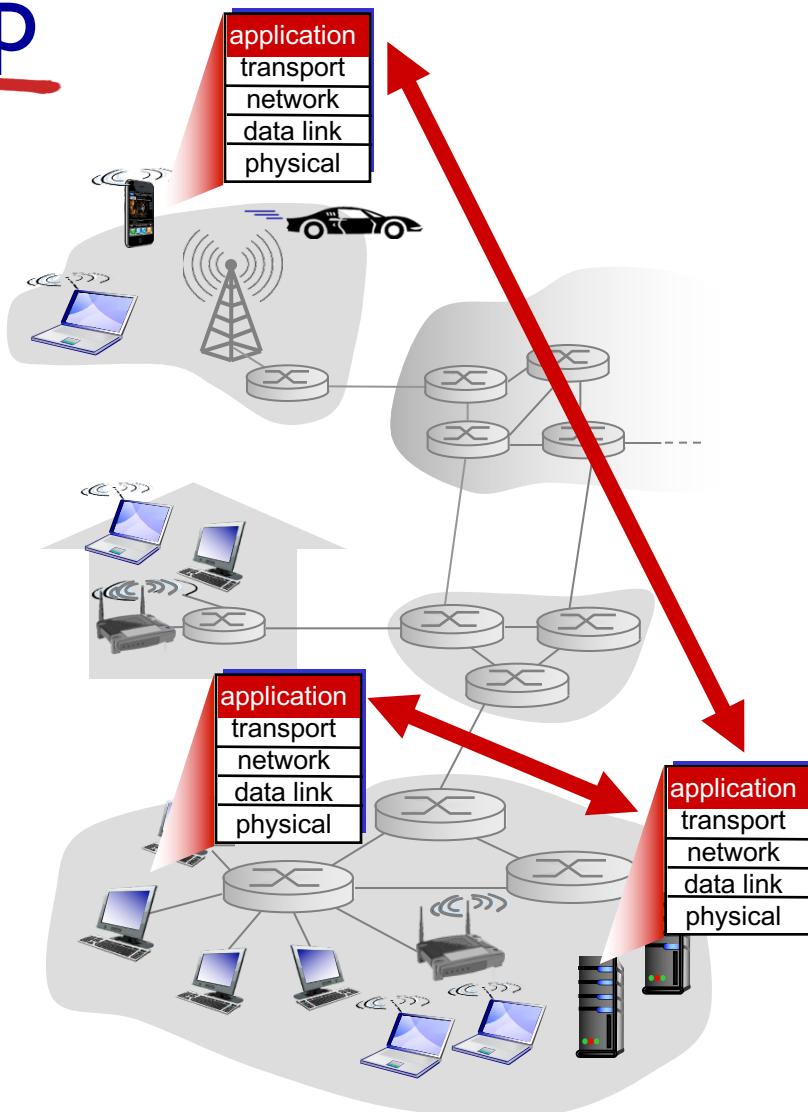
Creating a network app

write programs that:

- run on (different) end systems
- communicate over network
- e.g., web server software communicates with browser software

no need to write software for network-core devices

- network-core devices do not run user applications
- applications on end systems allows for rapid app development, propagation

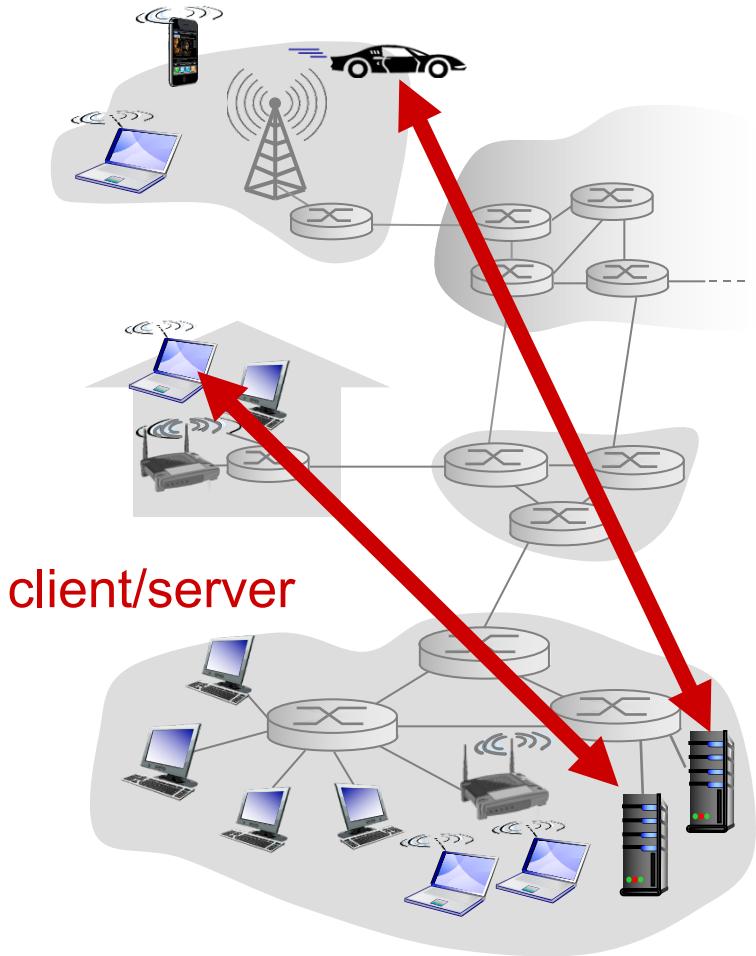


Application architectures

possible structure of applications:

- client-server
- peer-to-peer (P2P)

Client-server architecture



server:

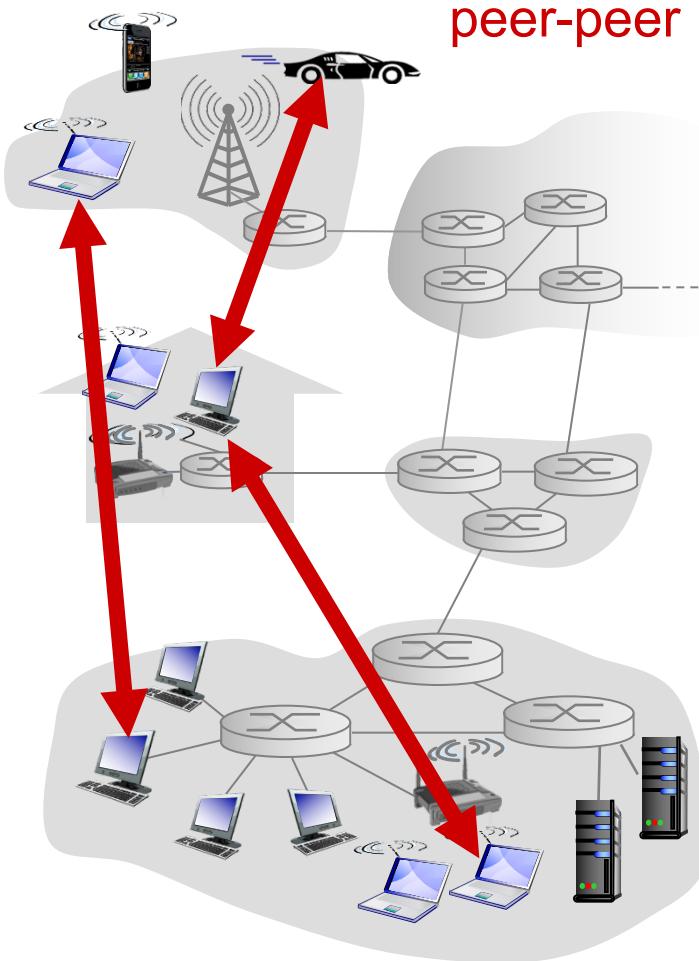
- always-on host
- permanent IP address
- data centers for scaling

clients:

- communicate with server
- may be intermittently connected
- may have dynamic IP addresses
- do not communicate directly with each other

P2P architecture

- no always-on server
- arbitrary end systems directly communicate
- peers request service from other peers, provide service in return to other peers
 - *self scalability* – new peers bring new service capacity, as well as new service demands
- peers are intermittently connected and change IP addresses
 - complex management



Processes communicating

process: program running within a host

- within same host, two processes communicate using **inter-process communication** (defined by OS)
- processes in different hosts communicate by exchanging **messages**

clients, servers

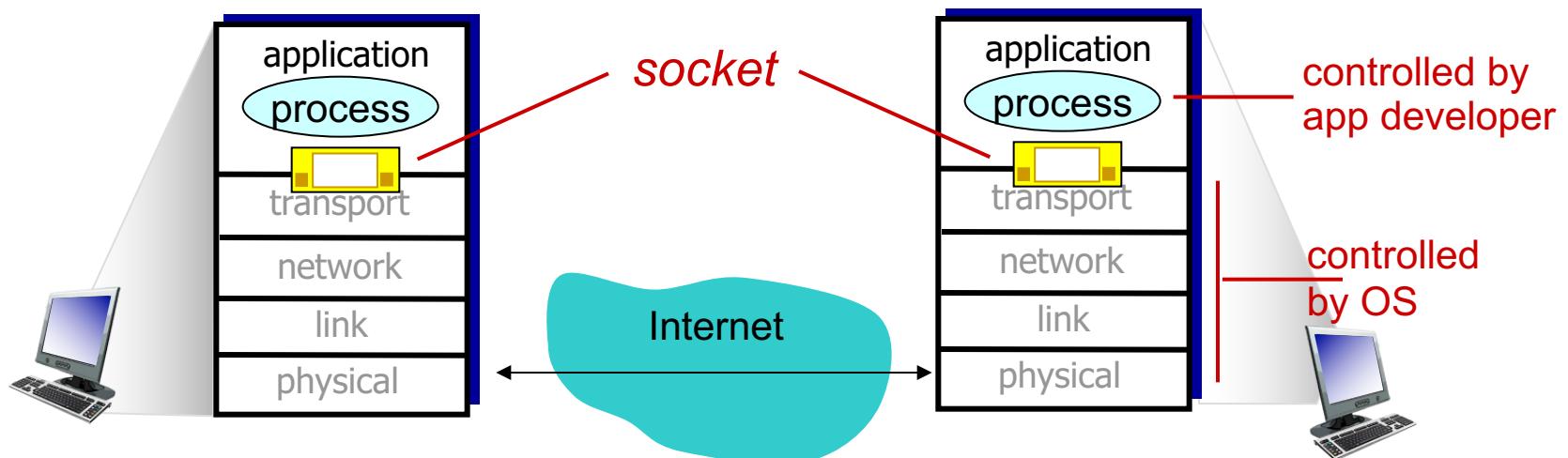
client process: process that initiates communication

server process: process that waits to be contacted

- aside: applications with P2P architectures have client processes & server processes

Sockets

- process sends/receives messages to/from its **socket**
- socket analogous to door
 - sending process shoves message out door
 - sending process relies on transport infrastructure on other side of door to deliver message to socket at receiving process



Addressing processes

- to receive messages, process must have *identifier*
- host device has unique 32-bit IP address
- *Q:* does IP address of host on which process runs suffice for identifying the process?
 - *A:* no, many processes can be running on same host
- *identifier* includes both IP address and port numbers associated with process on host.
- example port numbers:
 - HTTP server: 80
 - mail server: 25
- to send HTTP message to gaia.cs.umass.edu web server:
 - IP address: 128.119.245.12
 - port number: 80
- more shortly...

App-layer protocol defines

- types of messages exchanged,
 - e.g., request, response
- message syntax:
 - what fields in messages & how fields are delineated
- message semantics
 - meaning of information in fields
- rules for when and how processes send & respond to messages

open protocols:

- defined in RFCs
- allows for interoperability
- e.g., HTTP, SMTP

proprietary protocols:

- e.g., Skype

What transport service does an app need?

data integrity

- some apps (e.g., file transfer, web transactions) require 100% reliable data transfer
- other apps (e.g., audio) can tolerate some loss

timing

- some apps (e.g., Internet telephony, interactive games) require low delay to be “effective”

throughput

- some apps (e.g., multimedia) require minimum amount of throughput to be “effective”
- other apps (“elastic apps”) make use of whatever throughput they get

security

- encryption, data integrity, ...

Transport service requirements: common apps

application	data loss	throughput	time sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	no loss	elastic	no
real-time audio/video	loss-tolerant	audio: 5kbps-1Mbps video:10kbps-5Mbps	yes, 100' s msec
stored audio/video	loss-tolerant	same as above	
interactive games	loss-tolerant	few kbps up	yes, few secs
text messaging	no loss	elastic	yes, 100' s msec yes and no

Internet transport protocols services

TCP service:

- *reliable transport* between sending and receiving process
- *flow control*: sender won't overwhelm receiver
- *congestion control*: throttle sender when network overloaded
- *does not provide*: timing, minimum throughput guarantee, security
- *connection-oriented*: setup required between client and server processes

UDP service:

- *unreliable data transfer* between sending and receiving process
- *does not provide*: reliability, flow control, congestion control, timing, throughput guarantee, security, or connection setup,

Q: why bother? Why is there a UDP?

Internet apps: application, transport protocols

application	application layer protocol	underlying transport protocol
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	HTTP (e.g., YouTube), RTP [RFC 1889]	TCP or UDP
Internet telephony	SIP, RTP, proprietary (e.g., Skype)	TCP or UDP

Securing TCP

TCP & UDP

- no encryption
- cleartext passwords sent into socket traverse Internet in cleartext

SSL

- provides encrypted TCP connection
- data integrity
- end-point authentication

SSL is at app layer

- apps use SSL libraries, that “talk” to TCP

SSL socket API

- cleartext passwords sent into socket traverse Internet encrypted
- see Chapter 8

Chapter 2: outline

2.1 principles of network
applications

2.2 Web and HTTP

2.3 electronic mail

- SMTP, POP3, IMAP

2.4 DNS

2.5 P2P applications

2.6 video streaming and
content distribution
networks

2.7 socket programming
with UDP and TCP

Web and HTTP

First, a review...

- *web page* consists of *objects*
- object can be HTML file, JPEG image, Java applet, audio file,...
- web page consists of *base HTML-file* which includes *several referenced objects*
- each object is addressable by a *URL*, e.g.,

www.someschool.edu/someDept/pic.gif

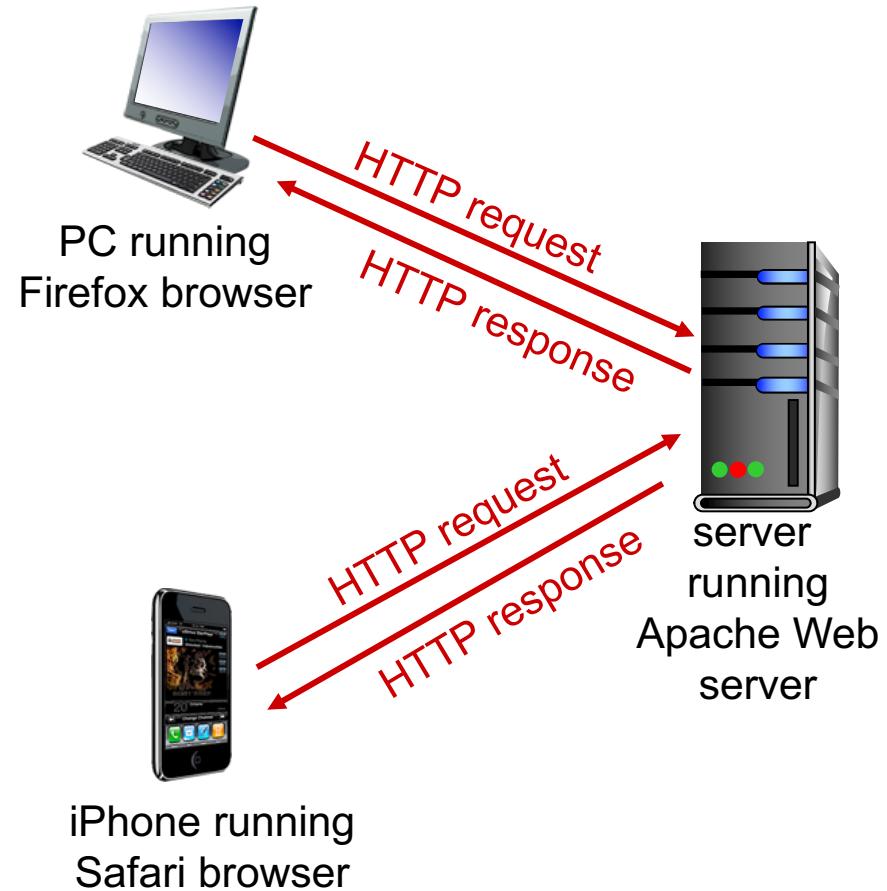
host name

path name

HTTP overview

HTTP: hypertext transfer protocol

- Web's application layer protocol
- client/server model
 - **client:** browser that requests, receives, (using HTTP protocol) and "displays" Web objects
 - **server:** Web server sends (using HTTP protocol) objects in response to requests



HTTP overview (continued)

uses TCP:

- client initiates TCP connection (creates socket) to server, port 80
- server accepts TCP connection from client
- HTTP messages (application-layer protocol messages) exchanged between browser (HTTP client) and Web server (HTTP server)
- TCP connection closed

HTTP is “stateless”

- server maintains no information about past client requests

aside

protocols that maintain “state” are complex!

- past history (state) must be maintained
- if server/client crashes, their views of “state” may be inconsistent, must be reconciled

HTTP connections

non-persistent HTTP

- at most one object sent over TCP connection
 - connection then closed
- downloading multiple objects required multiple connections

persistent HTTP

- multiple objects can be sent over single TCP connection between client, server

Non-persistent HTTP

suppose user enters URL:

`www.someSchool.edu/someDepartment/home.index`

(contains text,
references to 10
jpeg images)

Ia. HTTP client initiates TCP connection to HTTP server (process) at `www.someSchool.edu` on port 80

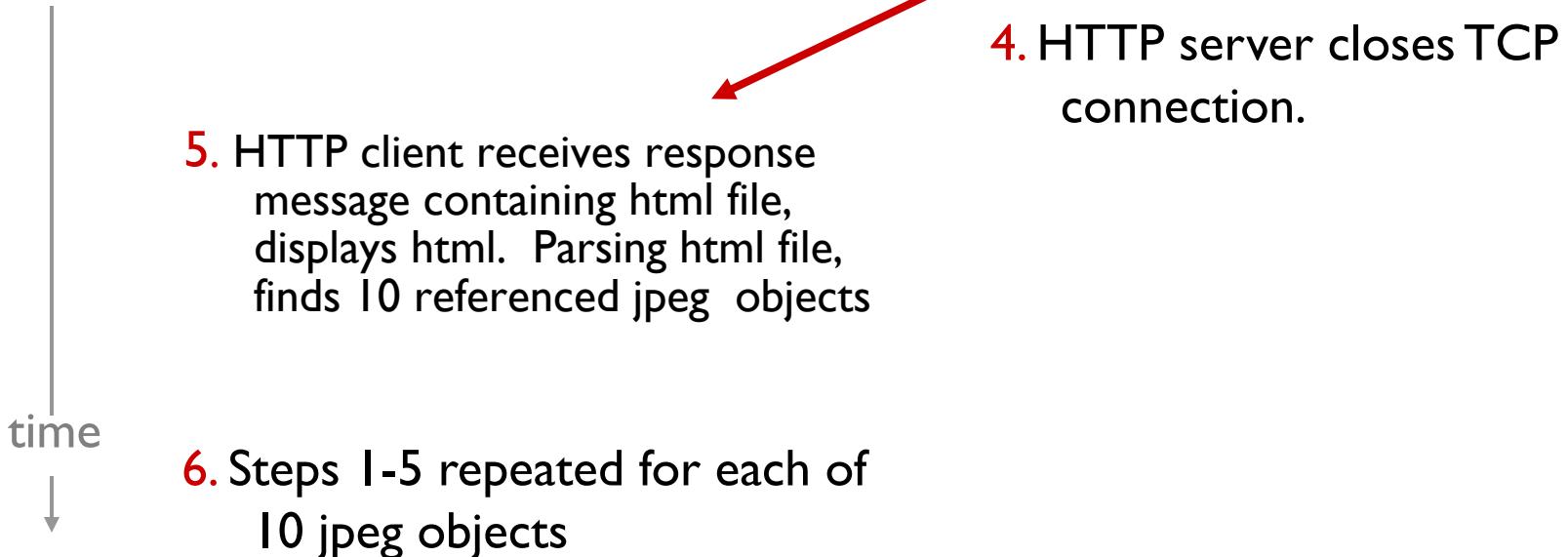
Ib. HTTP server at host `www.someSchool.edu` waiting for TCP connection at port 80. “accepts” connection, notifying client

2. HTTP client sends HTTP *request message* (containing URL) into TCP connection socket. Message indicates that client wants object `someDepartment/home.index`

3. HTTP server receives request message, forms *response message* containing requested object, and sends message into its socket

time
↓

Non-persistent HTTP (cont.)

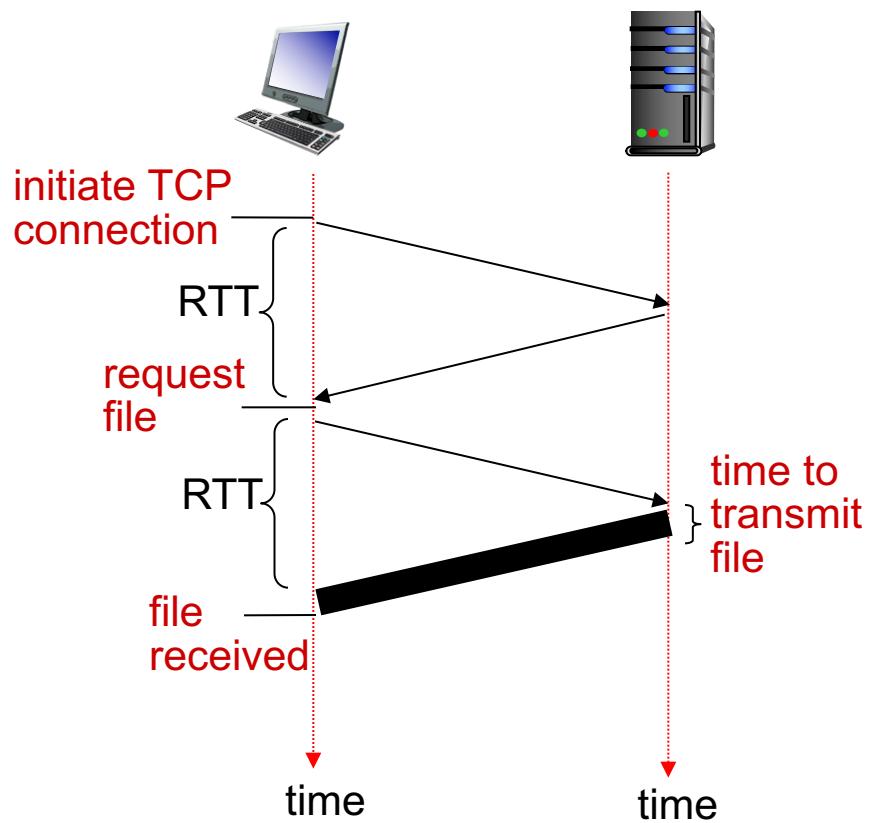


Non-persistent HTTP: response time

RTT (definition): time for a small packet to travel from client to server and back

HTTP response time:

- one RTT to initiate TCP connection
- one RTT for HTTP request and first few bytes of HTTP response to return
- file transmission time
- non-persistent HTTP response time = $2\text{RTT} + \text{file transmission time}$



Persistent HTTP

non-persistent HTTP issues:

- requires 2 RTTs per object
- OS overhead for each TCP connection
- browsers often open parallel TCP connections to fetch referenced objects

persistent HTTP:

- server leaves connection open after sending response
- subsequent HTTP messages between same client/server sent over open connection
- client sends requests as soon as it encounters a referenced object
- as little as one RTT for all the referenced objects

HTTP request message

- two types of HTTP messages: *request, response*
- **HTTP request message:**
 - ASCII (human-readable format)

request line
(GET, POST,
HEAD commands)

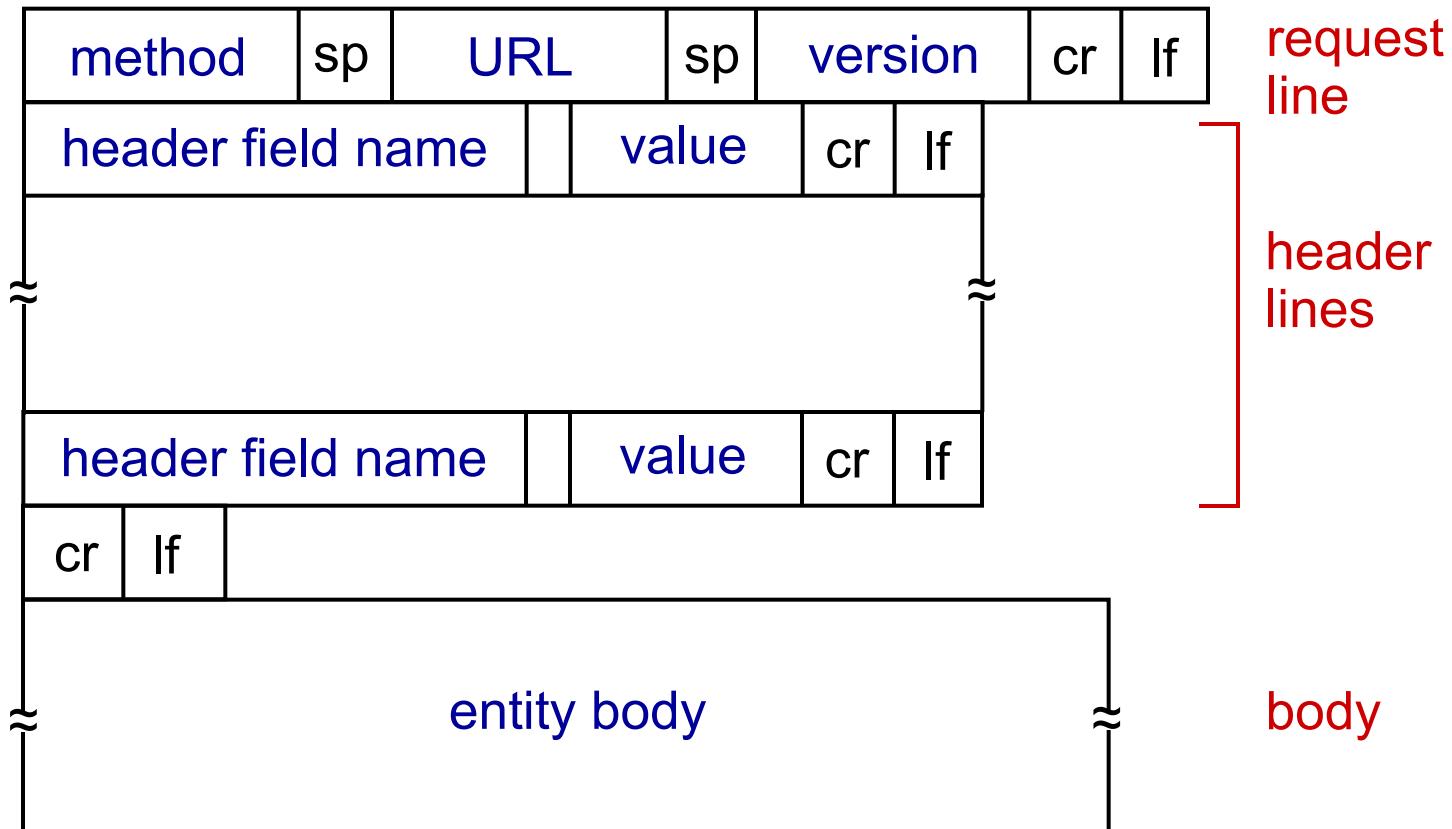
header
lines
carriage return,
line feed at start
of line indicates
end of header lines

```
GET /index.html HTTP/1.1\r\n
Host: www-net.cs.umass.edu\r\n
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n
```

carriage return character
line-feed character

* Check out the online interactive exercises for more
examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

HTTP request message: general format



Uploading form input

POST method:

- web page often includes form input
- input is uploaded to server in entity body

URL method:

- uses GET method
- input is uploaded in URL field of request line:

`www.somesite.com/animalsearch?monkeys&banana`

Method types

HTTP/1.0:

- GET
- POST
- HEAD
 - asks server to leave requested object out of response

HTTP/1.1:

- GET, POST, HEAD
- PUT
 - uploads file in entity body to path specified in URL field
- DELETE
 - deletes file specified in the URL field

HTTP response message

status line

(protocol

status code

status phrase)

header
lines

data, e.g.,
requested
HTML file

```
HTTP/1.1 200 OK\r\nDate: Sun, 26 Sep 2010 20:09:20 GMT\r\nServer: Apache/2.0.52 (CentOS)\r\nLast-Modified: Tue, 30 Oct 2007 17:00:02  
GMT\r\nETag: "17dc6-a5c-bf716880"\r\nAccept-Ranges: bytes\r\nContent-Length: 2652\r\nKeep-Alive: timeout=10, max=100\r\nConnection: Keep-Alive\r\nContent-Type: text/html; charset=ISO-8859-  
1\r\n\r\ndata data data data data ...
```

* Check out the online interactive exercises for more
examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

HTTP response status codes

- status code appears in 1st line in server-to-client response message.
- some sample codes:

200 OK

- request succeeded, requested object later in this msg

301 Moved Permanently

- requested object moved, new location specified later in this msg
(Location:)

400 Bad Request

- request msg not understood by server

404 Not Found

- requested document not found on this server

505 HTTP Version Not Supported

Trying out HTTP (client side) for yourself

I. Telnet to your favorite Web server:

```
telnet gaia.cs.umass.edu 80
```

opens TCP connection to port 80
(default HTTP server port)
at gaia.cs.umass.edu.
anything typed in will be sent
to port 80 at gaia.cs.umass.edu

2. type in a GET HTTP request:

```
GET /kurose_ross/interactive/index.php HTTP/1.1
```

```
Host: gaia.cs.umass.edu
```

by typing this in (hit carriage return twice), you send this minimal (but complete) GET request to HTTP server

3. look at response message sent by HTTP server!

(or use Wireshark to look at captured HTTP request/response)

User-server state: cookies

many Web sites use cookies

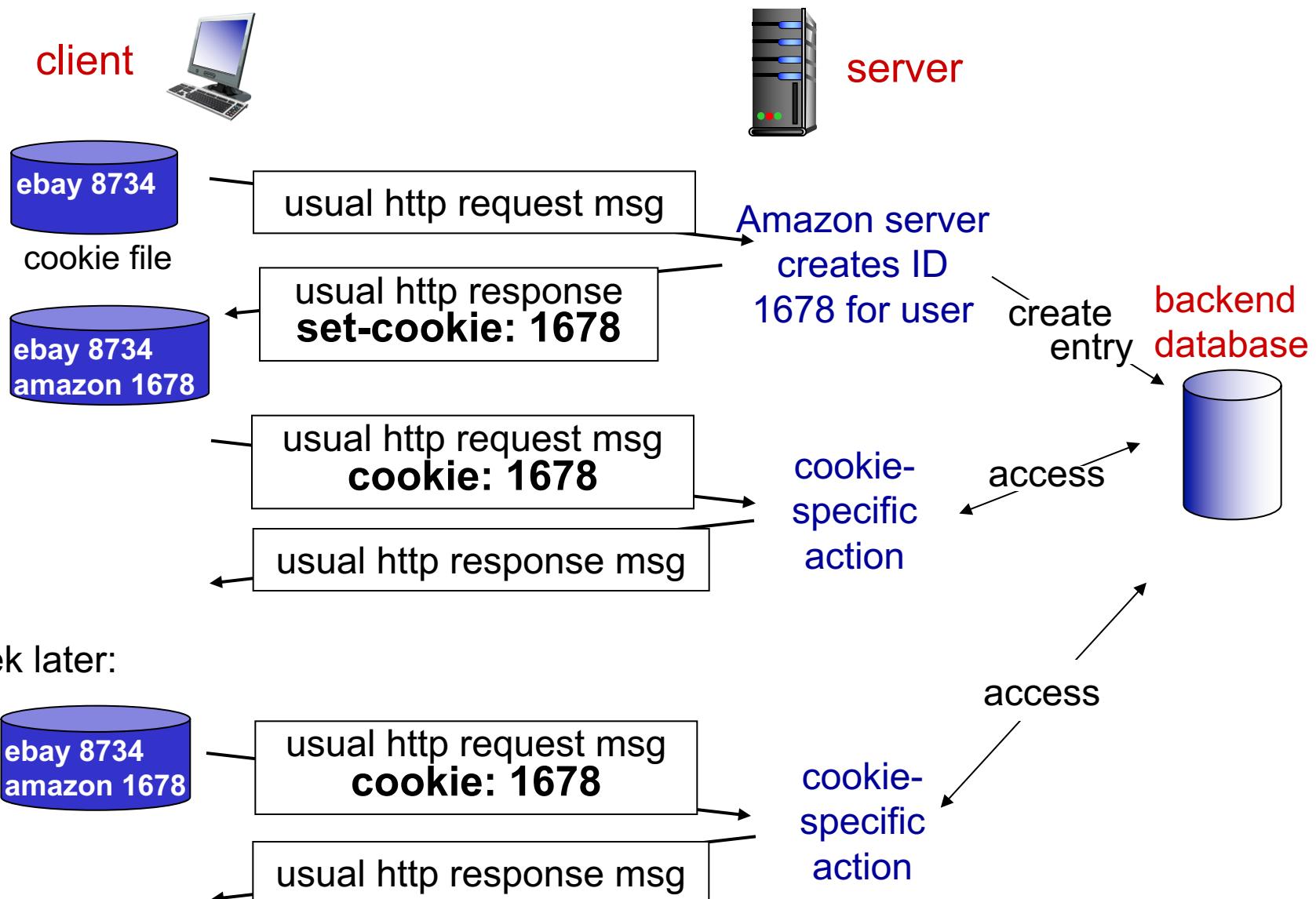
four components:

- 1) cookie header line of
HTTP *response*
message
- 2) cookie header line in
next HTTP *request*
message
- 3) cookie file kept on
user's host, managed
by user's browser
- 4) back-end database at
Web site

example:

- Susan always access Internet from PC
- visits specific e-commerce site for first time
- when initial HTTP requests arrives at site, site creates:
 - unique ID
 - entry in backend database for ID

Cookies: keeping “state” (cont.)



Cookies (continued)

*what cookies can be used
for:*

- authorization
- shopping carts
- recommendations
- user session state (Web e-mail)

aside

cookies and privacy:

- cookies permit sites to learn a lot about you
- you may supply name and e-mail to sites

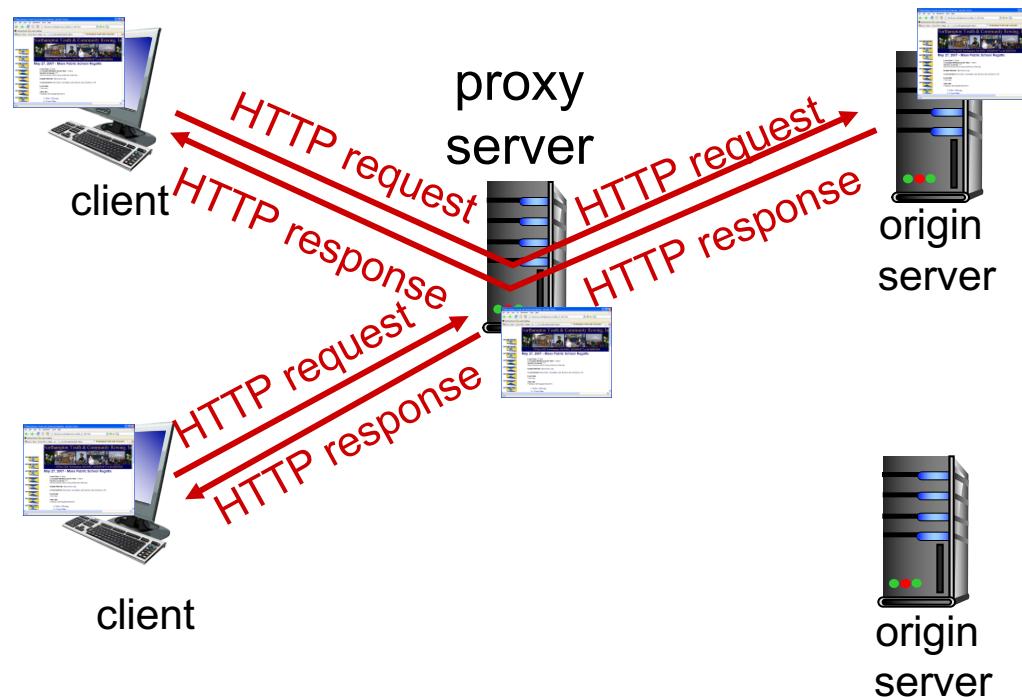
how to keep “state”:

- protocol endpoints: maintain state at sender/receiver over multiple transactions
- cookies: http messages carry state

Web caches (proxy server)

goal: satisfy client request without involving origin server

- user sets browser: Web accesses via cache
- browser sends all HTTP requests to cache
 - object in cache: cache returns object
 - else cache requests object from origin server, then returns object to client



More about Web caching

- cache acts as both client and server
 - server for original requesting client
 - client to origin server
- typically cache is installed by ISP (university, company, residential ISP)

why Web caching?

- reduce response time for client request
- reduce traffic on an institution's access link
- Internet dense with caches: enables “poor” content providers to effectively deliver content (so too does P2P file sharing)

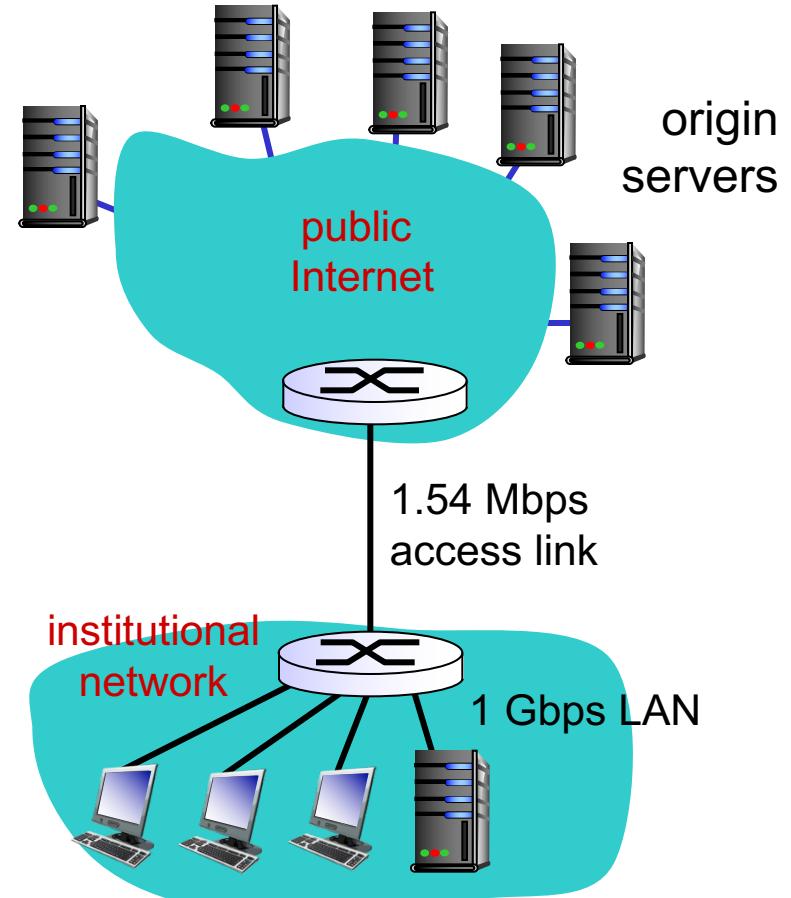
Caching example:

assumptions:

- avg object size: 100K bits
- avg request rate from browsers to origin servers: 15/sec
- avg data rate to browsers: 1.50 Mbps
- RTT from institutional router to any origin server: 2 sec
- access link rate: 1.54 Mbps

consequences:

- LAN utilization: 15% *problem!*
- access link utilization = **99%**
- total delay = Internet delay + access delay + LAN delay
= 2 sec + minutes + usecs



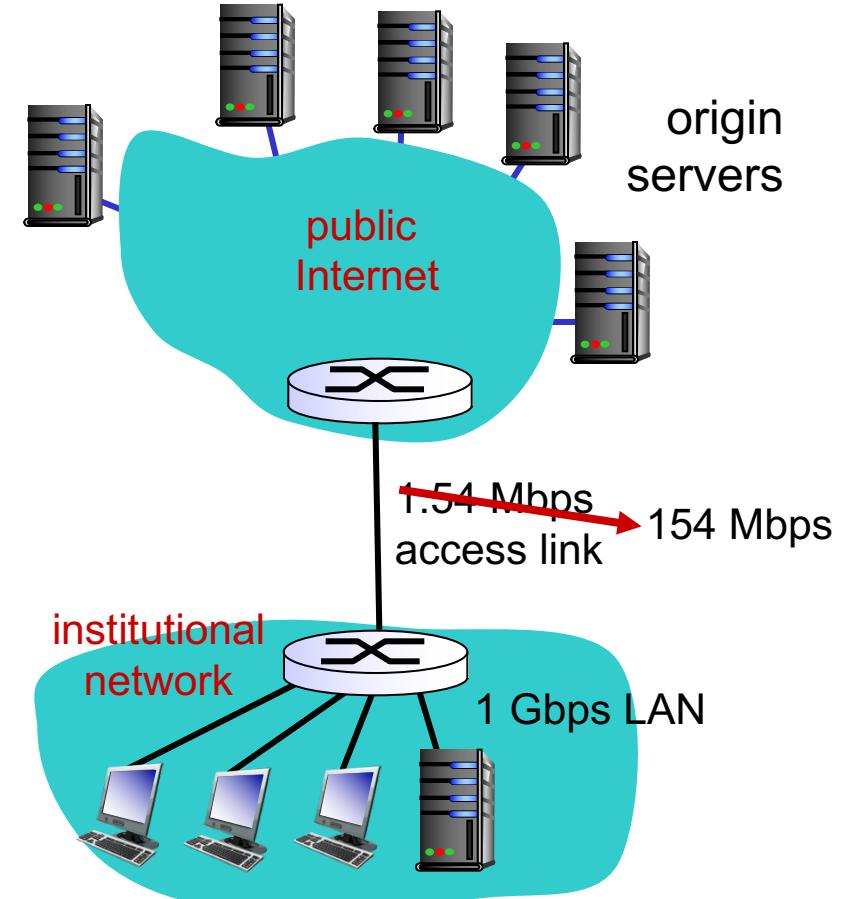
Caching example: fatter access link

assumptions:

- avg object size: 100K bits
- avg request rate from browsers to origin servers: 15/sec
- avg data rate to browsers: 1.50 Mbps
- RTT from institutional router to any origin server: 2 sec
- access link rate: ~~1.54 Mbps~~ \rightarrow 154 Mbps

consequences:

- LAN utilization: 15%
- access link utilization = ~~99%~~ \rightarrow 9.9%
- total delay = Internet delay + access delay + LAN delay
 $= 2 \text{ sec} + \cancel{\text{minutes}} + \cancel{\text{usecs}} \rightarrow \text{msecs}$



Cost: increased access link speed (not cheap!)

Caching example: install local cache

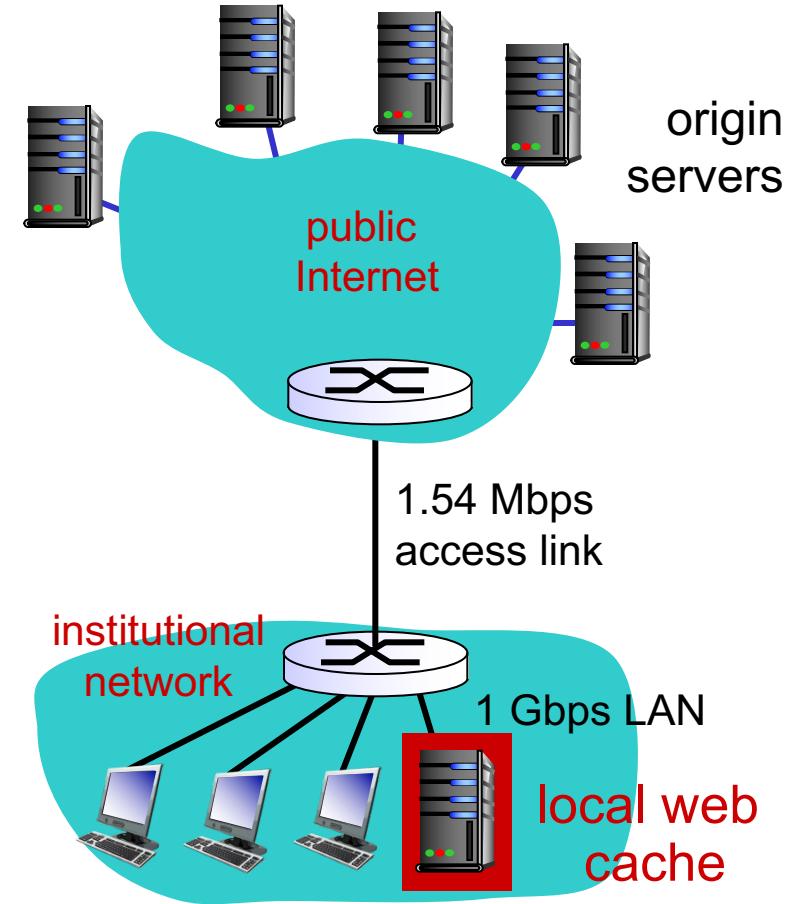
assumptions:

- avg object size: 100K bits
- avg request rate from browsers to origin servers: 15/sec
- avg data rate to browsers: 1.50 Mbps
- RTT from institutional router to any origin server: 2 sec
- access link rate: 1.54 Mbps

consequences:

- LAN utilization: 15%
- access link utilization = ?
- total delay = ?

How to compute link utilization, delay?

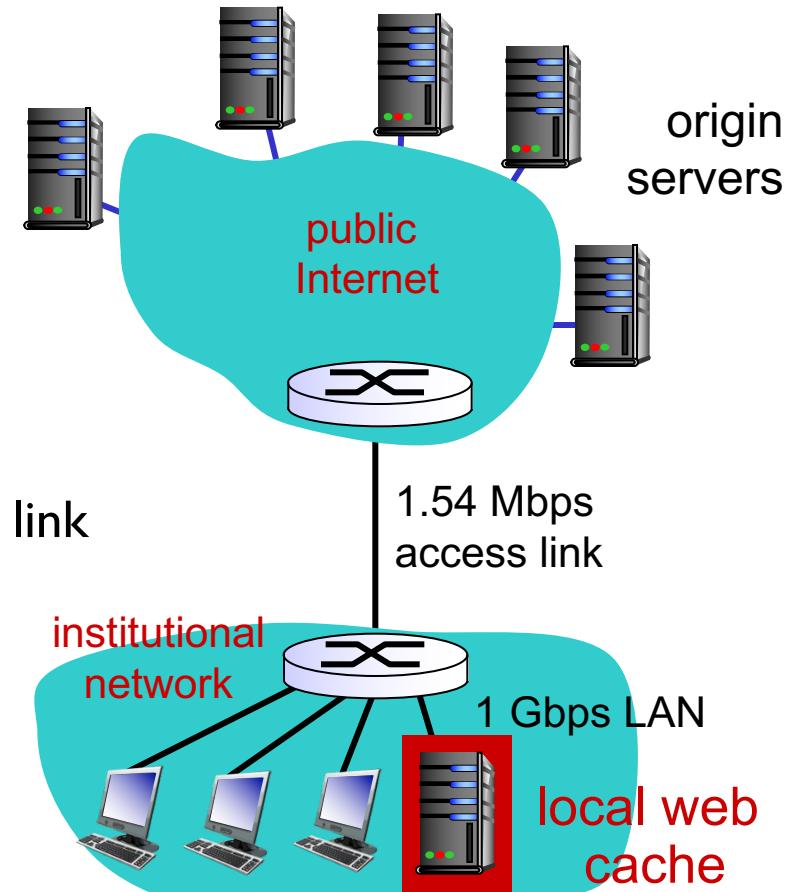


Cost: web cache (cheap!)

Caching example: install local cache

Calculating access link utilization, delay with cache:

- suppose cache hit rate is 0.4
 - 40% requests satisfied at cache, 60% requests satisfied at origin
- access link utilization:
 - 60% of requests use access link
- data rate to browsers over access link
 $= 0.6 * 1.50 \text{ Mbps} = .9 \text{ Mbps}$
 - utilization = $0.9 / 1.54 = .58$
- total delay
 - $= 0.6 * (\text{delay from origin servers}) + 0.4 * (\text{delay when satisfied at cache})$
 - $= 0.6 (2.01) + 0.4 (\sim \text{msecs}) = \sim 1.2 \text{ secs}$
 - less than with 154 Mbps link (and cheaper too!)



Conditional GET

- **Goal:** don't send object if cache has up-to-date cached version

- no object transmission delay

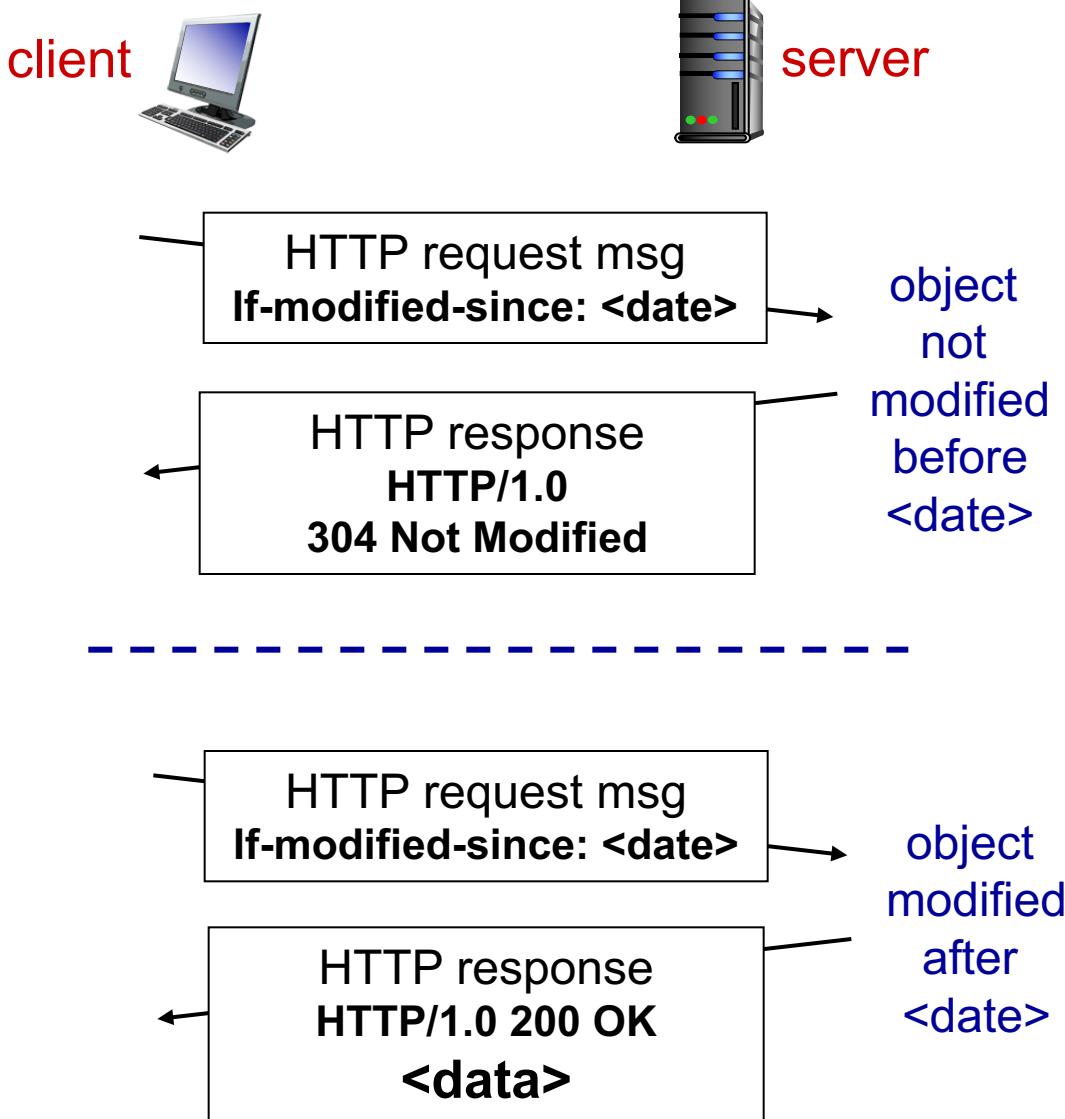
- lower link utilization

- **cache:** specify date of cached copy in HTTP request

If-modified-since:
<date>

- **server:** response contains no object if cached copy is up-to-date:

HTTP/1.0 304 Not Modified



Chapter 2: outline

2.1 principles of network
applications

2.2 Web and HTTP

2.3 electronic mail

- SMTP, POP3, IMAP

2.4 DNS

2.5 P2P applications

2.6 video streaming and
content distribution
networks

2.7 socket programming
with UDP and TCP

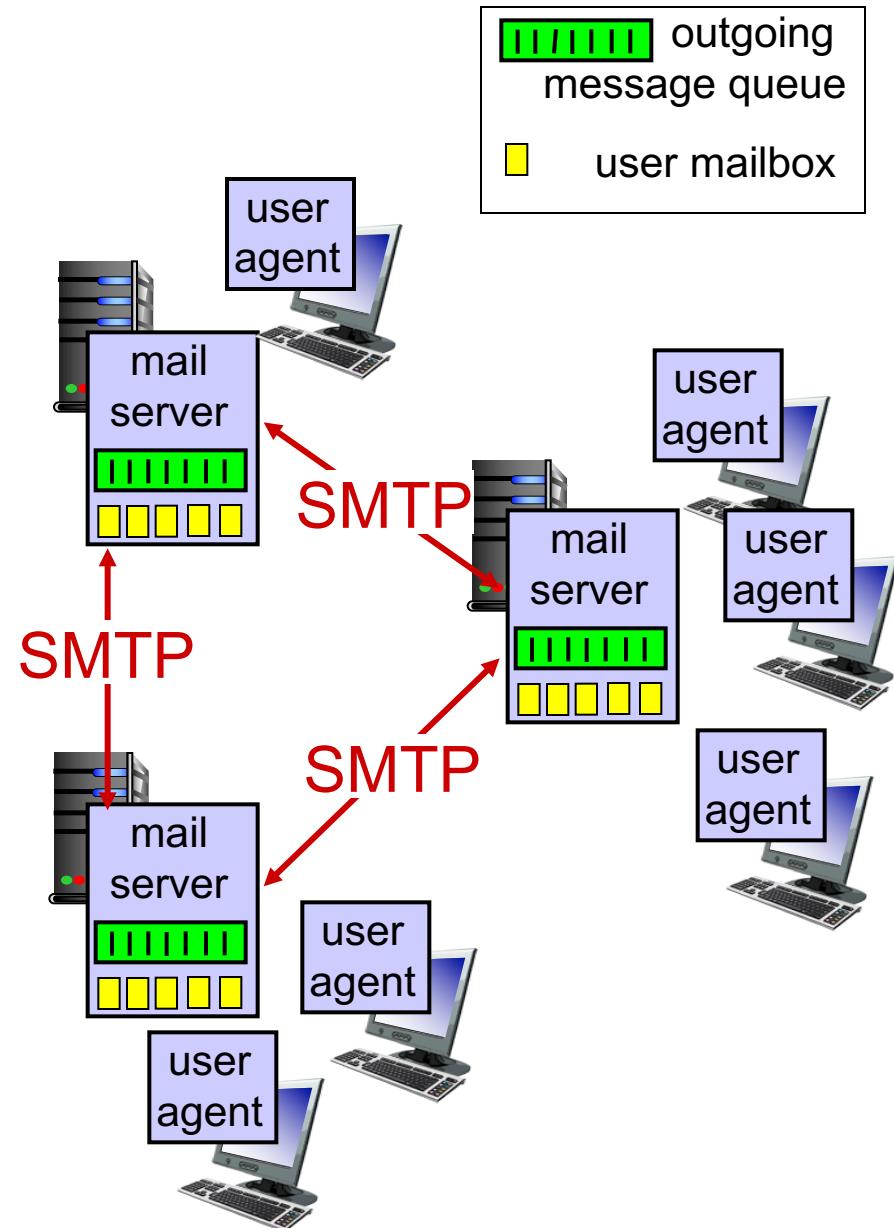
Electronic mail

Three major components:

- user agents
- mail servers
- simple mail transfer protocol: SMTP

User Agent

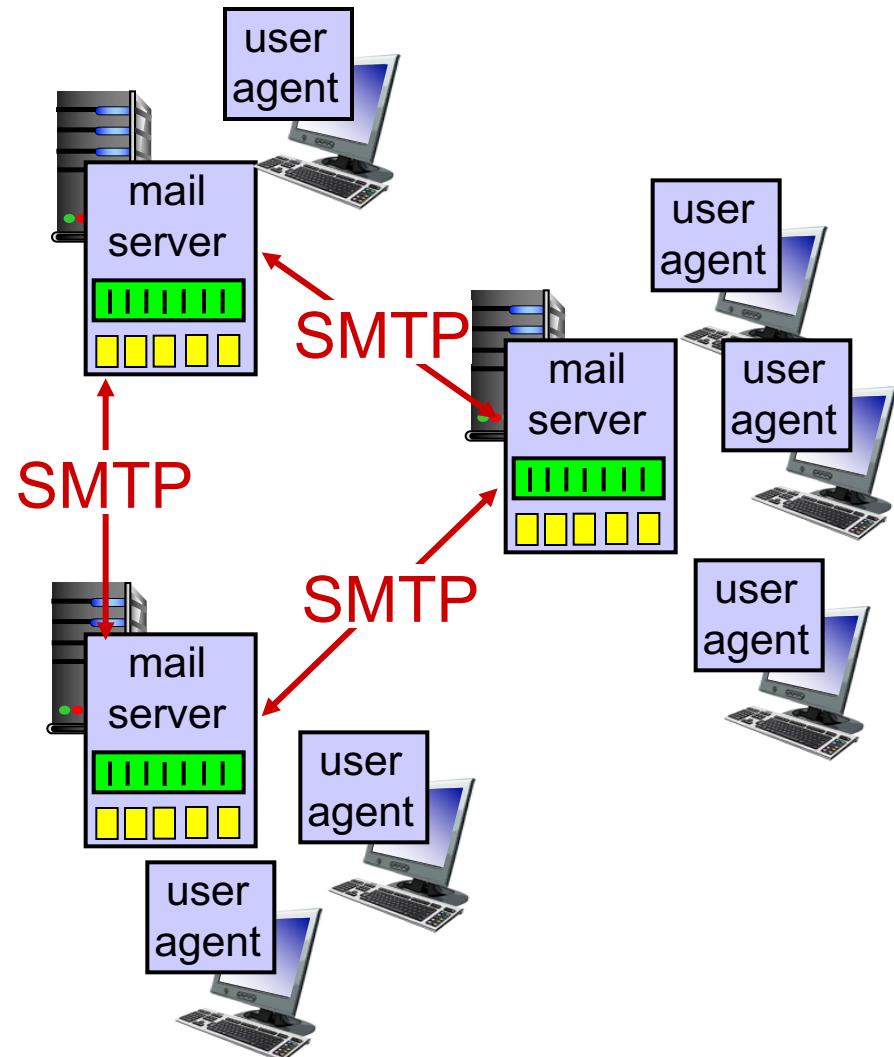
- a.k.a. “mail reader”
- composing, editing, reading mail messages
- e.g., Outlook, Thunderbird, iPhone mail client
- outgoing, incoming messages stored on server



Electronic mail: mail servers

mail servers:

- *mailbox* contains incoming messages for user
- *message queue* of outgoing (to be sent) mail messages
- *SMTP protocol* between mail servers to send email messages
 - client: sending mail server
 - “server”: receiving mail server

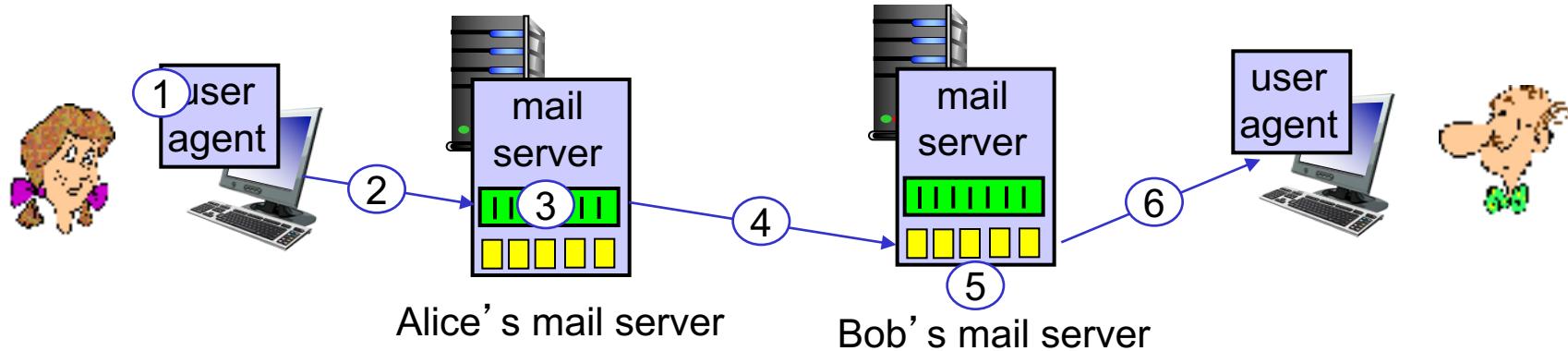


Electronic Mail: SMTP [RFC 2821]

- uses TCP to reliably transfer email message from client to server, port 25
- direct transfer: sending server to receiving server
- three phases of transfer
 - handshaking (greeting)
 - transfer of messages
 - closure
- command/response interaction (like HTTP)
 - commands: ASCII text
 - response: status code and phrase
- messages must be in 7-bit ASCII

Scenario: Alice sends message to Bob

- 1) Alice uses UA to compose message “to” bob@someschool.edu
- 2) Alice’s UA sends message to her mail server; message placed in message queue
- 3) client side of SMTP opens TCP connection with Bob’s mail server
- 4) SMTP client sends Alice’s message over the TCP connection
- 5) Bob’s mail server places the message in Bob’s mailbox
- 6) Bob invokes his user agent to read message



Sample SMTP interaction

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250 Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

Try SMTP interaction for yourself:

- **telnet servername 25**
- see 220 reply from server
- enter HELO, MAIL FROM, RCPT TO, DATA, QUIT commands

above lets you send email without using email client (reader)

SMTP: final words

- SMTP uses persistent connections
- SMTP requires message (header & body) to be in 7-bit ASCII
- SMTP server uses CRLF.CRLF to determine end of message

comparison with HTTP:

- HTTP: pull
- SMTP: push
- both have ASCII command/response interaction, status codes
- HTTP: each object encapsulated in its own response message
- SMTP: multiple objects sent in multipart message

Mail message format

SMTP: protocol for
exchanging email messages

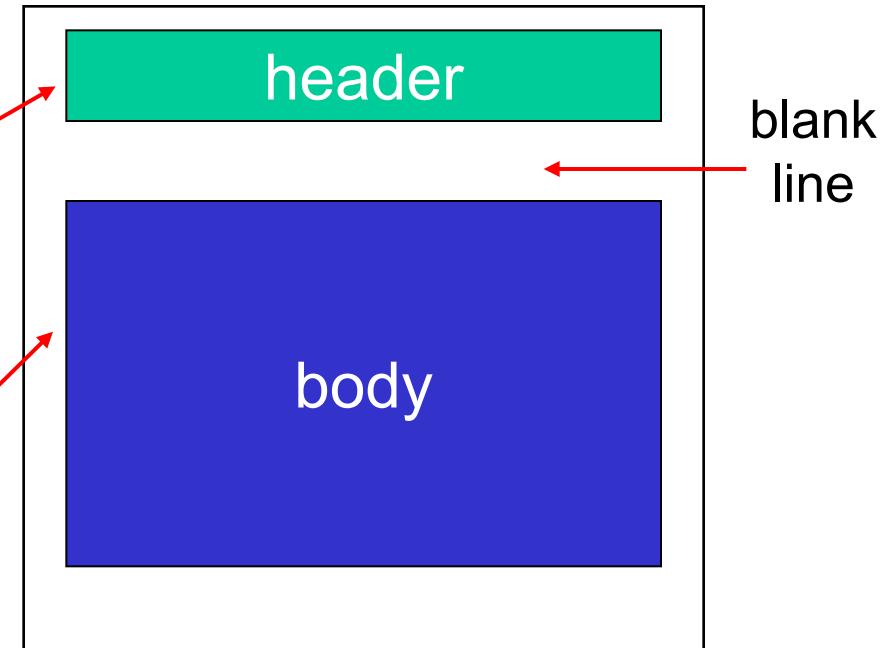
RFC 822: standard for text
message format:

- header lines, e.g.,

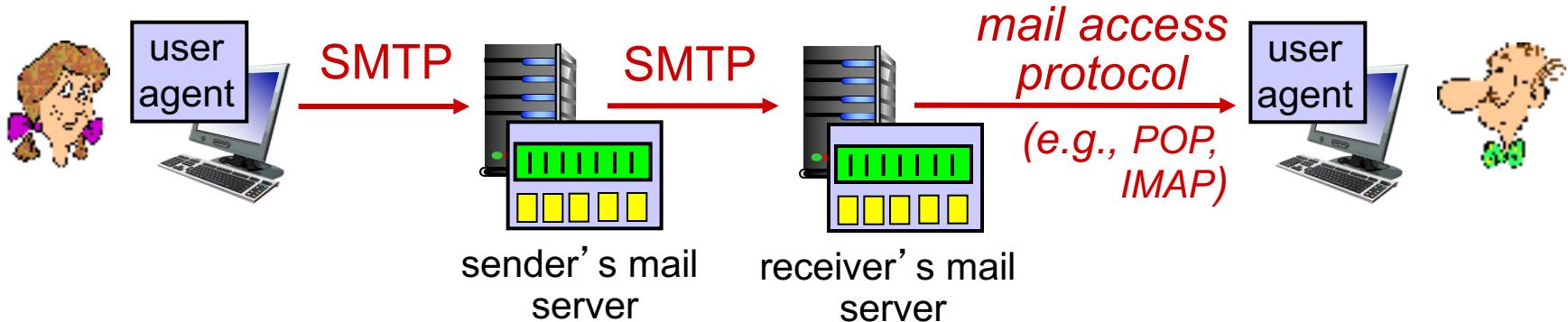
- To:
 - From:
 - Subject:

*different from SMTP MAIL
FROM, RCPT TO:
commands!*

- Body: the “message”
 - ASCII characters only



Mail access protocols



- **SMTP:** delivery/storage to receiver's server
- mail access protocol: retrieval from server
 - **POP:** Post Office Protocol [RFC 1939]: authorization, download
 - **IMAP:** Internet Mail Access Protocol [RFC 1730]: more features, including manipulation of stored messages on server
 - **HTTP:** gmail, Hotmail, Yahoo! Mail, etc.

POP3 protocol

authorization phase

- client commands:
 - **user**: declare username
 - **pass**: password
- server responses
 - +OK
 - -ERR

transaction phase, client:

- **list**: list message numbers
- **retr**: retrieve message by number
- **dele**: delete
- **quit**

```
S: +OK POP3 server ready
C: user bob
S: +OK
C: pass hungry
S: +OK user successfully logged on

C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

POP3 (more) and IMAP

more about POP3

- previous example uses POP3 “download and delete” mode
 - Bob cannot re-read e-mail if he changes client
- POP3 “download-and-keep”: copies of messages on different clients
- POP3 is stateless across sessions

IMAP

- keeps all messages in one place: at server
- allows user to organize messages in folders
- keeps user state across sessions:
 - names of folders and mappings between message IDs and folder name

Chapter 2: outline

2.1 principles of network
applications

2.2 Web and HTTP

2.3 electronic mail

- SMTP, POP3, IMAP

2.4 DNS

2.5 P2P applications

2.6 video streaming and
content distribution
networks

2.7 socket programming
with UDP and TCP

DNS: domain name system

people: many identifiers:

- SSN, name, passport #

Internet hosts, routers:

- IP address (32 bit) - used for addressing datagrams
- “name”, e.g., www.yahoo.com - used by humans

Q: how to map between IP address and name, and vice versa ?

Domain Name System:

- *distributed database* implemented in hierarchy of many *name servers*
- *application-layer protocol*: hosts, name servers communicate to *resolve* names (address/name translation)
 - note: core Internet function, implemented as application-layer protocol
 - complexity at network’s “edge”

DNS: services, structure

DNS services

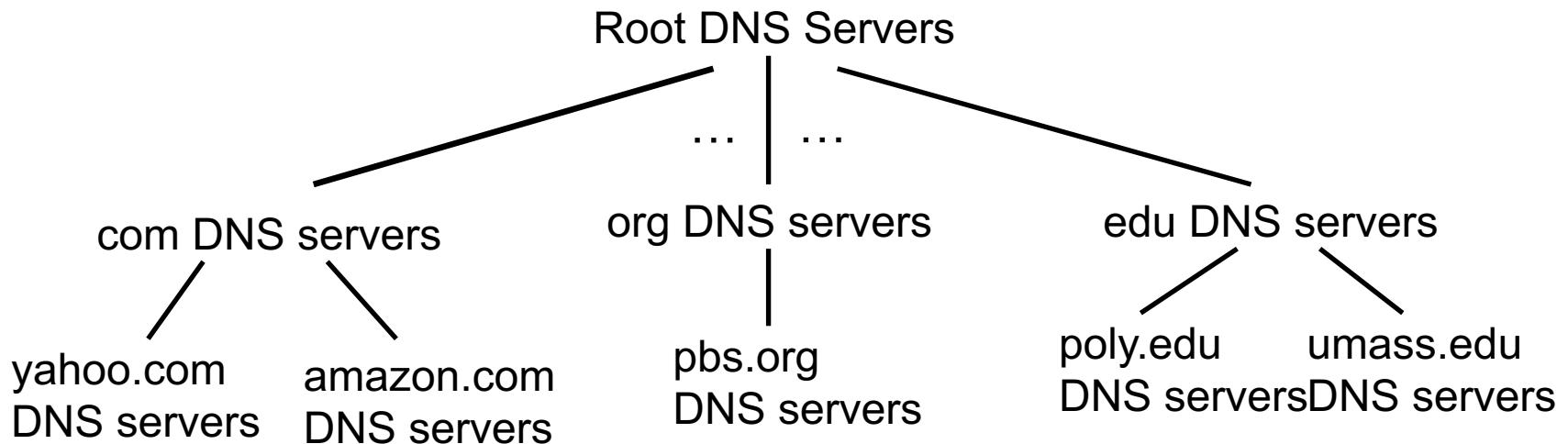
- hostname to IP address translation
- host aliasing
 - canonical, alias names
- mail server aliasing
- load distribution
 - replicated Web servers: many IP addresses correspond to one name

why not centralize DNS?

- single point of failure
- traffic volume
- distant centralized database
- maintenance

A: *doesn't scale!*

DNS: a distributed, hierarchical database

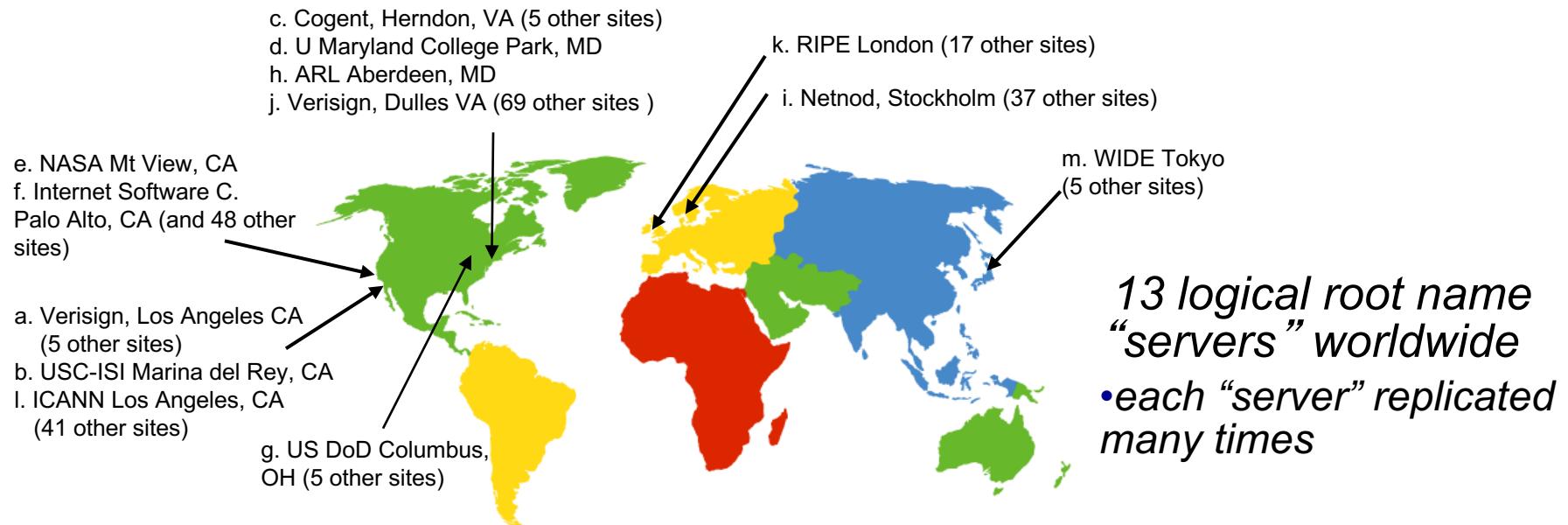


client wants IP for www.amazon.com; 1st approximation:

- client queries root server to find com DNS server
- client queries .com DNS server to get amazon.com DNS server
- client queries amazon.com DNS server to get IP address for www.amazon.com

DNS: root name servers

- contacted by local name server that can not resolve name
- root name server:
 - contacts authoritative name server if name mapping not known
 - gets mapping
 - returns mapping to local name server



TLD, authoritative servers

top-level domain (TLD) servers:

- responsible for com, org, net, edu, aero, jobs, museums, and all top-level country domains, e.g.: uk, fr, ca, jp
- Network Solutions maintains servers for .com TLD
- Educause for .edu TLD

authoritative DNS servers:

- organization's own DNS server(s), providing authoritative hostname to IP mappings for organization's named hosts
- can be maintained by organization or service provider

Local DNS name server

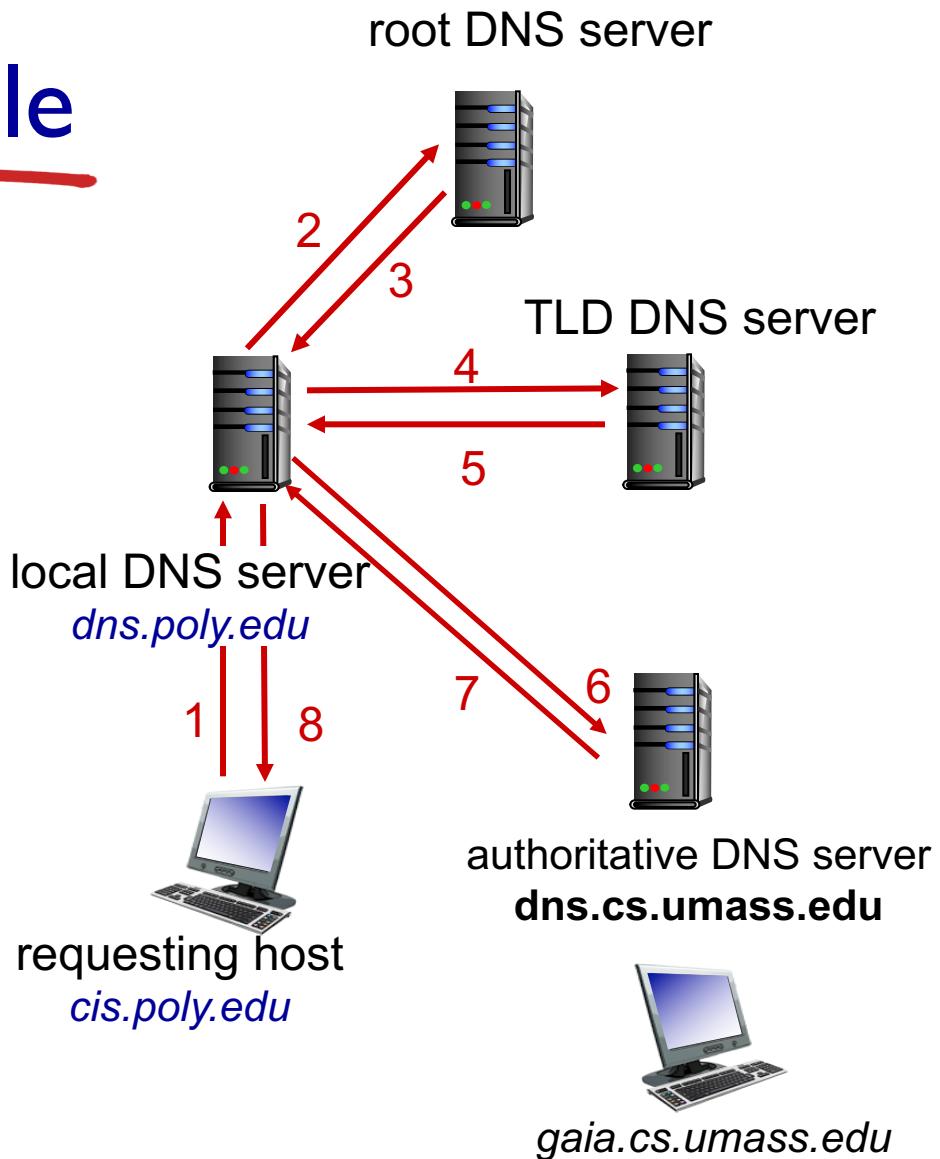
- does not strictly belong to hierarchy
- each ISP (residential ISP, company, university) has one
 - also called “default name server”
- when host makes DNS query, query is sent to its local DNS server
 - has local cache of recent name-to-address translation pairs (but may be out of date!)
 - acts as proxy, forwards query into hierarchy

DNS name resolution example

- host at `cis.poly.edu` wants IP address for `gaia.cs.umass.edu`

iterated query:

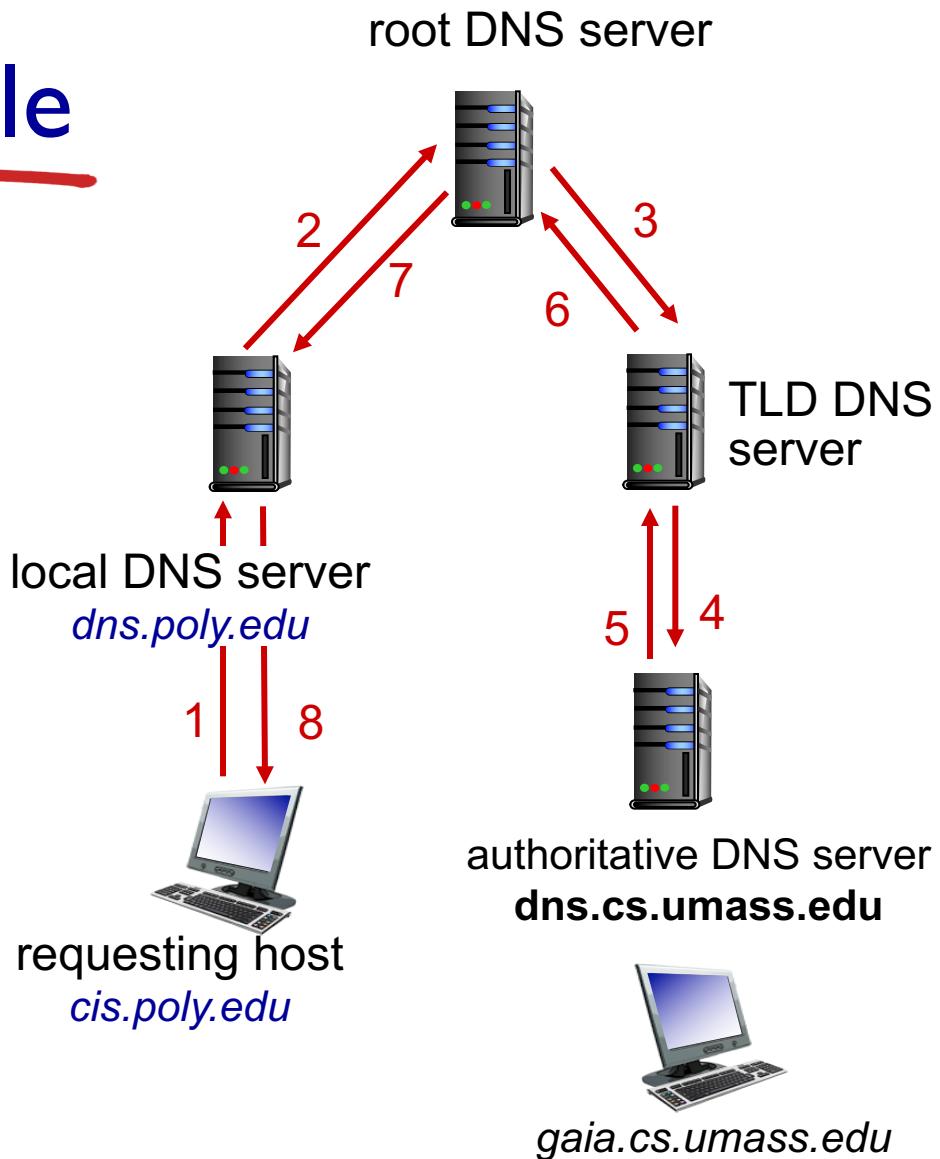
- contacted server replies with name of server to contact
- “I don’t know this name, but ask this server”



DNS name resolution example

recursive query:

- puts burden of name resolution on contacted name server
- heavy load at upper levels of hierarchy?



DNS: caching, updating records

- once (any) name server learns mapping, it *caches* mapping
 - cache entries timeout (disappear) after some time (TTL)
 - TLD servers typically cached in local name servers
 - thus root name servers not often visited
- cached entries may be *out-of-date* (best effort name-to-address translation!)
 - if name host changes IP address, may not be known Internet-wide until all TTLs expire
- update/notify mechanisms proposed IETF standard
 - RFC 2136

DNS records

DNS: distributed database storing resource records (**RR**)

RR format: `(name, value, type, ttl)`

type=A

- **name** is hostname
- **value** is IP address

type=NS

- **name** is domain (e.g.,
foo.com)
- **value** is hostname of
authoritative name
server for this domain

type=CNAME

- **name** is alias name for some
“canonical” (the real) name
- `www.ibm.com` is really
`servereast.backup2.ibm.com`
- **value** is canonical name

type=MX

- **value** is name of mailserver
associated with **name**

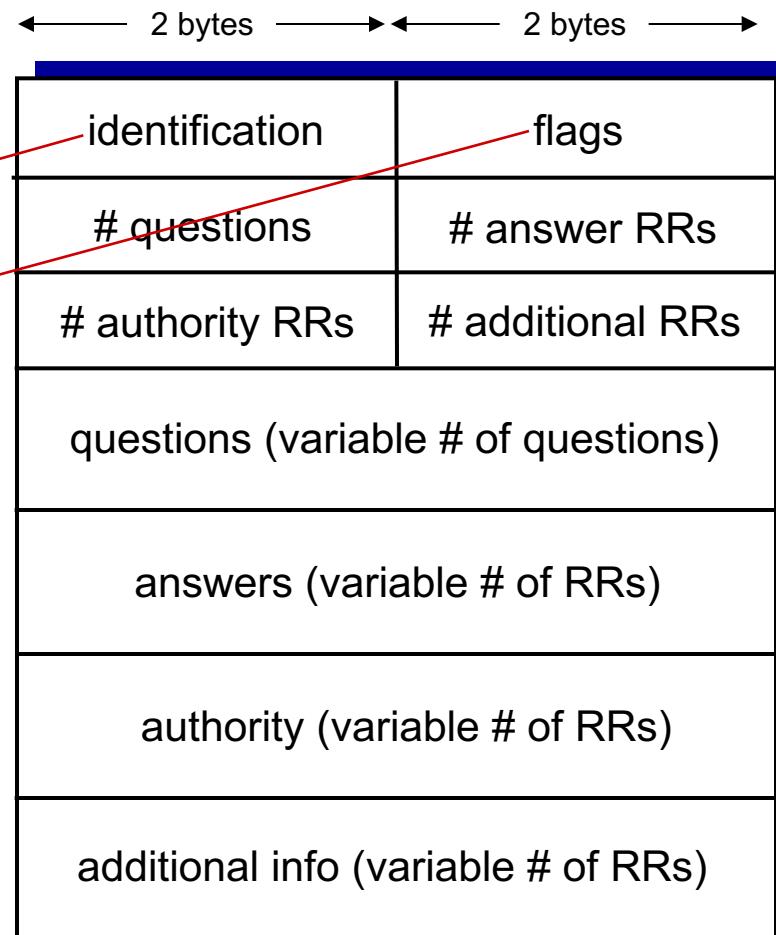
DNS protocol, messages

- *query* and *reply* messages, both with same *message format*

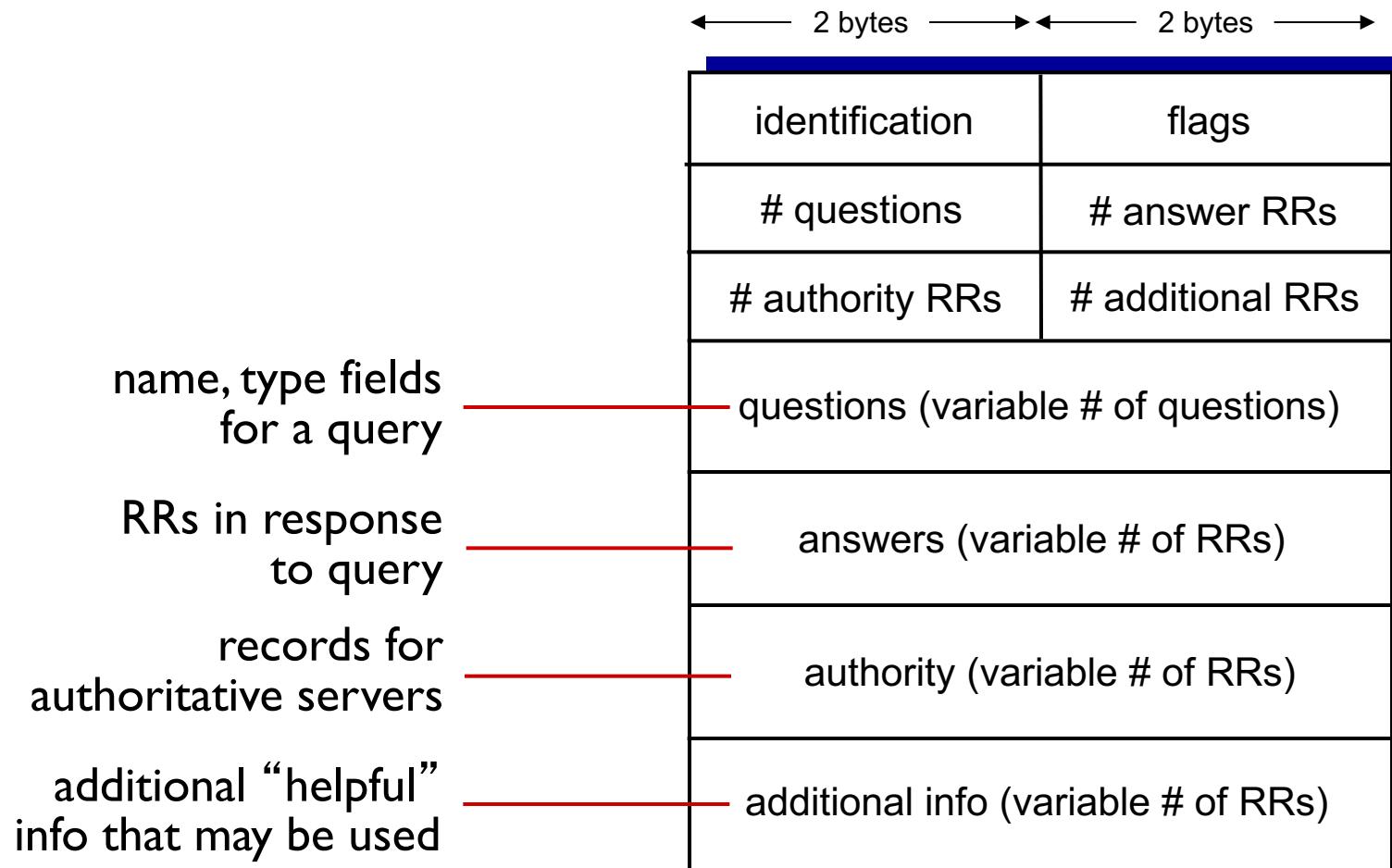
message header

- identification: 16 bit # for query, reply to query uses same #

- flags:
 - query or reply
 - recursion desired
 - recursion available
 - reply is authoritative



DNS protocol, messages



Inserting records into DNS

- example: new startup “Network Utopia”
- register name `networkutopia.com` at *DNS registrar* (e.g., Network Solutions)
 - provide names, IP addresses of authoritative name server (primary and secondary)
 - registrar inserts two RRs into .com TLD server:
`(networkutopia.com, dns1.networkutopia.com, NS)`
`(dns1.networkutopia.com, 212.212.212.1, A)`
- create authoritative server type A record for `www.networkutopia.com`; type MX record for `networkutopia.com`

Attacking DNS

DDoS attacks

- bombard root servers with traffic
 - not successful to date
 - traffic filtering
 - local DNS servers cache IPs of TLD servers, allowing root server bypass
- bombard TLD servers
 - potentially more dangerous

redirect attacks

- man-in-middle
 - Intercept queries
- DNS poisoning
 - Send bogus replies to DNS server, which caches

exploit DNS for DDoS

- send queries with spoofed source address: target IP
- requires amplification

Chapter 2: outline

2.1 principles of network
applications

2.2 Web and HTTP

2.3 electronic mail

- SMTP, POP3, IMAP

2.4 DNS

2.5 P2P applications

2.6 video streaming and
content distribution
networks

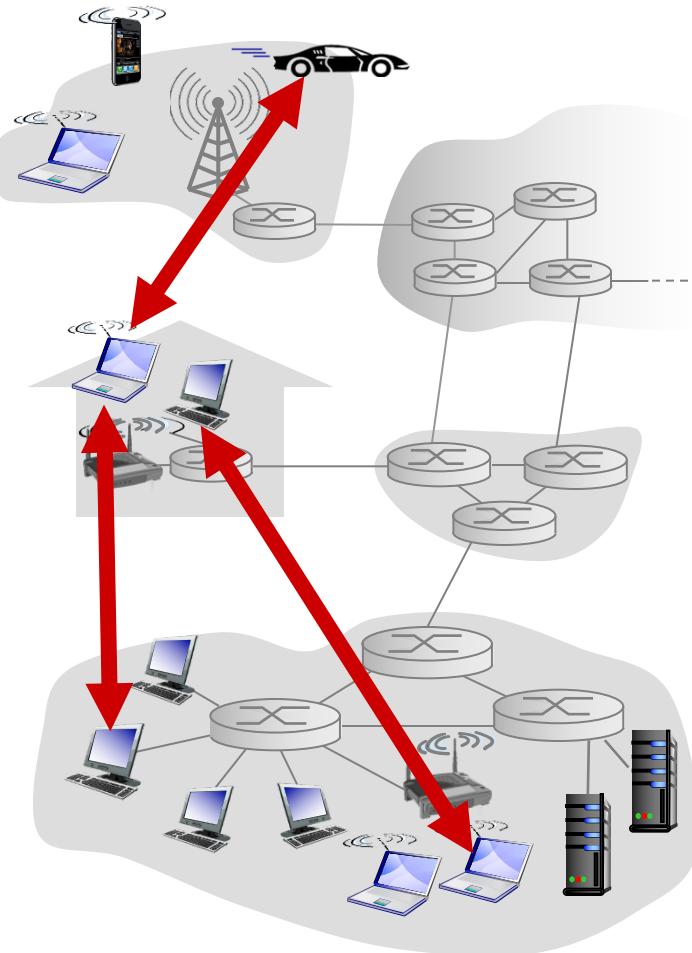
2.7 socket programming
with UDP and TCP

Pure P2P architecture

- no always-on server
- arbitrary end systems directly communicate
- peers are intermittently connected and change IP addresses

examples:

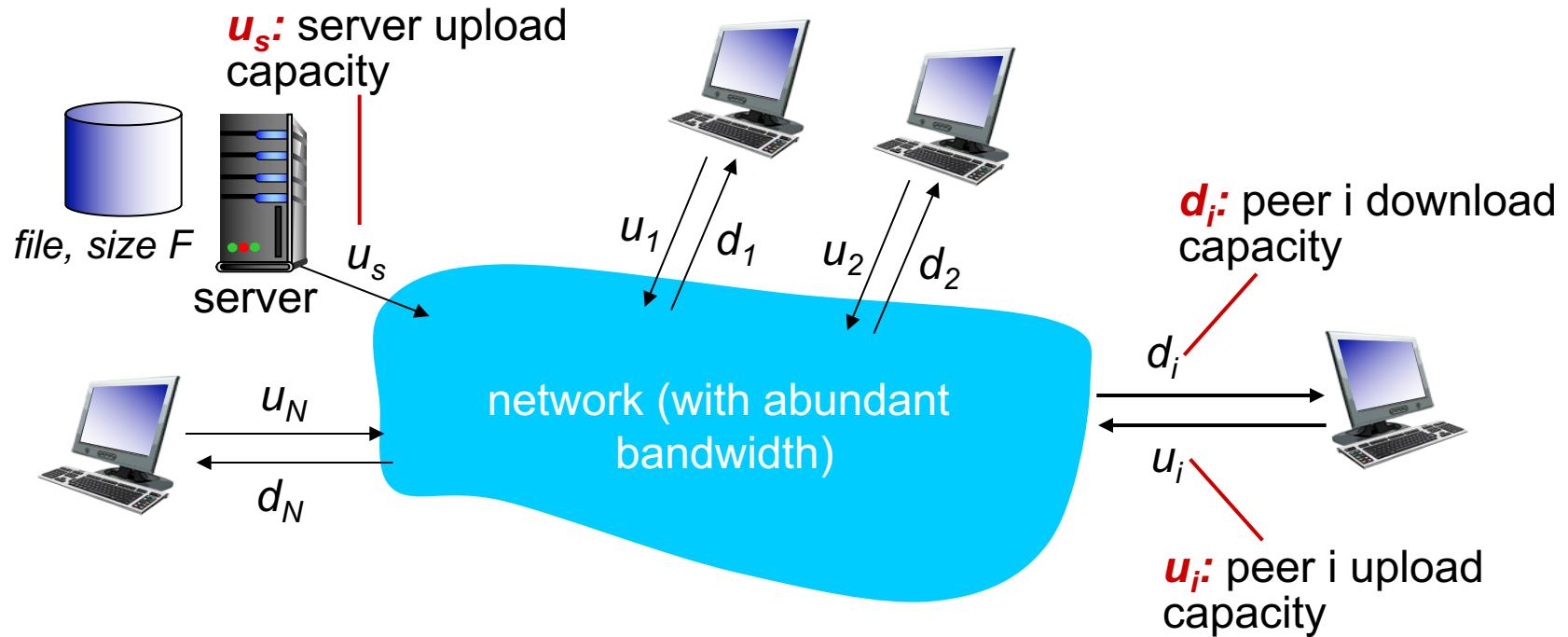
- file distribution (BitTorrent)
- Streaming (KanKan)
- VoIP (Skype)



File distribution: client-server vs P2P

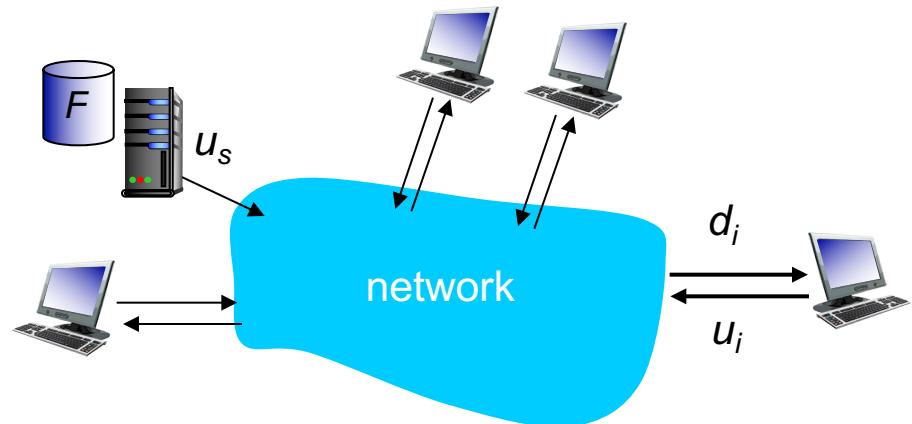
Question: how much time to distribute file (size F) from one server to N peers?

- peer upload/download capacity is limited resource



File distribution time: client-server

- **server transmission:** must sequentially send (upload) N file copies:
 - time to send one copy: F/u_s
 - time to send N copies: NF/u_s
- **client:** each client must download file copy
 - d_{min} = min client download rate
 - min client download time: F/d_{min}



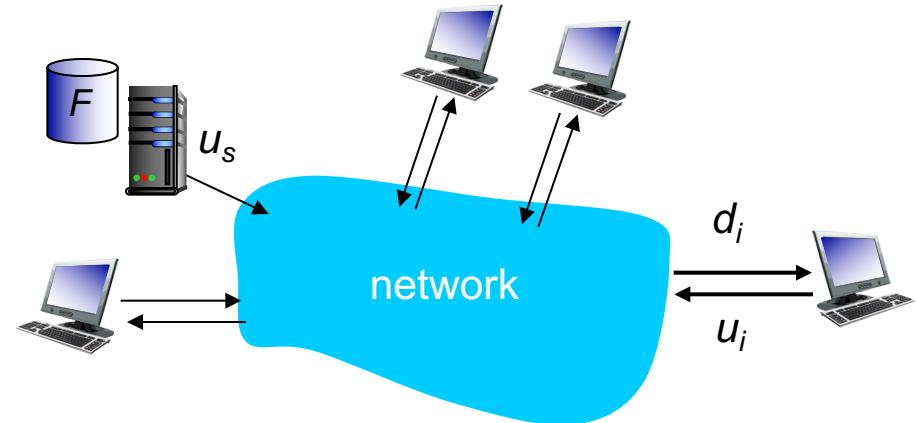
*time to distribute F
to N clients using
client-server approach*

$$D_{c-s} \geq \max\{NF/u_s, F/d_{min}\}$$

increases linearly in N

File distribution time: P2P

- **server transmission:** must upload at least one copy
 - time to send one copy: F/u_s



- **client:** each client must download file copy
 - min client download time: F/d_{\min}

- **clients:** as aggregate must download NF bits
 - max upload rate (limiting max download rate) is $u_s + \sum u_i$

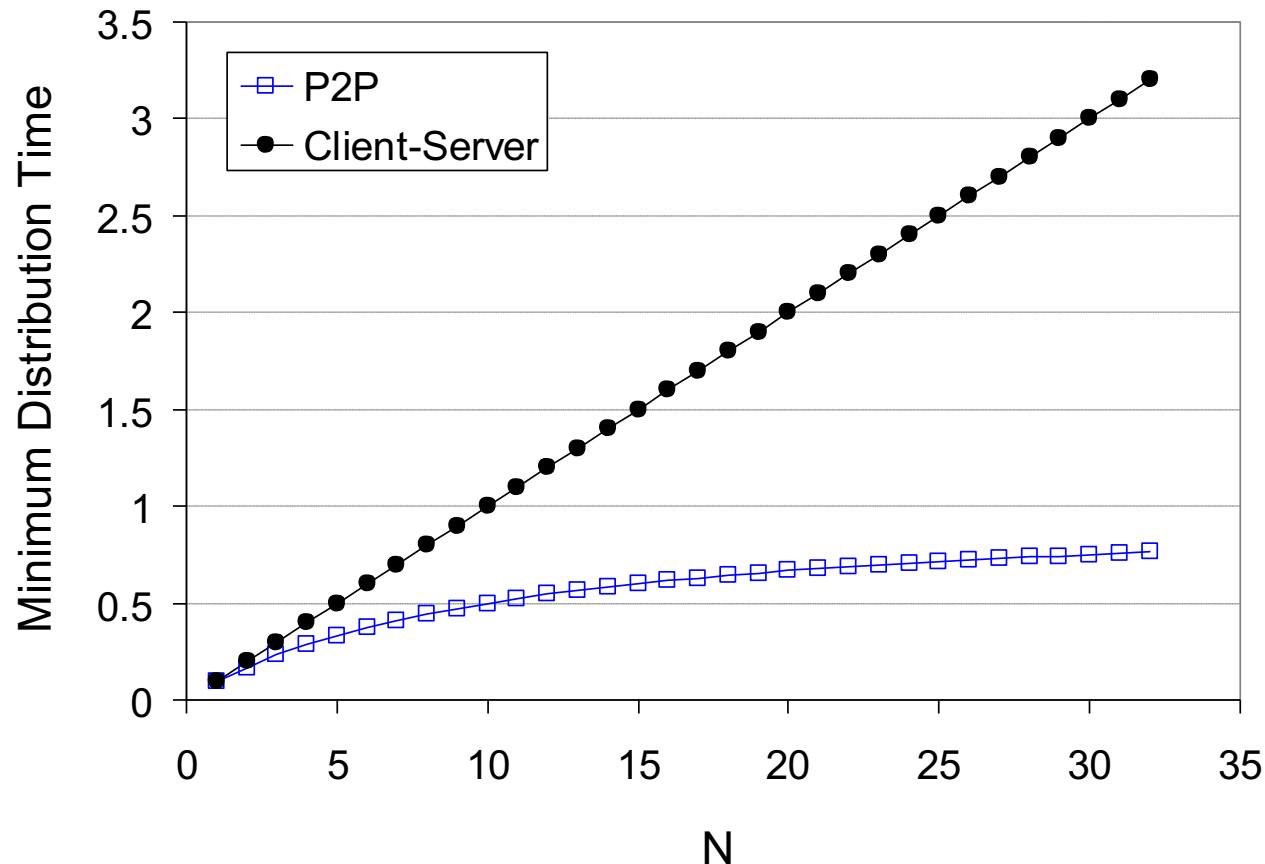
time to distribute F
to N clients using
P2P approach

$$D_{P2P} \geq \max\{F/u_s, F/d_{\min}, NF/(u_s + \sum u_i)\}$$

increases linearly in N ...
... but so does this, as each peer brings service capacity

Client-server vs. P2P: example

client upload rate = u , $F/u = 1$ hour, $u_s = 10u$, $d_{min} \geq u_s$

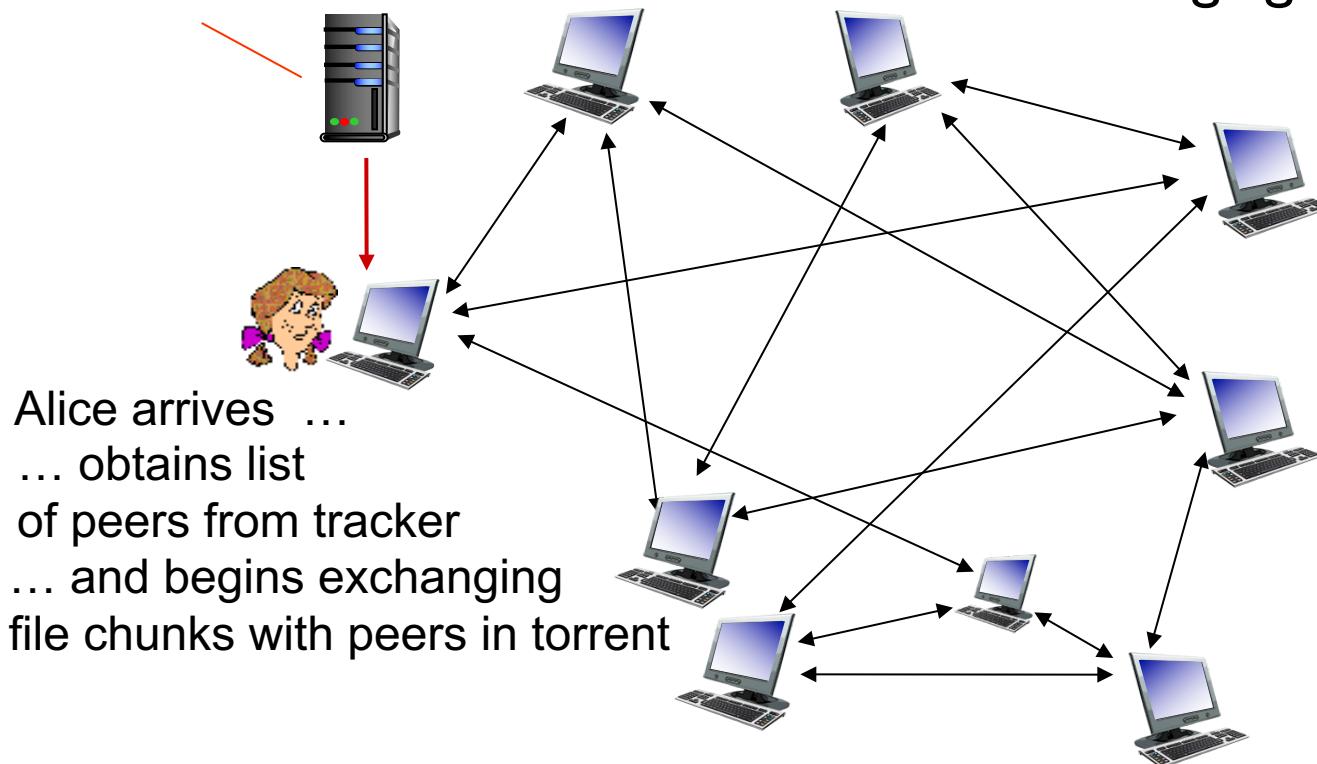


P2P file distribution: BitTorrent

- file divided into 256Kb chunks
- peers in torrent send/receive file chunks

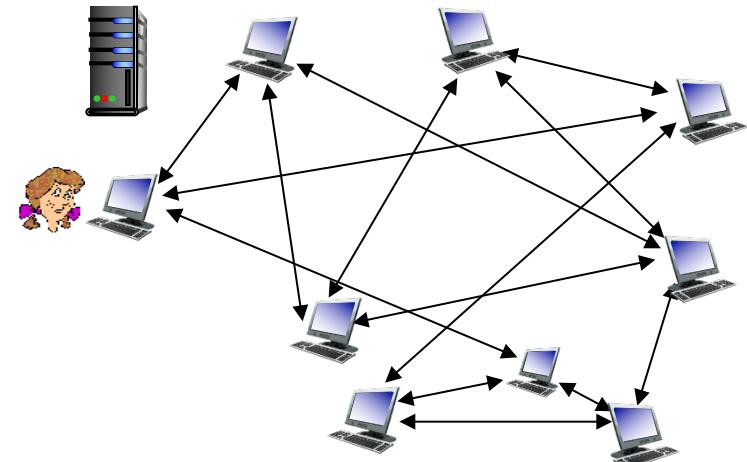
tracker: tracks peers
participating in torrent

torrent: group of peers
exchanging chunks of a file



P2P file distribution: BitTorrent

- peer joining torrent:
 - has no chunks, but will accumulate them over time from other peers
 - registers with tracker to get list of peers, connects to subset of peers (“neighbors”)
- while downloading, peer uploads chunks to other peers
- peer may change peers with whom it exchanges chunks
- *churn*: peers may come and go
- once peer has entire file, it may (selfishly) leave or (altruistically) remain in torrent



BitTorrent: requesting, sending file chunks

requesting chunks:

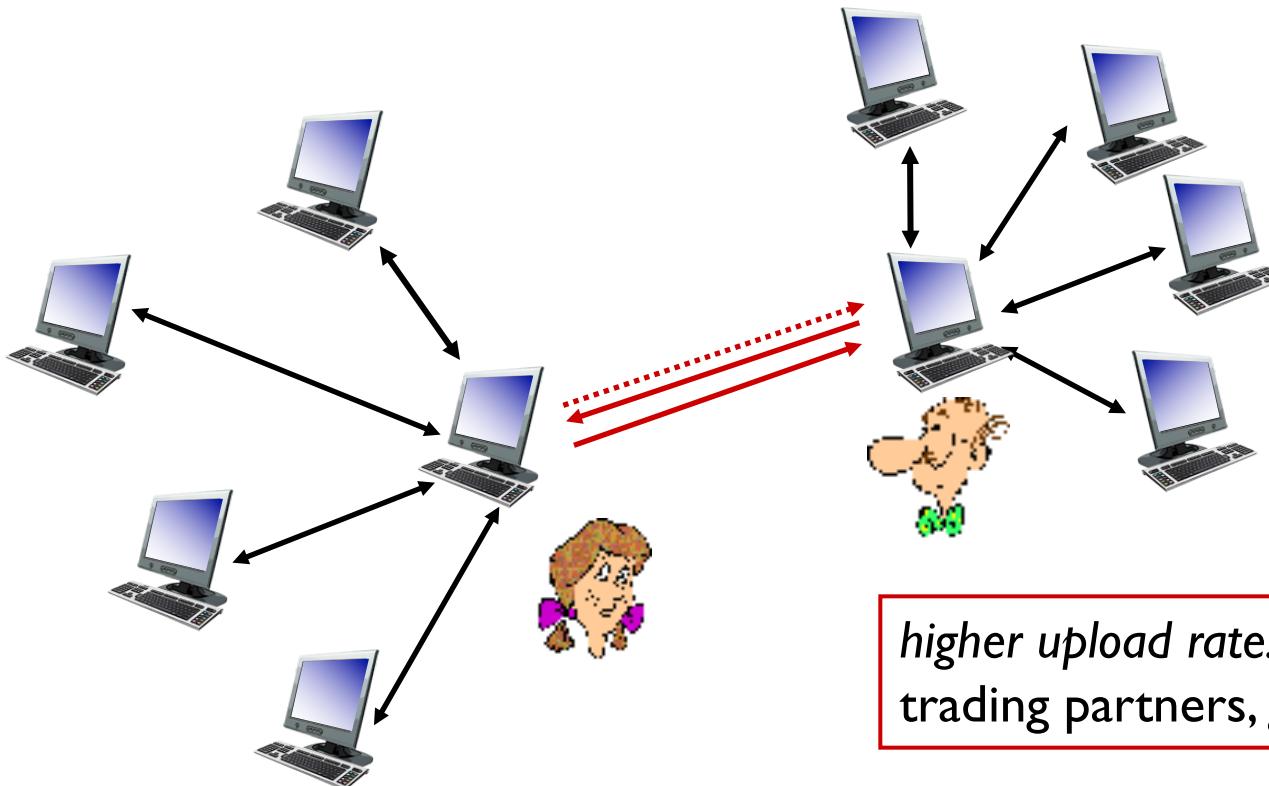
- at any given time, different peers have different subsets of file chunks
- periodically, Alice asks each peer for list of chunks that they have
- Alice requests missing chunks from peers, rarest first

sending chunks: tit-for-tat

- Alice sends chunks to those four peers currently sending her chunks *at highest rate*
 - other peers are choked by Alice (do not receive chunks from her)
 - re-evaluate top 4 every 10 secs
- every 30 secs: randomly select another peer, starts sending chunks
 - “optimistically unchoke” this peer
 - newly chosen peer may join top 4

BitTorrent: tit-for-tat

- (1) Alice “optimistically unchoke” Bob
- (2) Alice becomes one of Bob’s top-four providers; Bob reciprocates
- (3) Bob becomes one of Alice’s top-four providers



higher upload rate: find better trading partners, get file faster !

Chapter 2: outline

2.1 principles of network applications

2.2 Web and HTTP

2.3 electronic mail

- SMTP, POP3, IMAP

2.4 DNS

2.5 P2P applications

2.6 video streaming and content distribution networks (CDNs)

2.7 socket programming with UDP and TCP

Video Streaming and CDNs: context

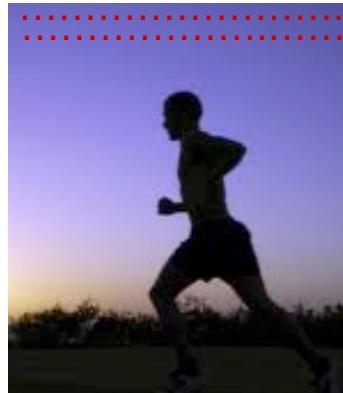
- video traffic: major consumer of Internet bandwidth
 - Netflix, YouTube: 37%, 16% of downstream residential ISP traffic
 - ~1B YouTube users, ~75M Netflix users
- challenge: scale - how to reach ~1B users?
 - single mega-video server won't work (why?)
- challenge: heterogeneity
 - different users have different capabilities (e.g., wired versus mobile; bandwidth rich versus bandwidth poor)
- *solution:* distributed, application-level infrastructure



Multimedia: video

- video: sequence of images displayed at constant rate
 - e.g., 24 images/sec
- digital image: array of pixels
 - each pixel represented by bits
- coding: use redundancy *within* and *between* images to decrease # bits used to encode image
 - spatial (within image)
 - temporal (from one image to next)

spatial coding example: instead of sending N values of same color (all purple), send only two values: color value (*purple*) and number of repeated values (N)



frame i

temporal coding example: instead of sending complete frame at $i+1$, send only differences from frame i

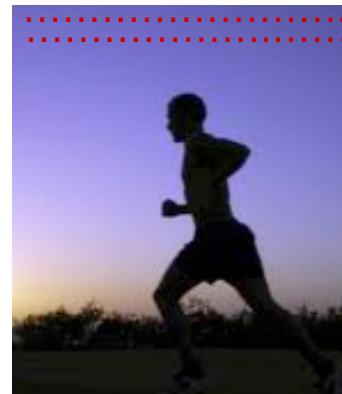


frame $i+1$

Multimedia: video

- **CBR: (constant bit rate):**
video encoding rate fixed
- **VBR: (variable bit rate):**
video encoding rate changes
as amount of spatial,
temporal coding changes
- **examples:**
 - MPEG I (CD-ROM) 1.5 Mbps
 - MPEG2 (DVD) 3-6 Mbps
 - MPEG4 (often used in Internet, < 1 Mbps)

spatial coding example: instead of sending N values of same color (all purple), send only two values: color value (*purple*) and number of repeated values (N)



frame i

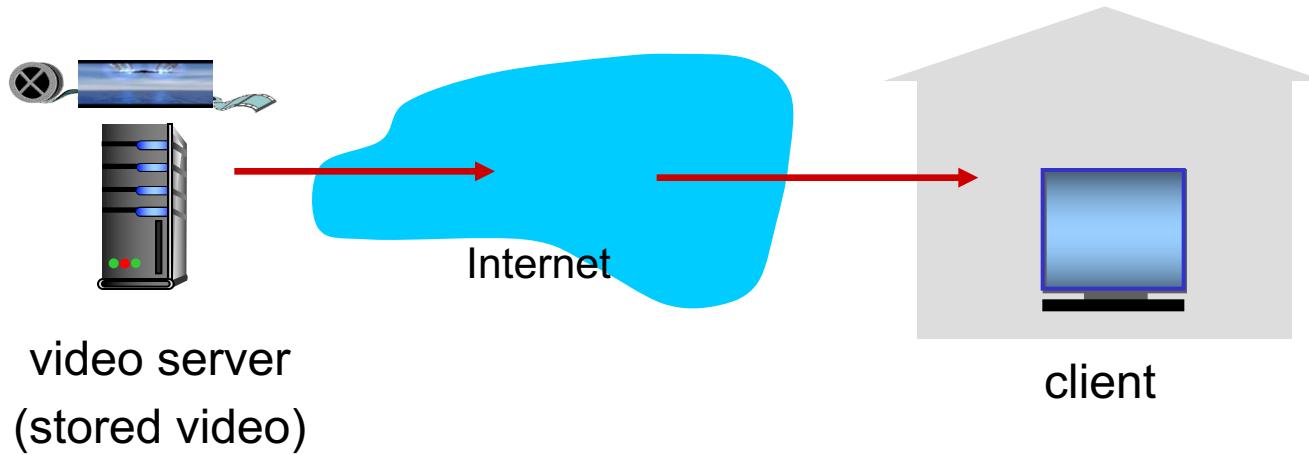
temporal coding example:
instead of sending complete frame at $i+1$,
send only differences from frame i



frame $i+1$

Streaming stored video:

simple scenario:



Streaming multimedia: DASH

- **DASH: Dynamic, Adaptive Streaming over HTTP**
- **server:**
 - divides video file into multiple chunks
 - each chunk stored, encoded at different rates
 - *manifest file*: provides URLs for different chunks
- **client:**
 - periodically measures server-to-client bandwidth
 - consulting manifest, requests one chunk at a time
 - chooses maximum coding rate sustainable given current bandwidth
 - can choose different coding rates at different points in time (depending on available bandwidth at time)

Streaming multimedia: DASH

- *DASH: Dynamic, Adaptive Streaming over HTTP*
- “*intelligence*” at client: client determines
 - *when* to request chunk (so that buffer starvation, or overflow does not occur)
 - *what encoding rate* to request (higher quality when more bandwidth available)
 - *where* to request chunk (can request from URL server that is “close” to client or has high available bandwidth)

Content distribution networks

- *challenge*: how to stream content (selected from millions of videos) to hundreds of thousands of *simultaneous* users?
- *option 1*: single, large “mega-server”
 - single point of failure
 - point of network congestion
 - long path to distant clients
 - multiple copies of video sent over outgoing link

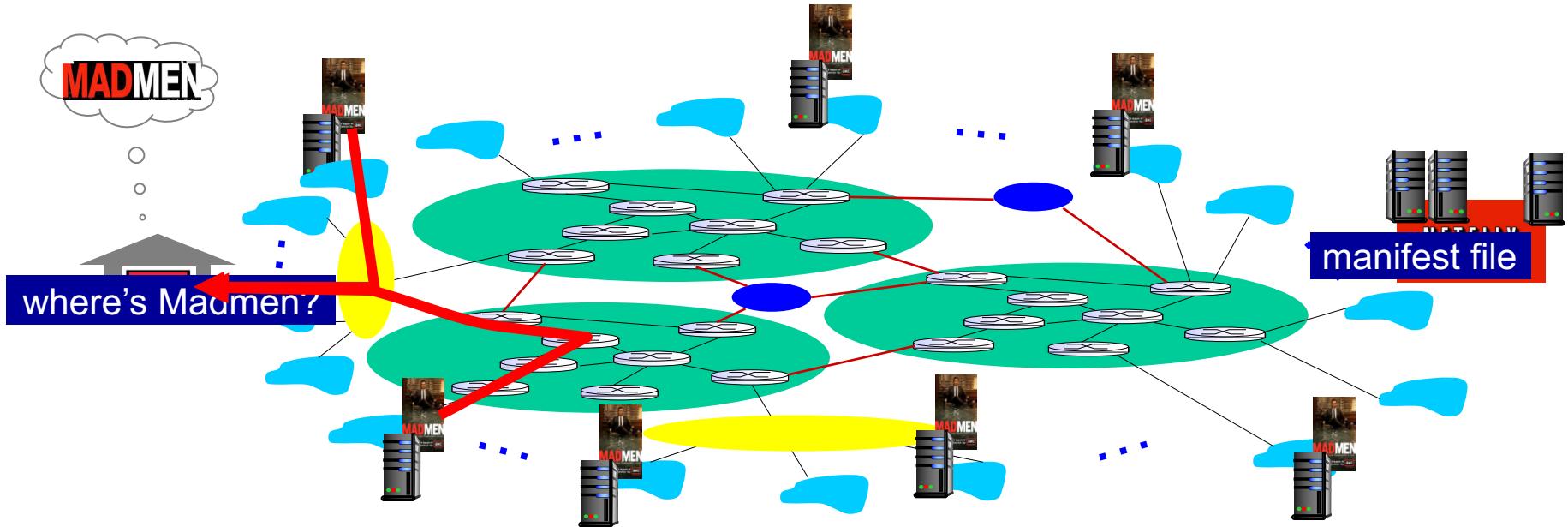
....quite simply: this solution *doesn't scale*

Content distribution networks

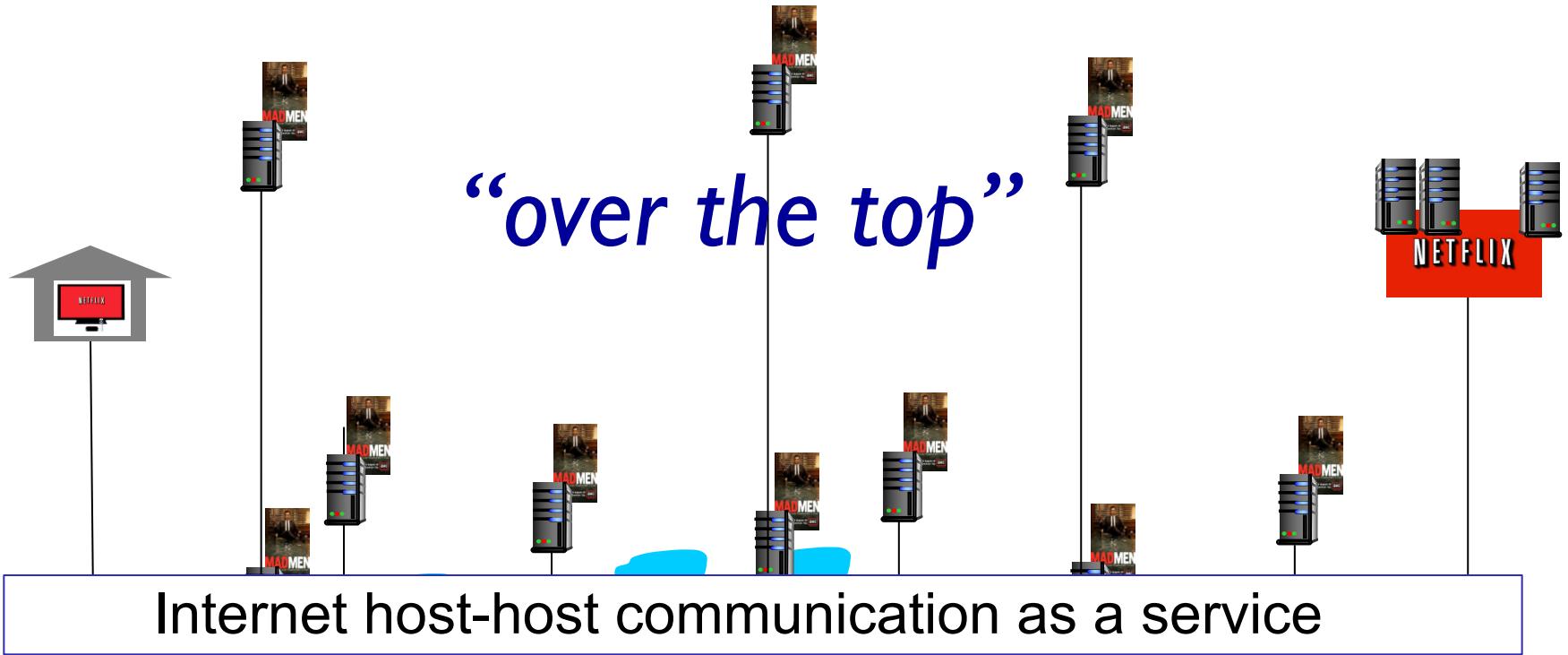
- ***challenge:*** how to stream content (selected from millions of videos) to hundreds of thousands of simultaneous users?
- ***option 2:*** store/serve multiple copies of videos at multiple geographically distributed sites (***CDN***)
 - ***enter deep:*** push CDN servers deep into many access networks
 - close to users
 - used by Akamai, 1700 locations
 - ***bring home:*** smaller number (10's) of larger clusters in POPs near (but not within) access networks
 - used by Limelight

Content Distribution Networks (CDNs)

- CDN: stores copies of content at CDN nodes
 - e.g. Netflix stores copies of MadMen
- subscriber requests content from CDN
 - directed to nearby copy, retrieves content
 - may choose different copy if network path congested



Content Distribution Networks (CDNs)



OTT challenges: coping with a congested Internet

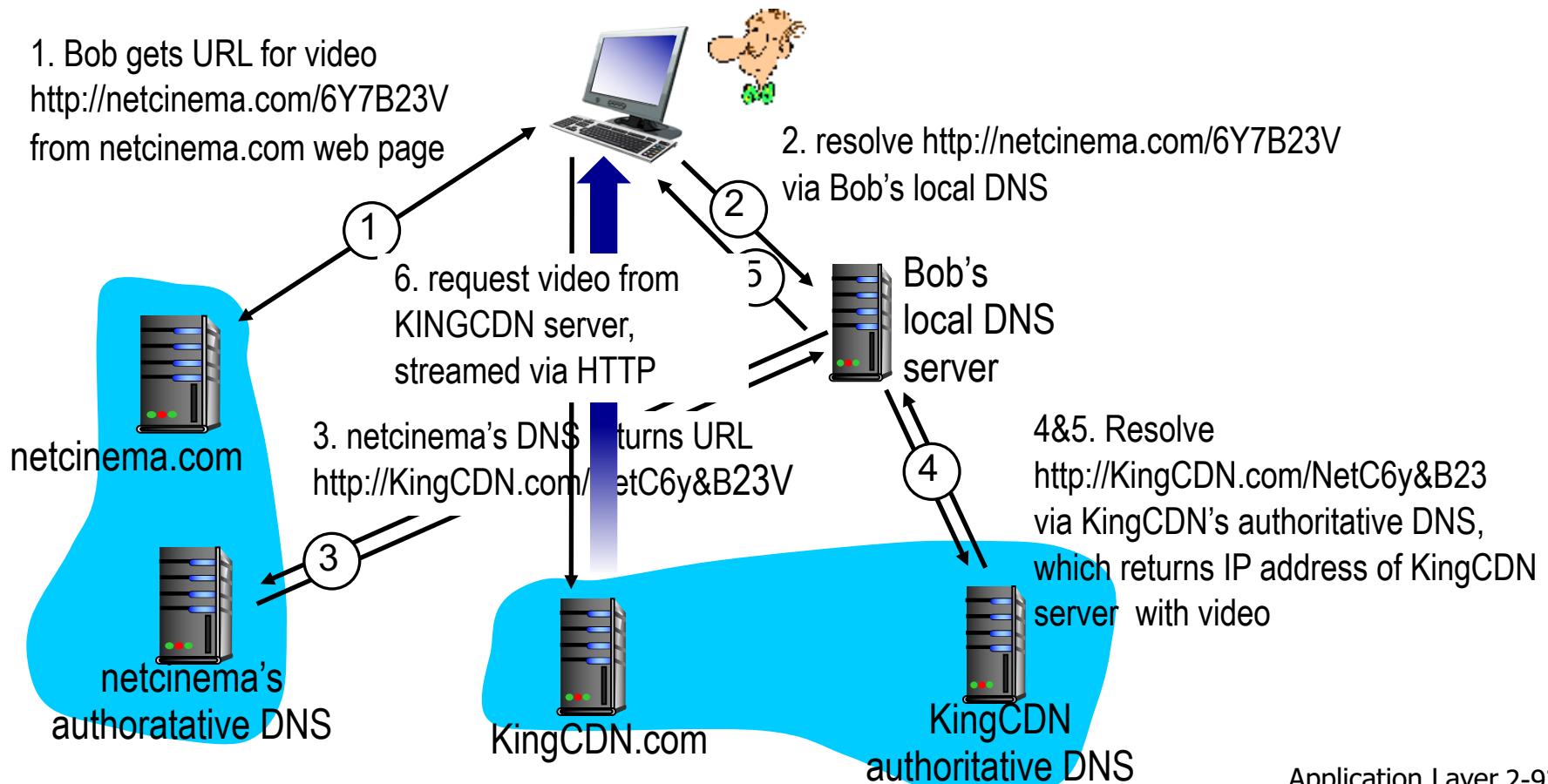
- from which CDN node to retrieve content?
- viewer behavior in presence of congestion?
- what content to place in which CDN node?

more .. in chapter 7

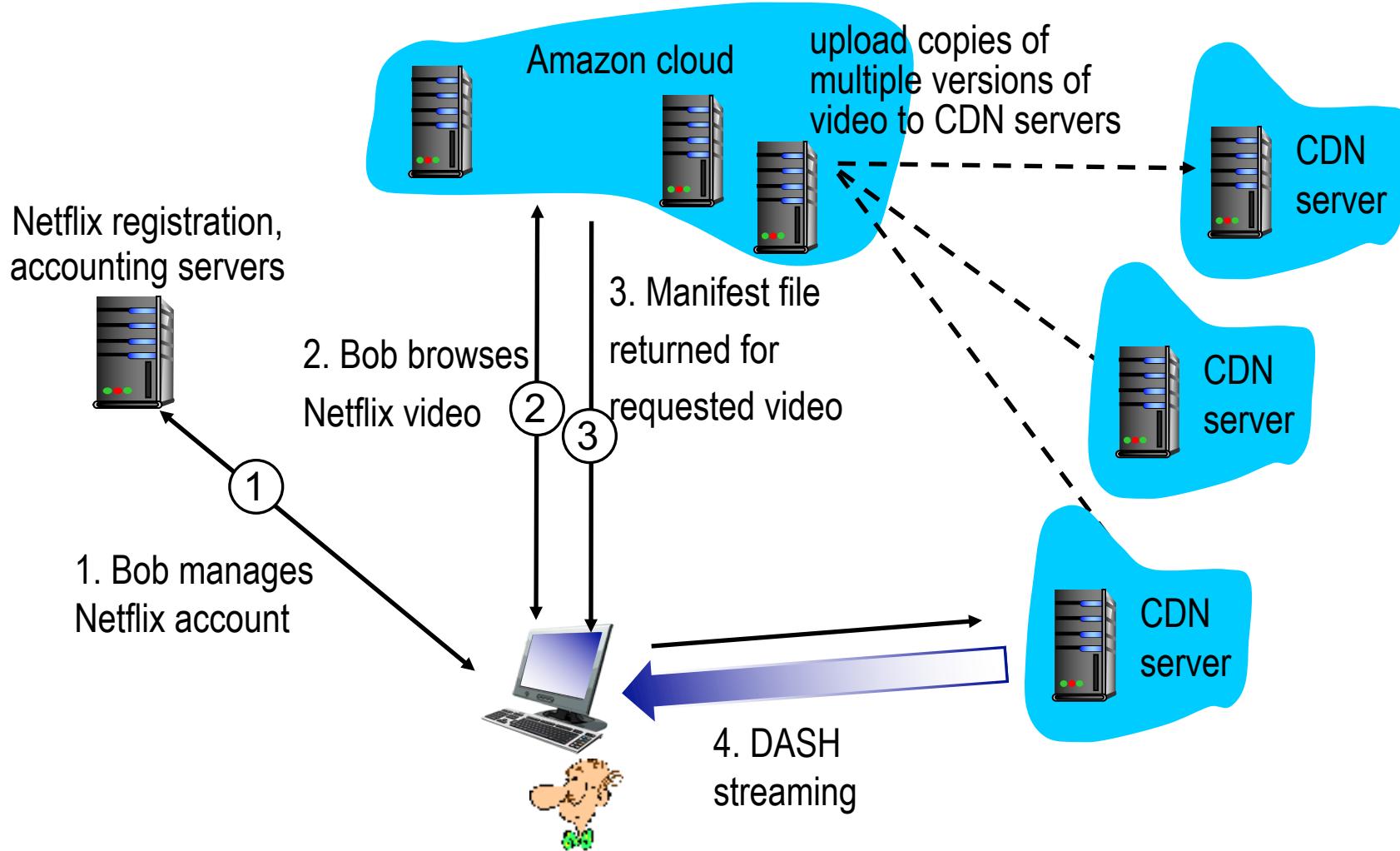
CDN content access: a closer look

Bob (client) requests video <http://netcinema.com/6Y7B23V>

- video stored in CDN at <http://KingCDN.com/NetC6y&B23V>



Case study: Netflix



Chapter 2: outline

2.1 principles of network
applications

2.2 Web and HTTP

2.3 electronic mail

- SMTP, POP3, IMAP

2.4 DNS

2.5 P2P applications

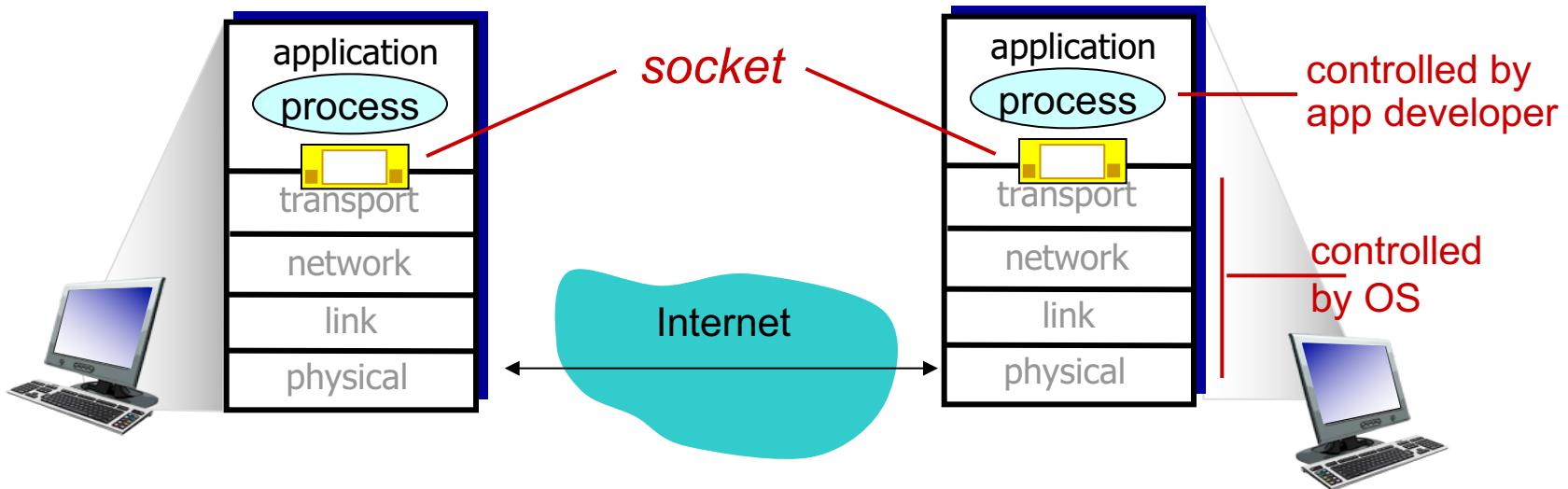
2.6 video streaming and
content distribution
networks

2.7 socket programming
with UDP and TCP

Socket programming

goal: learn how to build client/server applications that communicate using sockets

socket: door between application process and end-end-transport protocol



Socket programming

Two socket types for two transport services:

- **UDP**: unreliable datagram
- **TCP**: reliable, byte stream-oriented

Application Example:

1. client reads a line of characters (data) from its keyboard and sends data to server
2. server receives the data and converts characters to uppercase
3. server sends modified data to client
4. client receives modified data and displays line on its screen

Socket programming with UDP

UDP: no “connection” between client & server

- no handshaking before sending data
- sender explicitly attaches IP destination address and port # to each packet
- receiver extracts sender IP address and port# from received packet

UDP: transmitted data may be lost or received out-of-order

Application viewpoint:

- UDP provides *unreliable* transfer of groups of bytes (“datagrams”) between client and server

Client/server socket interaction: UDP

server (running on serverIP)

create socket, port= x:

```
serverSocket =  
socket(AF_INET,SOCK_DGRAM)
```

read datagram from
serverSocket

write reply to
serverSocket
specifying
client address,
port number

client

create socket:

```
clientSocket =  
socket(AF_INET,SOCK_DGRAM)
```

Create datagram with server IP and
port=x; send datagram via
clientSocket

read datagram from
clientSocket
close
clientSocket

Example app: UDP client

Python UDPCClient

```
include Python's socket  
library → from socket import *  
  
create UDP socket for  
server → serverName = 'hostname'  
get user keyboard  
input → serverPort = 12000  
Attach server name, port to  
message; send into socket → clientSocket = socket(AF_INET,  
                                              SOCK_DGRAM)  
read reply characters from  
socket into string → message = raw_input('Input lowercase sentence:')  
print out received string  
and close socket → clientSocket.sendto(message.encode(),  
                                         (serverName, serverPort))  
→ modifiedMessage, serverAddress =  
clientSocket.recvfrom(2048)  
→ print modifiedMessage.decode()  
clientSocket.close()
```

Example app: UDP server

Python UDPServer

```
from socket import *
serverPort = 12000
create UDP socket -----> serverSocket = socket(AF_INET, SOCK_DGRAM)
bind socket to local port  
number 12000 -----> serverSocket.bind(("", serverPort))
print ("The server is ready to receive")
loop forever -----> while True:
Read from UDP socket into  
message, getting client's  
address (client IP and port) -----> message, clientAddress = serverSocket.recvfrom(2048)
                                            modifiedMessage = message.decode().upper()
send upper case string -----> serverSocket.sendto(modifiedMessage.encode(),  
                                         clientAddress)
```

Socket programming with TCP

client must contact server

- server process must first be running
- server must have created socket (door) that welcomes client's contact

client contacts server by:

- Creating TCP socket, specifying IP address, port number of server process
- *when client creates socket:* client TCP establishes connection to server TCP

- when contacted by client, *server TCP creates new socket* for server process to communicate with that particular client
 - allows server to talk with multiple clients
 - source port numbers used to distinguish clients (more in Chap 3)

application viewpoint:

TCP provides reliable, in-order byte-stream transfer (“pipe”) between client and server

Client/server socket interaction: TCP

server (running on hostid)

client

create socket,
port=**x**, for incoming
request:
serverSocket = socket()

wait for incoming
connection request
connectionSocket = serverSocket.accept()

read request from
connectionSocket

write reply to
connectionSocket
close
connectionSocket

TCP
connection setup

create socket,
connect to **hostid**, port=**x**
clientSocket = socket()

send request using
clientSocket

read reply from
clientSocket
close
clientSocket

Example app: TCP client

Python TCPClient

```
from socket import *
serverName = 'servername'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
sentence = raw_input('Input lowercase sentence:')
clientSocket.send(sentence.encode())
modifiedSentence = clientSocket.recv(1024)
print ('From Server:', modifiedSentence.decode())
clientSocket.close()
```

create TCP socket for
server, remote port 12000

→ clientSocket = socket(AF_INET, SOCK_STREAM)

No need to attach server
name, port

→ clientSocket.send(sentence.encode())

Example app: TCP server

Python TCPServer

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind(("",serverPort))
serverSocket.listen(1)
print 'The server is ready to receive'
while True:
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()
    capitalizedSentence = sentence.upper()
    connectionSocket.send(capitalizedSentence.encode())
    connectionSocket.close()
```

create TCP welcoming
socket → serverSocket = socket(AF_INET,SOCK_STREAM)

server begins listening for
incoming TCP requests → serverSocket.bind(("",serverPort))
serverSocket.listen(1)

loop forever → print 'The server is ready to receive'

server waits on accept()
for incoming requests, new
socket created on return → while True:

read bytes from socket (but
not address as in UDP) → connectionSocket, addr = serverSocket.accept()

close connection to this
client (but *not* welcoming
socket) → sentence = connectionSocket.recv(1024).decode()
capitalizedSentence = sentence.upper()
connectionSocket.send(capitalizedSentence.
encode())
connectionSocket.close()

Chapter 2: summary

our study of network apps now complete!

- application architectures
 - client-server
 - P2P
- application service requirements:
 - reliability, bandwidth, delay
- Internet transport service model
 - connection-oriented, reliable: TCP
 - unreliable, datagrams: UDP
- specific protocols:
 - HTTP
 - SMTP, POP, IMAP
 - DNS
 - P2P: BitTorrent
- video streaming, CDNs
- socket programming:
TCP, UDP sockets

Chapter 2: summary

most importantly: learned about protocols!

- typical request/reply message exchange:
 - client requests info or service
 - server responds with data, status code
- message formats:
 - *headers*: fields giving info about data
 - *data*: info(payload) being communicated

important themes:

- control vs. messages
 - in-band, out-of-band
- centralized vs. decentralized
- stateless vs. stateful
- reliable vs. unreliable message transfer
- “complexity at network edge”

Chapter 3

Transport Layer

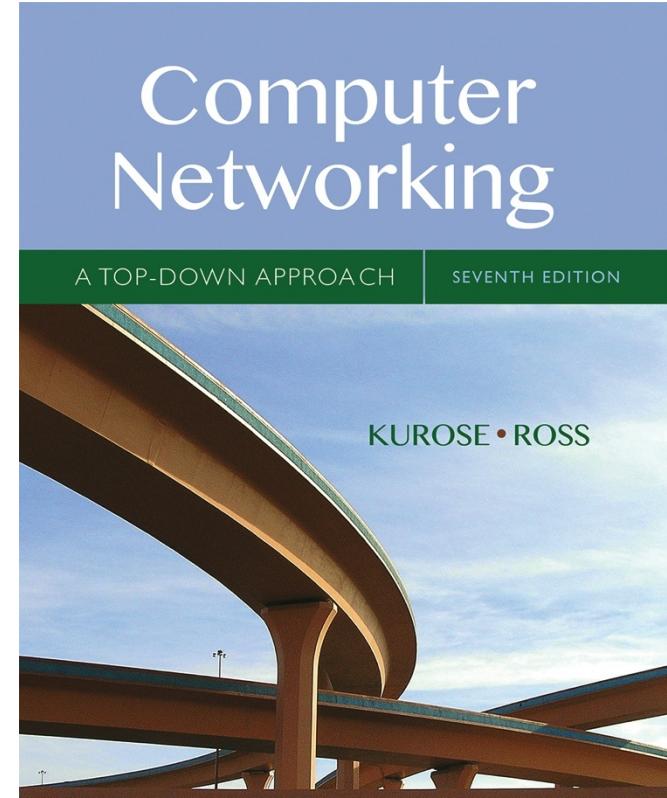
A note on the use of these Powerpoint slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR

© All material copyright 1996-2016
J.F Kurose and K.W. Ross, All Rights Reserved



*Computer
Networking: A Top
Down Approach*

7th edition

Jim Kurose, Keith Ross
Pearson/Addison Wesley
April 2016

Chapter 3: Transport Layer

our goals:

- understand principles behind transport layer services:
 - multiplexing, demultiplexing
 - reliable data transfer
 - flow control
 - congestion control
- learn about Internet transport layer protocols:
 - UDP: connectionless transport
 - TCP: connection-oriented reliable transport
 - TCP congestion control

Chapter 3 outline

3.1 transport-layer services

3.2 multiplexing and demultiplexing

3.3 connectionless transport: UDP

3.4 principles of reliable data transfer

3.5 connection-oriented transport: TCP

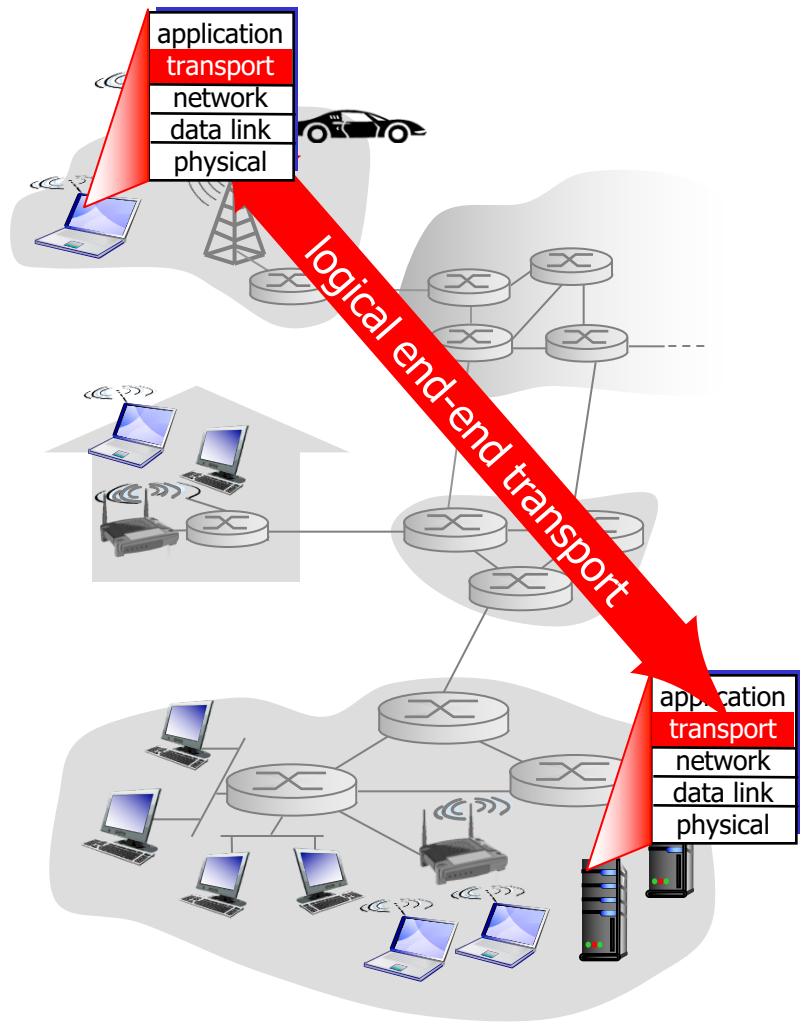
- segment structure
- reliable data transfer
- flow control
- connection management

3.6 principles of congestion control

3.7 TCP congestion control

Transport services and protocols

- provide *logical communication* between app processes running on different hosts
- transport protocols run in end systems
 - send side: breaks app messages into *segments*, passes to network layer
 - rcv side: reassembles segments into messages, passes to app layer
- more than one transport protocol available to apps
 - Internet: TCP and UDP



Transport vs. network layer

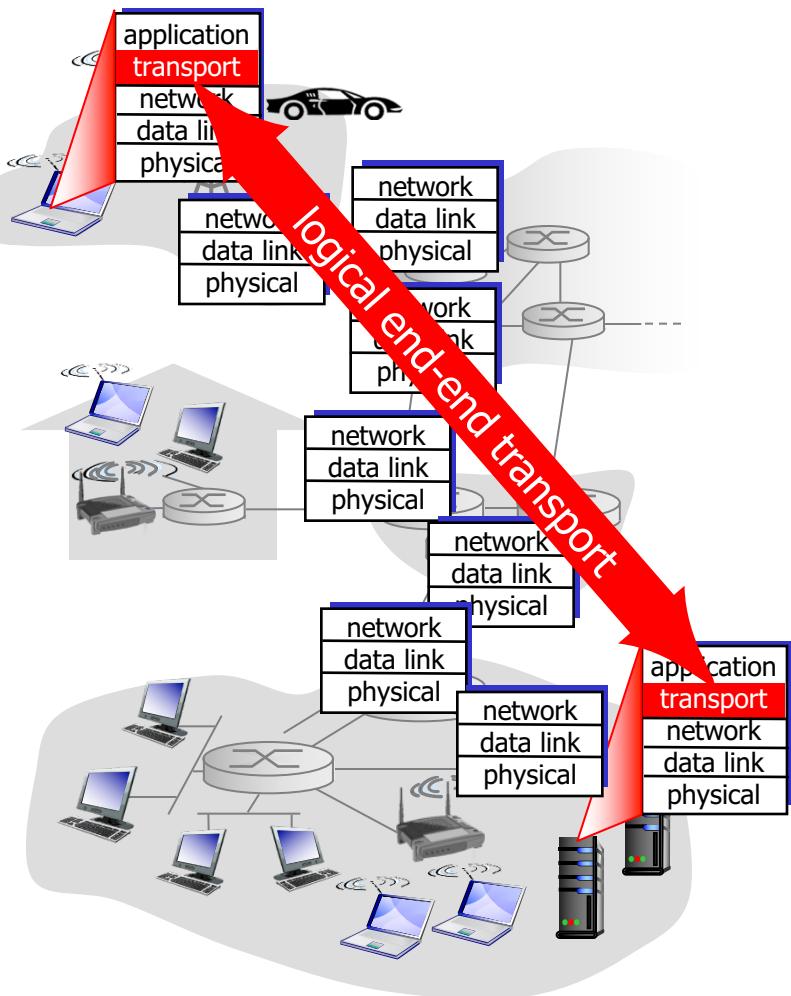
- *network layer*: logical communication between hosts
- *transport layer*: logical communication between processes
 - relies on, enhances, network layer services

household analogy:

- 12 kids in Ann's house sending letters to 12 kids in Bill's house:*
- hosts = houses
 - processes = kids
 - app messages = letters in envelopes
 - transport protocol = Ann and Bill who demux to in-house siblings
 - network-layer protocol = postal service

Internet transport-layer protocols

- reliable, in-order delivery (TCP)
 - congestion control
 - flow control
 - connection setup
- unreliable, unordered delivery: UDP
 - no-frills extension of “best-effort” IP
- services not available:
 - delay guarantees
 - bandwidth guarantees



Chapter 3 outline

3.1 transport-layer services

3.2 multiplexing and demultiplexing

3.3 connectionless transport: UDP

3.4 principles of reliable data transfer

3.5 connection-oriented transport: TCP

- segment structure
- reliable data transfer
- flow control
- connection management

3.6 principles of congestion control

3.7 TCP congestion control

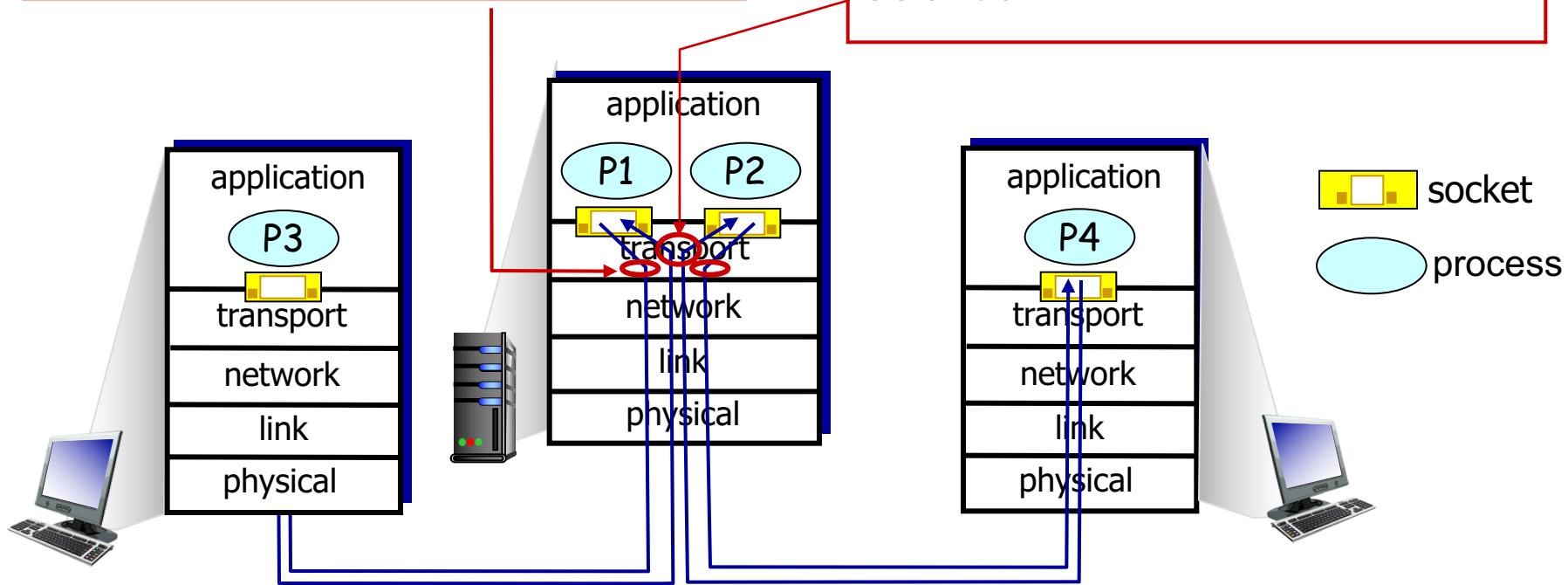
Multiplexing/demultiplexing

multiplexing at sender:

handle data from multiple sockets, add transport header (later used for demultiplexing)

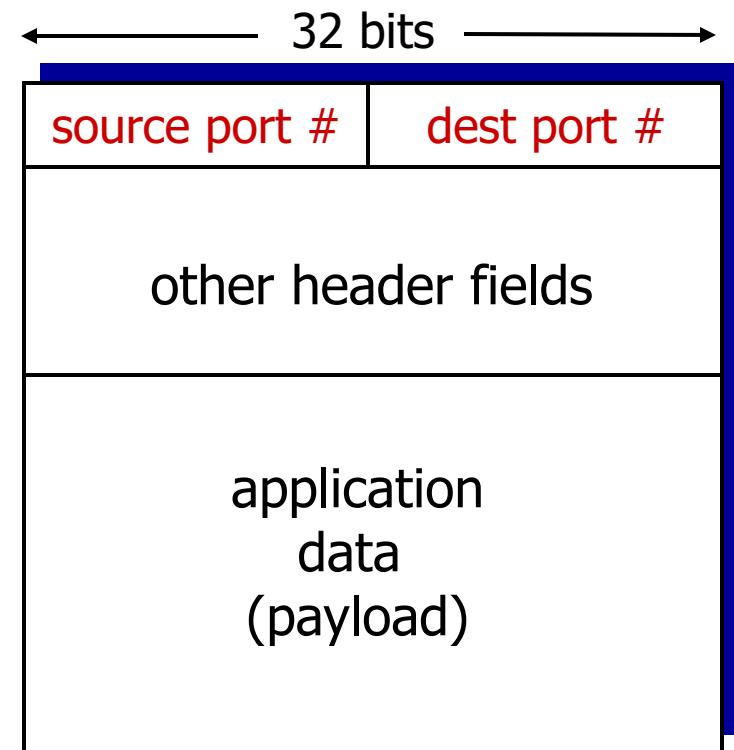
demultiplexing at receiver:

use header info to deliver received segments to correct socket



How demultiplexing works

- host receives IP datagrams
 - each datagram has source IP address, destination IP address
 - each datagram carries one transport-layer segment
 - each segment has source, destination port number
- host uses *IP addresses & port numbers* to direct segment to appropriate socket



TCP/UDP segment format

Connectionless demultiplexing

- *recall:* created socket has host-local port #:

```
DatagramSocket mySocket1  
= new DatagramSocket(1234) ;
```

- *recall:* when creating datagram to send into UDP socket, must specify
 - destination IP address
 - destination port #

-
- when host receives UDP segment:

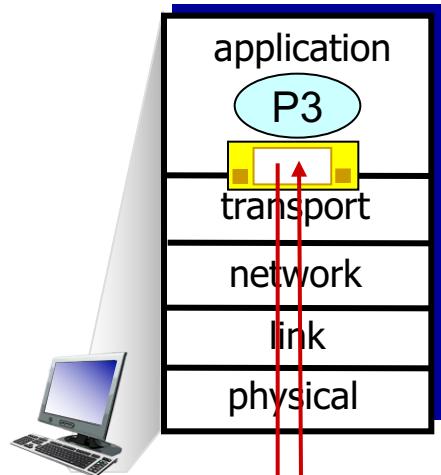
- checks destination port # in segment
- directs UDP segment to socket with that port #



IP datagrams with *same dest. port #*, but different source IP addresses and/or source port numbers will be directed to *same socket* at dest

Connectionless demux: example

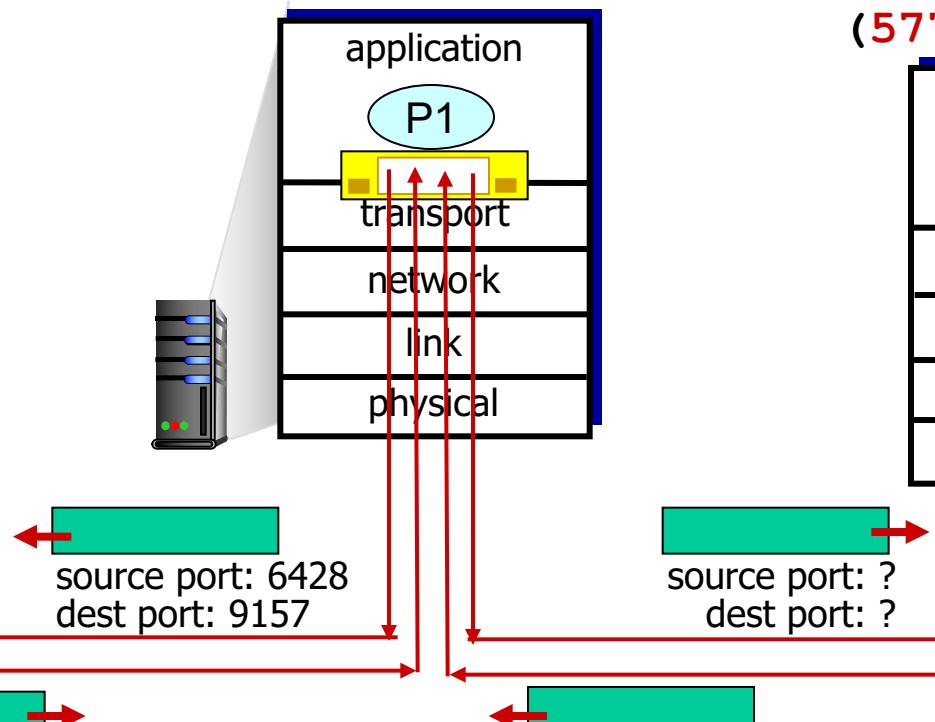
```
DatagramSocket  
mySocket2 = new  
DatagramSocket  
(9157);
```



source port: 9157
dest port: 6428

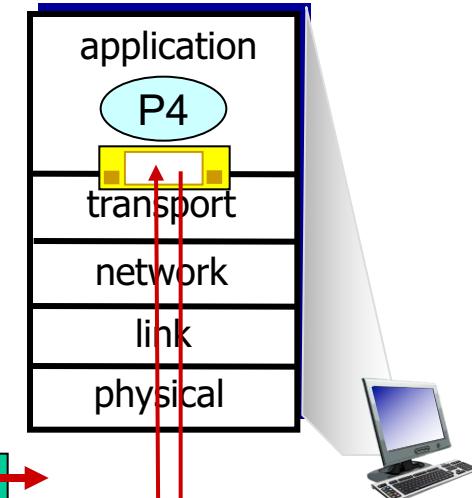
DatagramSocket

```
serverSocket = new  
DatagramSocket  
(6428);
```



source port: 6428
dest port: 9157

```
DatagramSocket  
mySocket1 = new  
DatagramSocket  
(5775);
```

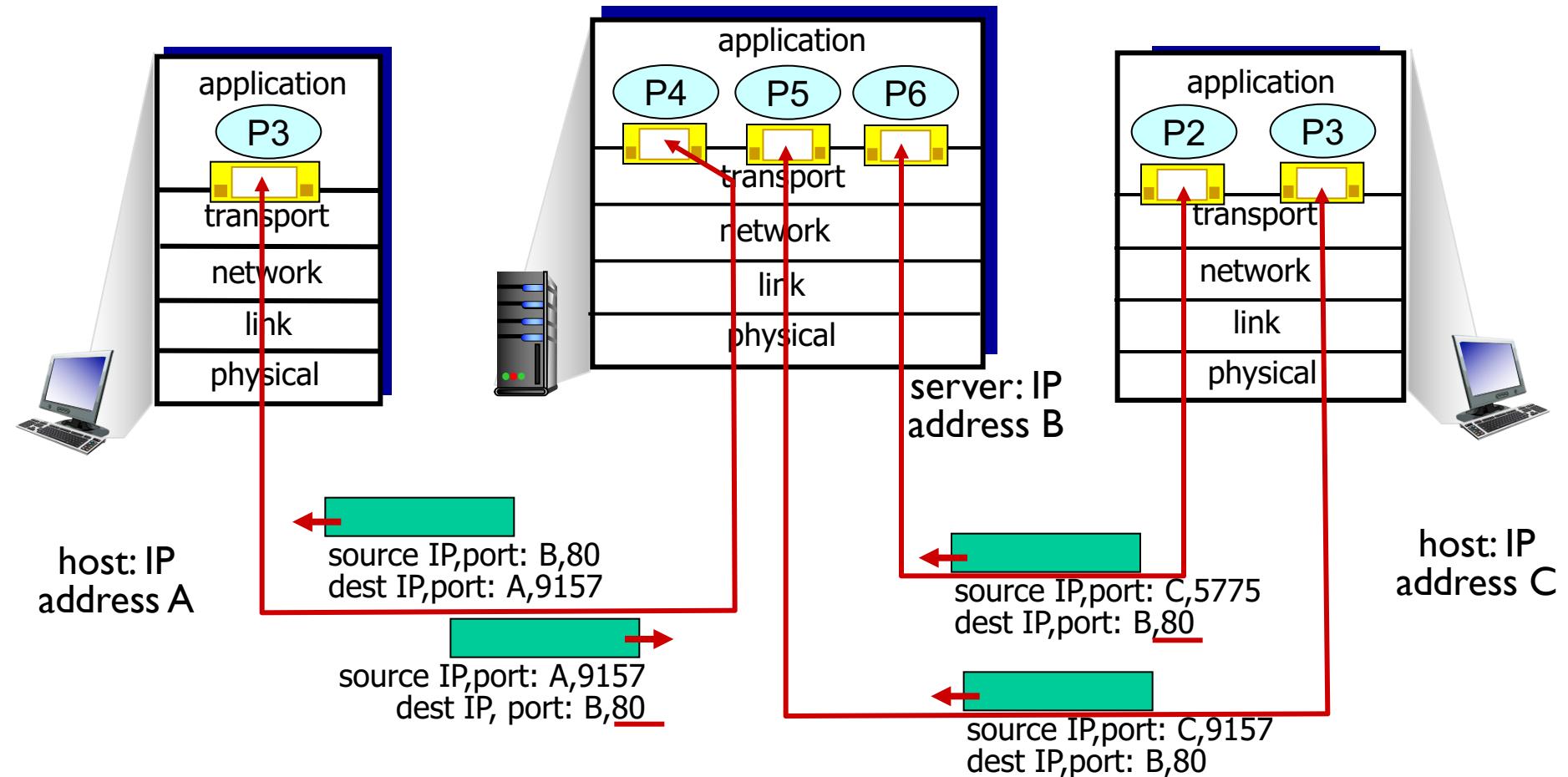


source port: ?
dest port: ?

Connection-oriented demux

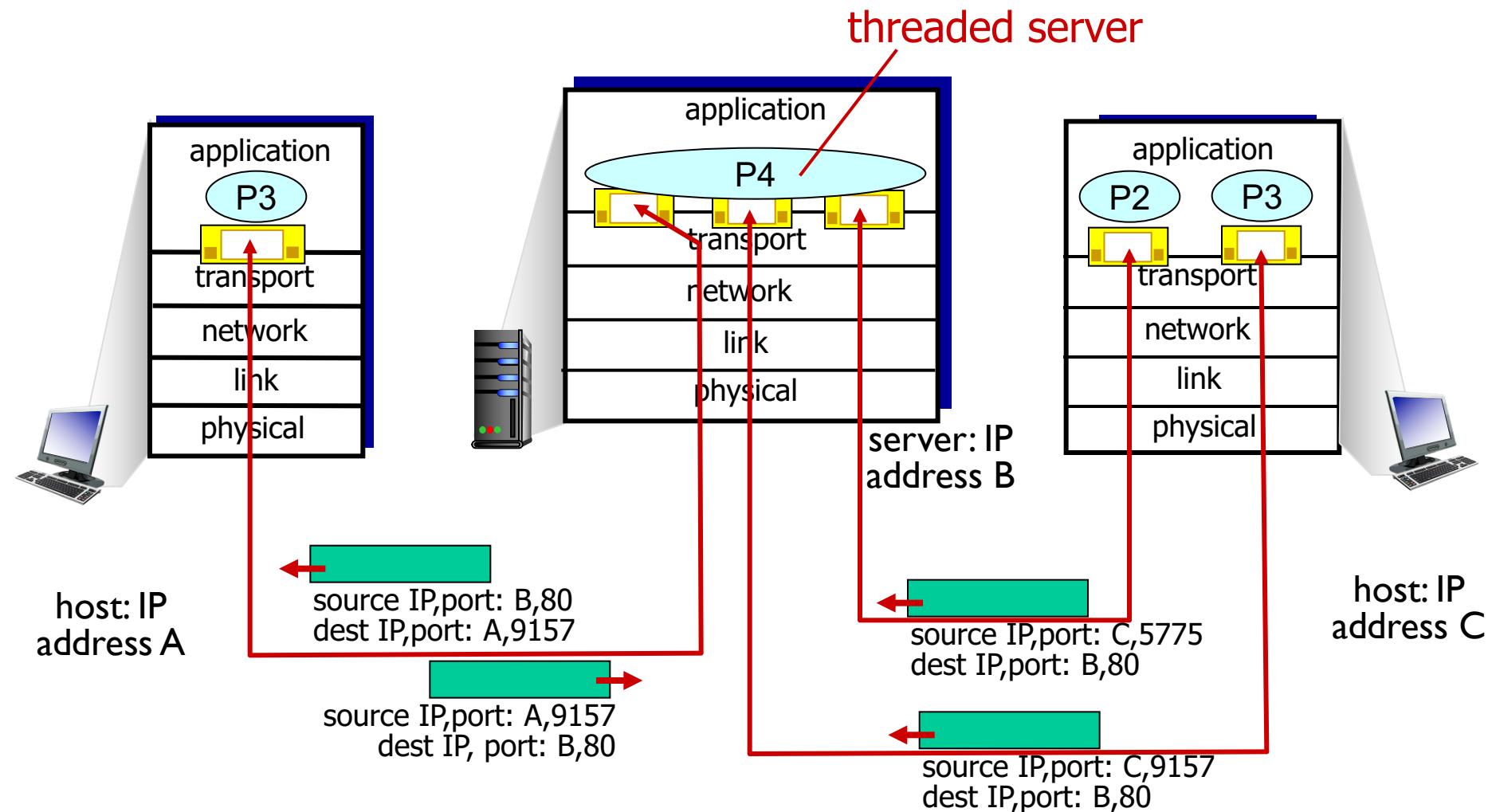
- TCP socket identified by 4-tuple:
 - source IP address
 - source port number
 - dest IP address
 - dest port number
- demux: receiver uses all four values to direct segment to appropriate socket
- server host may support many simultaneous TCP sockets:
 - each socket identified by its own 4-tuple
- web servers have different sockets for each connecting client
 - non-persistent HTTP will have different socket for each request

Connection-oriented demux: example



three segments, all destined to IP address: B,
dest port: 80 are demultiplexed to *different* sockets

Connection-oriented demux: example



Chapter 3 outline

3.1 transport-layer services

3.2 multiplexing and demultiplexing

3.3 connectionless transport: UDP

3.4 principles of reliable data transfer

3.5 connection-oriented transport: TCP

- segment structure
- reliable data transfer
- flow control
- connection management

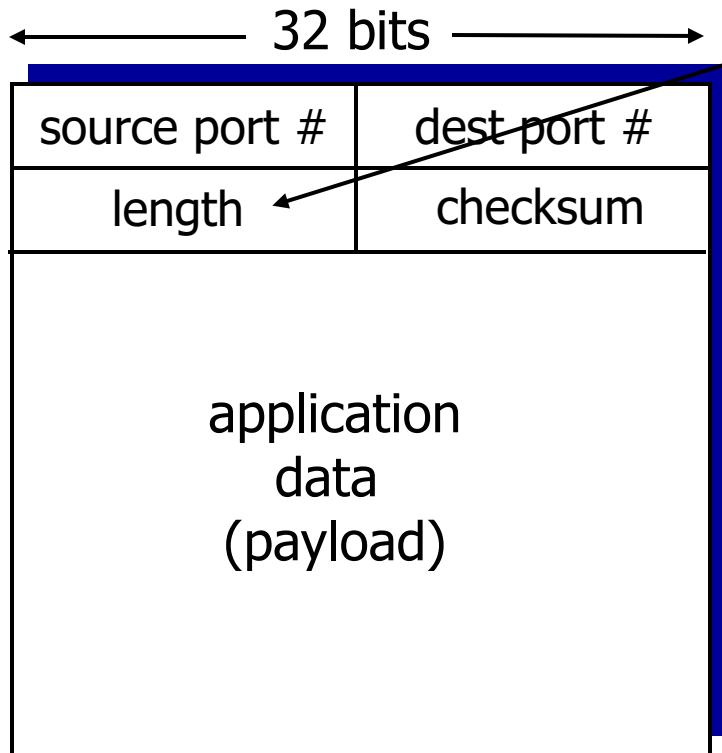
3.6 principles of congestion control

3.7 TCP congestion control

UDP: User Datagram Protocol [RFC 768]

- “no frills,” “bare bones” Internet transport protocol
- “best effort” service, UDP segments may be:
 - lost
 - delivered out-of-order to app
- **connectionless:**
 - no handshaking between UDP sender, receiver
 - each UDP segment handled independently of others
- UDP use:
 - streaming multimedia apps (loss tolerant, rate sensitive)
 - DNS
 - SNMP
- reliable transfer over UDP:
 - add reliability at application layer
 - application-specific error recovery!

UDP: segment header



UDP segment format

length, in bytes of
UDP segment,
including header

why is there a UDP?

- no connection establishment (which can add delay)
- simple: no connection state at sender, receiver
- small header size
- no congestion control: UDP can blast away as fast as desired

UDP checksum

Goal: detect “errors” (e.g., flipped bits) in transmitted segment

sender:

- treat segment contents, including header fields, as sequence of 16-bit integers
- checksum: addition (one's complement sum) of segment contents
- sender puts checksum value into UDP checksum field

receiver:

- compute checksum of received segment
- check if computed checksum equals checksum field value:
 - NO - error detected
 - YES - no error detected.
But maybe errors nonetheless? More later
....

Internet checksum: example

example: add two 16-bit integers

	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
<hr/>																
wraparound	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1
	<hr/>															
sum	1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	0
checksum	0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1

Note: when adding numbers, a carryout from the most significant bit needs to be added to the result

* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

Chapter 3 outline

3.1 transport-layer services

3.2 multiplexing and demultiplexing

3.3 connectionless transport: UDP

3.4 principles of reliable data transfer

3.5 connection-oriented transport: TCP

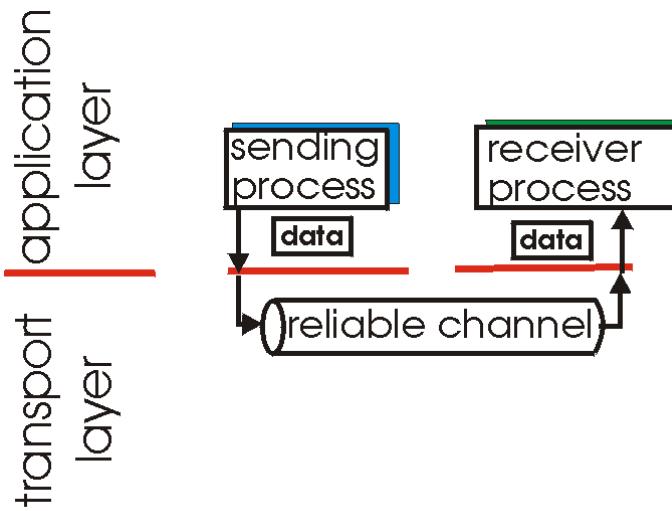
- segment structure
- reliable data transfer
- flow control
- connection management

3.6 principles of congestion control

3.7 TCP congestion control

Principles of reliable data transfer

- important in application, transport, link layers
 - top-10 list of important networking topics!

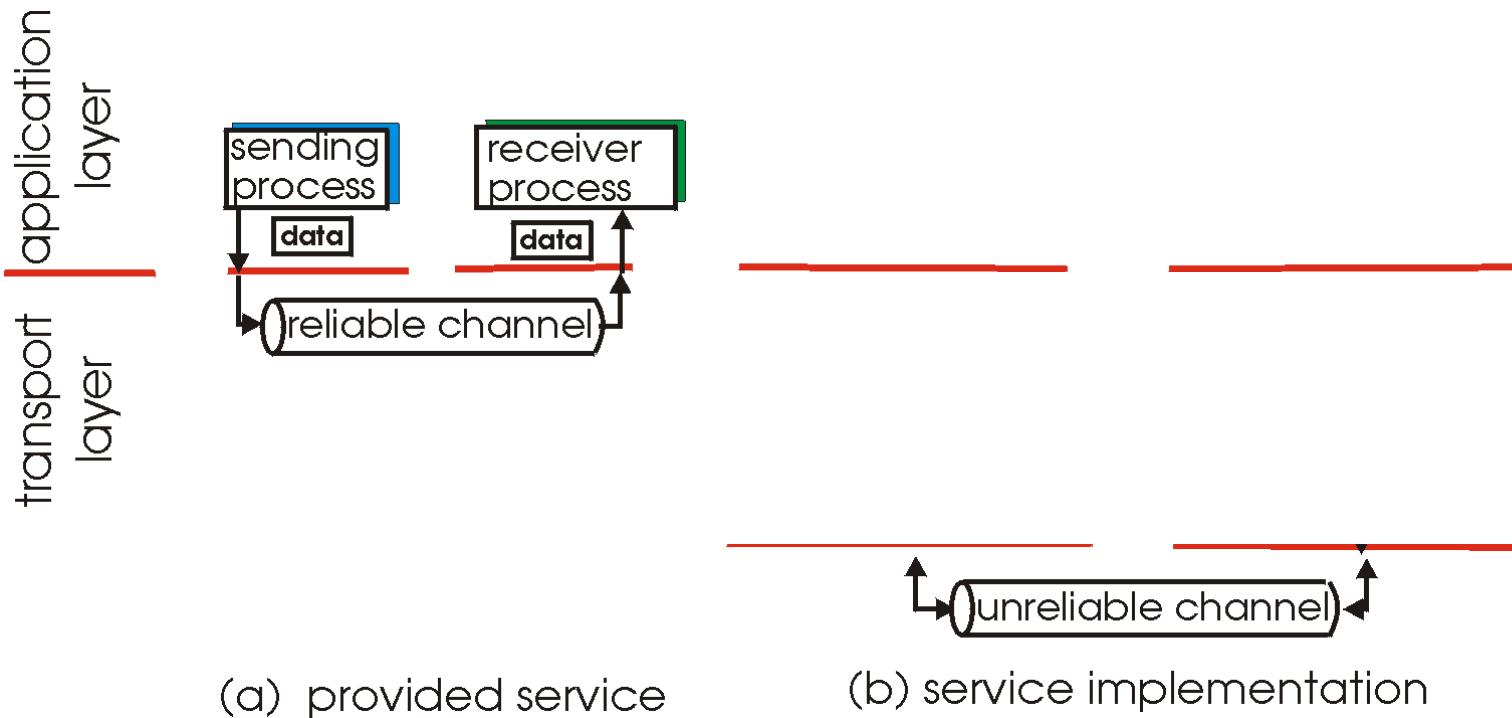


(a) provided service

- characteristics of unreliable channel will determine complexity of reliable data transfer protocol (rdt)

Principles of reliable data transfer

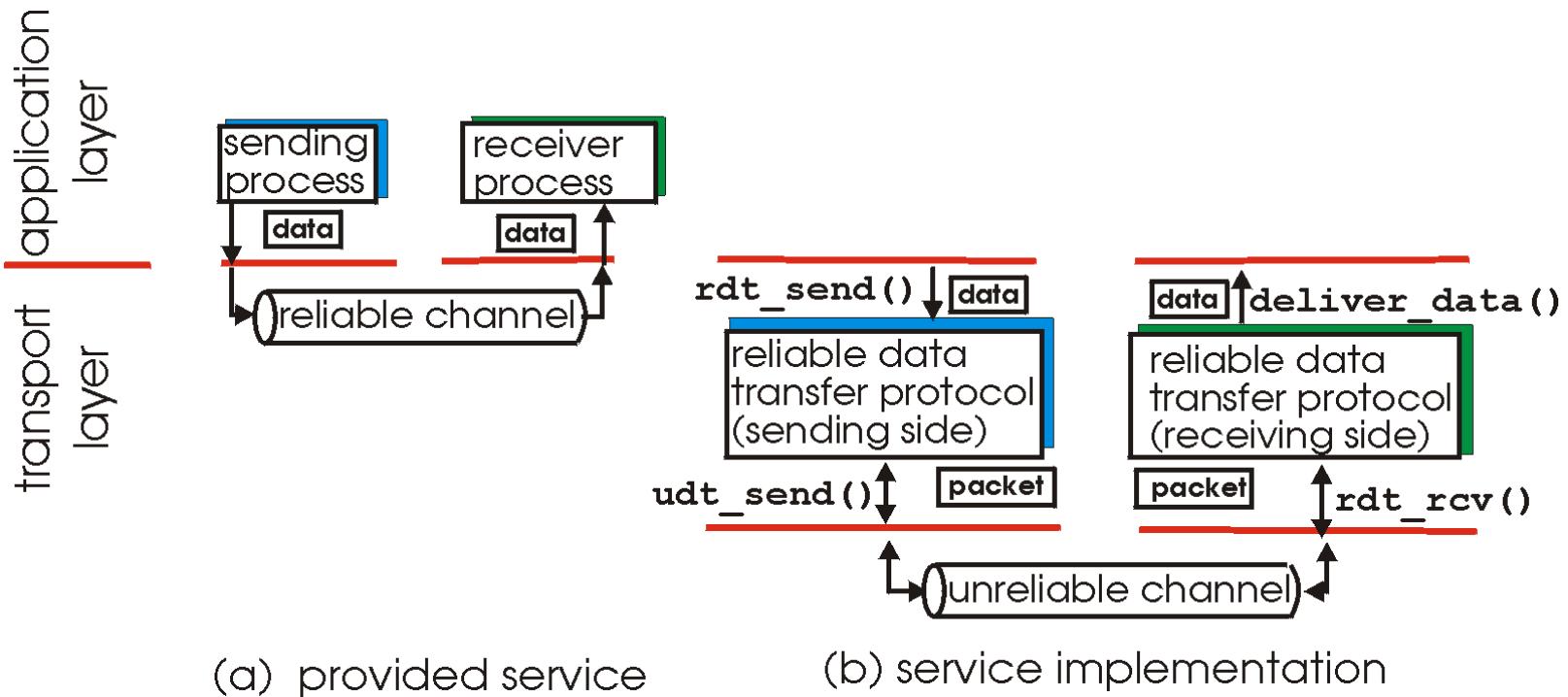
- important in application, transport, link layers
 - top-10 list of important networking topics!



- characteristics of unreliable channel will determine complexity of reliable data transfer protocol (rdt)

Principles of reliable data transfer

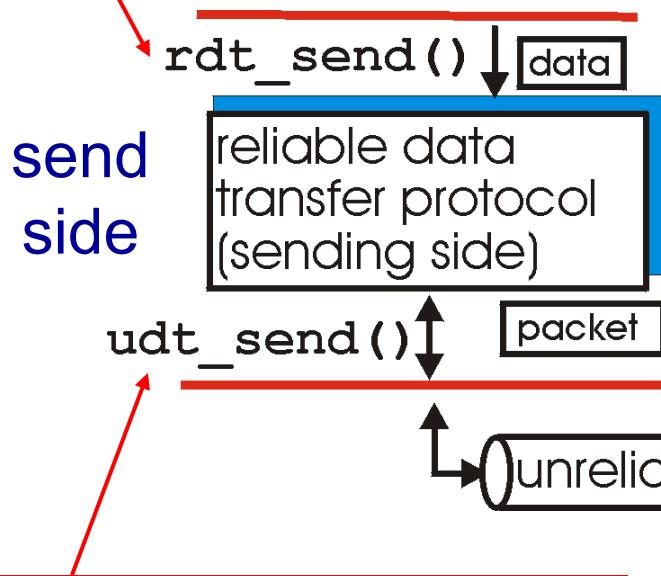
- important in application, transport, link layers
 - top-10 list of important networking topics!



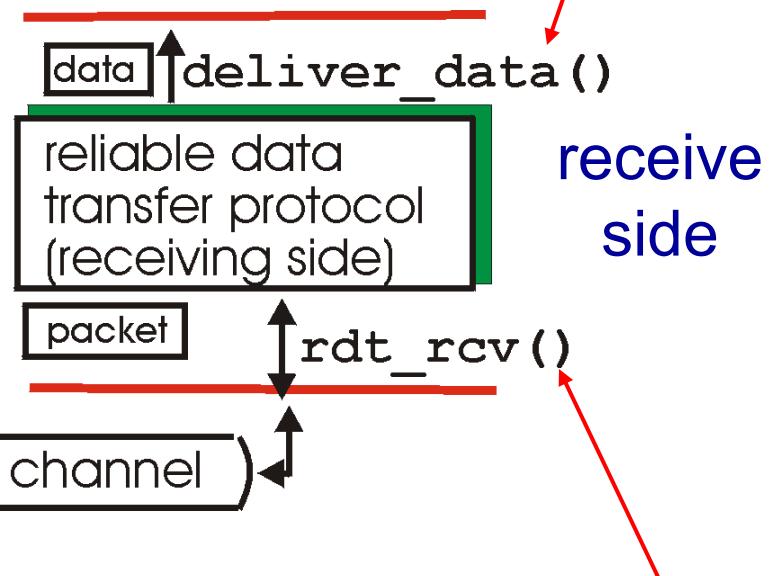
- characteristics of unreliable channel will determine complexity of reliable data transfer protocol (rdt)

Reliable data transfer: getting started

`rdt_send()`: called from above,
(e.g., by app.). Passed data to
deliver to receiver upper layer



`deliver_data()`: called by
rdt to deliver data to upper



`udt_send()`: called by rdt,
to transfer packet over
unreliable channel to receiver

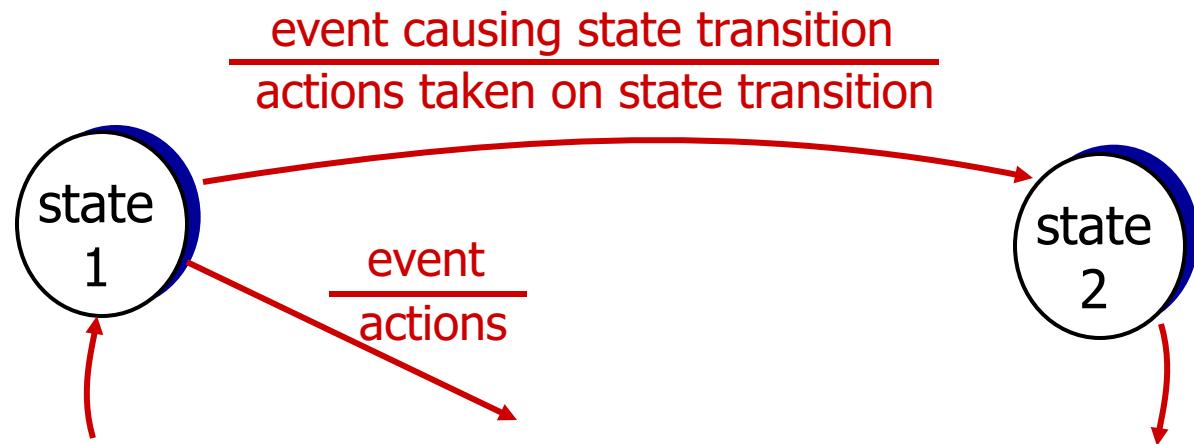
`rdt_rcv()`: called when packet
arrives on rcv-side of channel

Reliable data transfer: getting started

we'll:

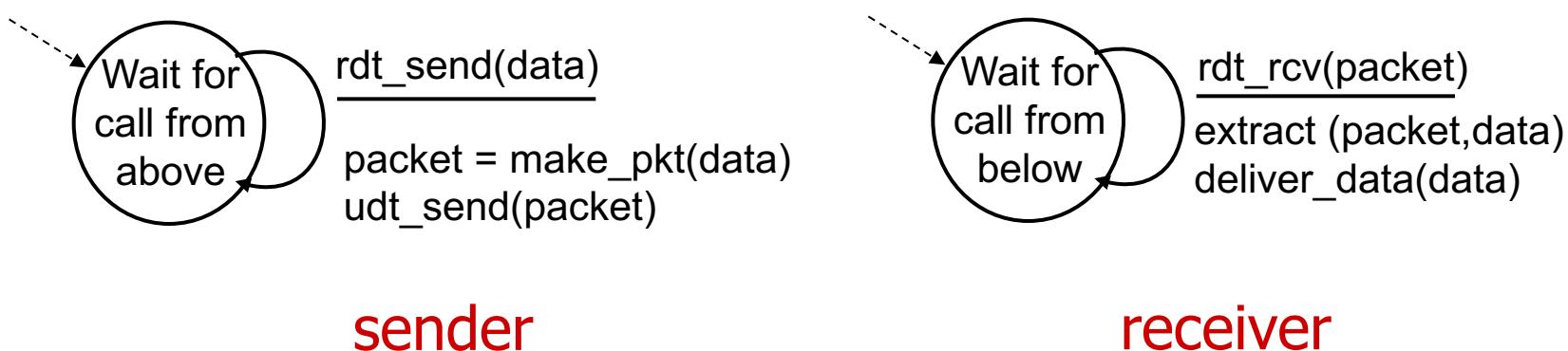
- incrementally develop sender, receiver sides of reliable data transfer protocol (rdt)
- consider only unidirectional data transfer
 - but control info will flow on both directions!
- use finite state machines (FSM) to specify sender, receiver

state: when in this “state” next state uniquely determined by next event



rdt1.0: reliable transfer over a reliable channel

- underlying channel perfectly reliable
 - no bit errors
 - no loss of packets
- separate FSMs for sender, receiver:
 - sender sends data into underlying channel
 - receiver reads data from underlying channel



rdt2.0: channel with bit errors

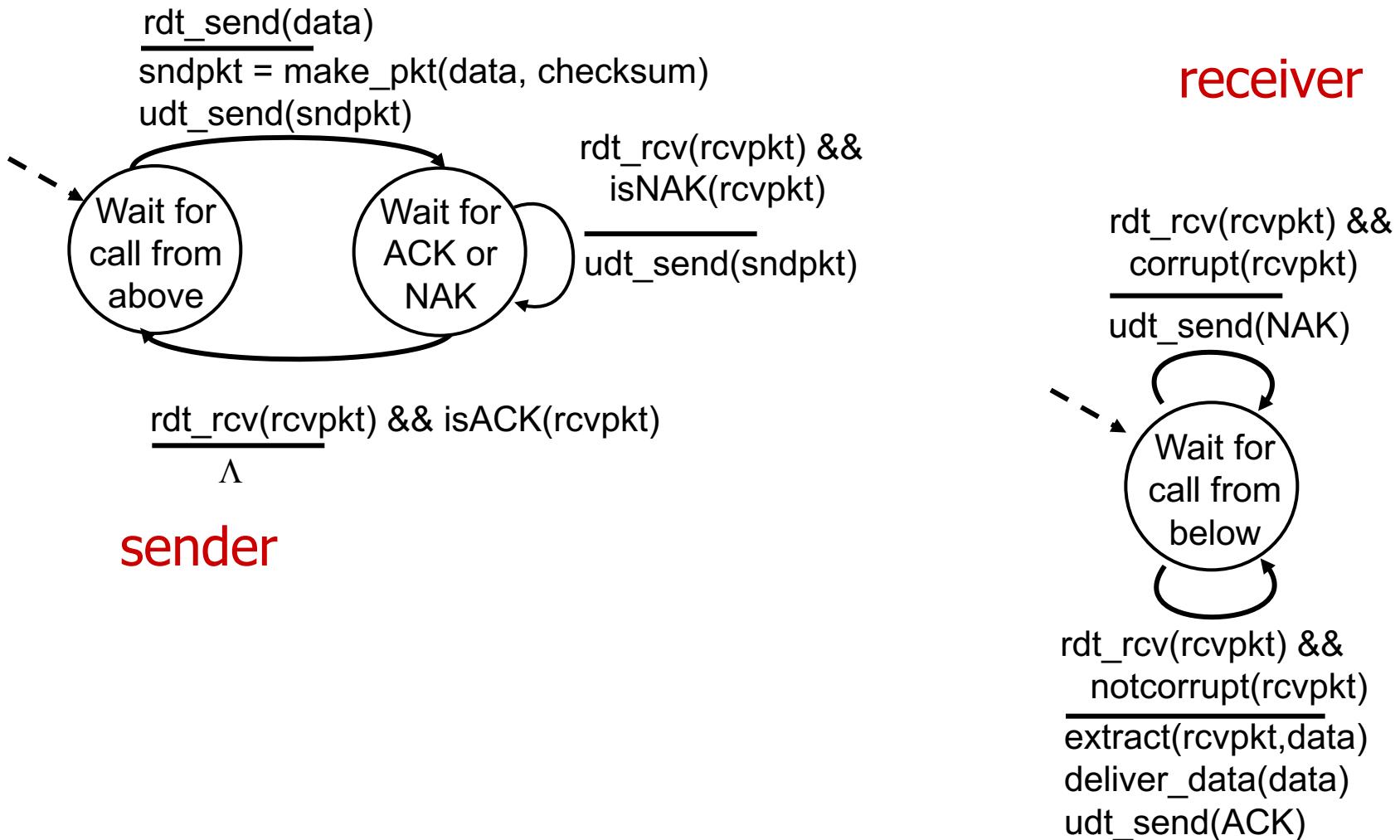
- underlying channel may flip bits in packet
 - checksum to detect bit errors
- the question: how to recover from errors:

*How do humans recover from “errors”
during conversation?*

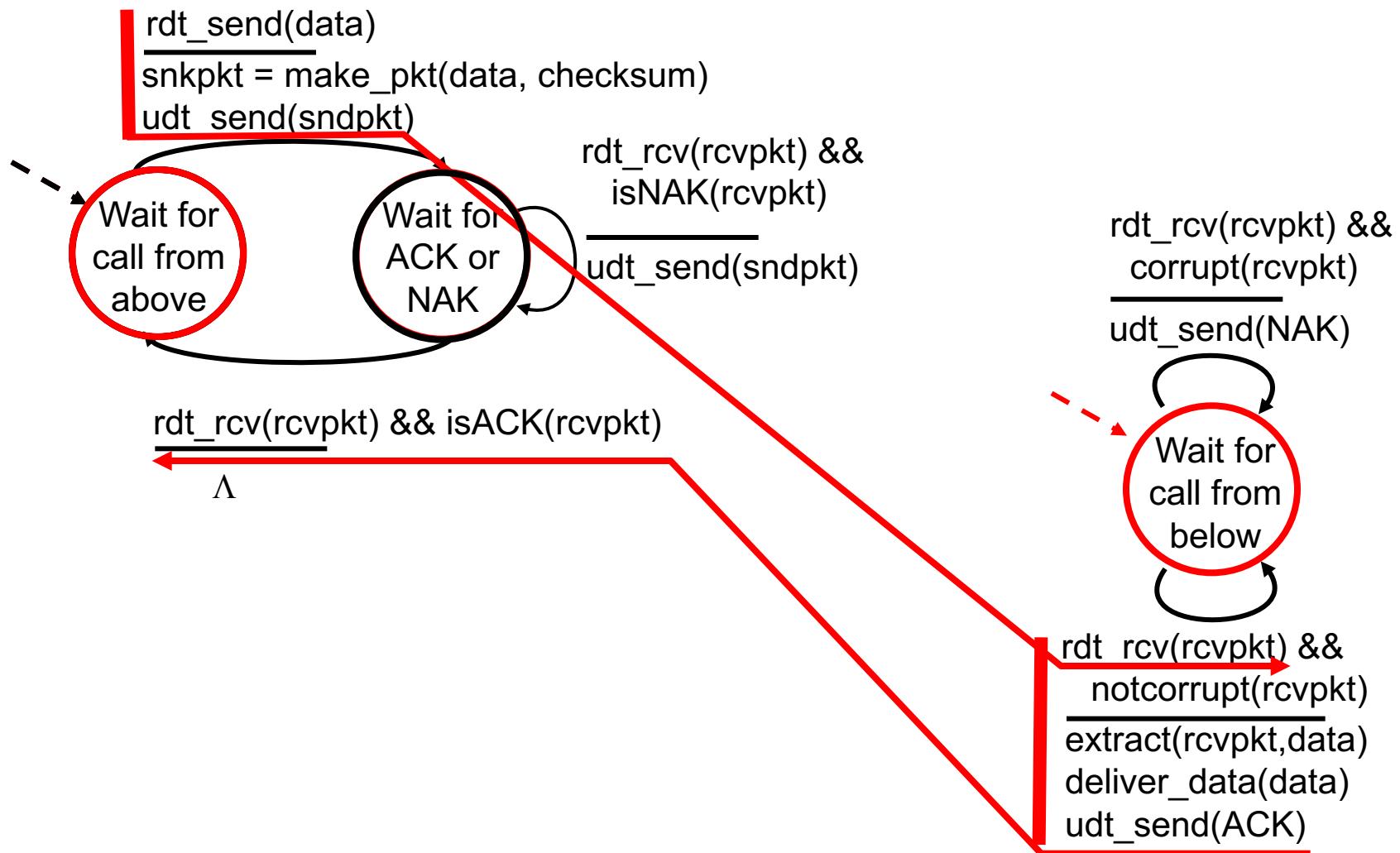
rdt2.0: channel with bit errors

- underlying channel may flip bits in packet
 - checksum to detect bit errors
- the question: how to recover from errors:
 - *acknowledgements (ACKs)*: receiver explicitly tells sender that pkt received OK
 - *negative acknowledgements (NAKs)*: receiver explicitly tells sender that pkt had errors
 - sender retransmits pkt on receipt of NAK
- new mechanisms in rdt2.0 (beyond rdt1.0):
 - error detection
 - feedback: control msgs (ACK,NAK) from receiver to sender

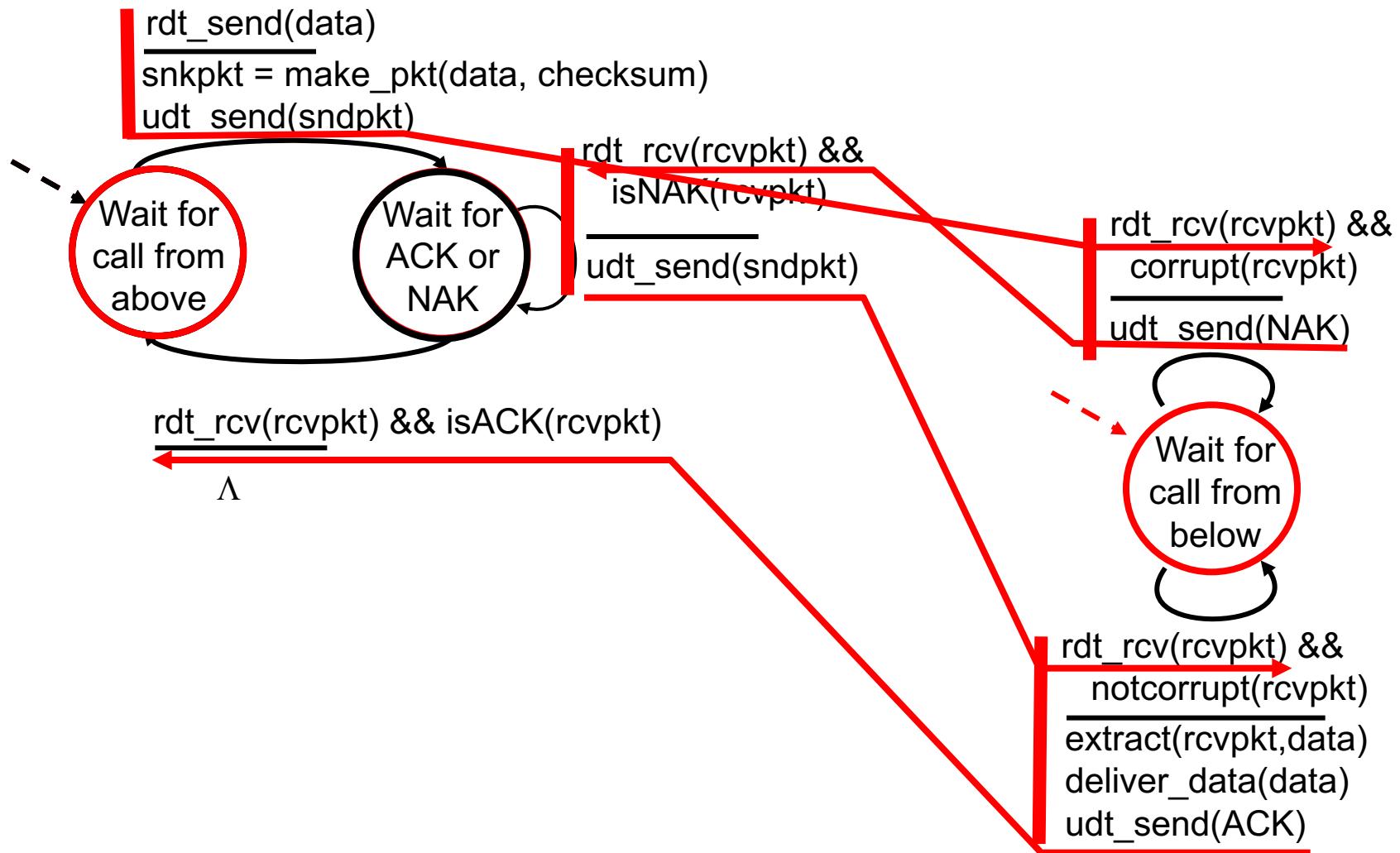
rdt2.0: FSM specification



rdt2.0: operation with no errors



rdt2.0: error scenario



rdt2.0 has a fatal flaw!

what happens if ACK/NAK corrupted?

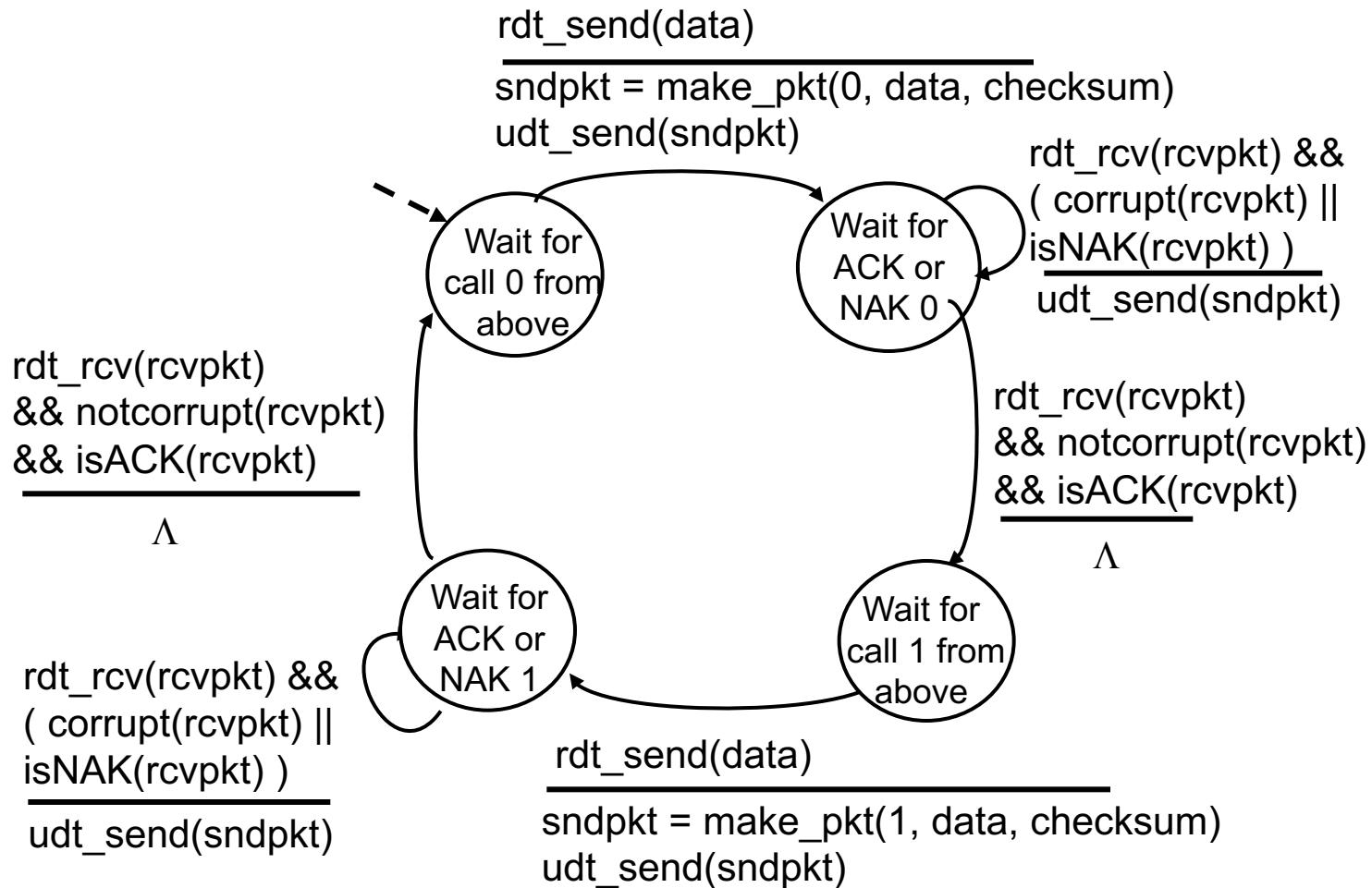
- sender doesn't know what happened at receiver!
- can't just retransmit: possible duplicate

handling duplicates:

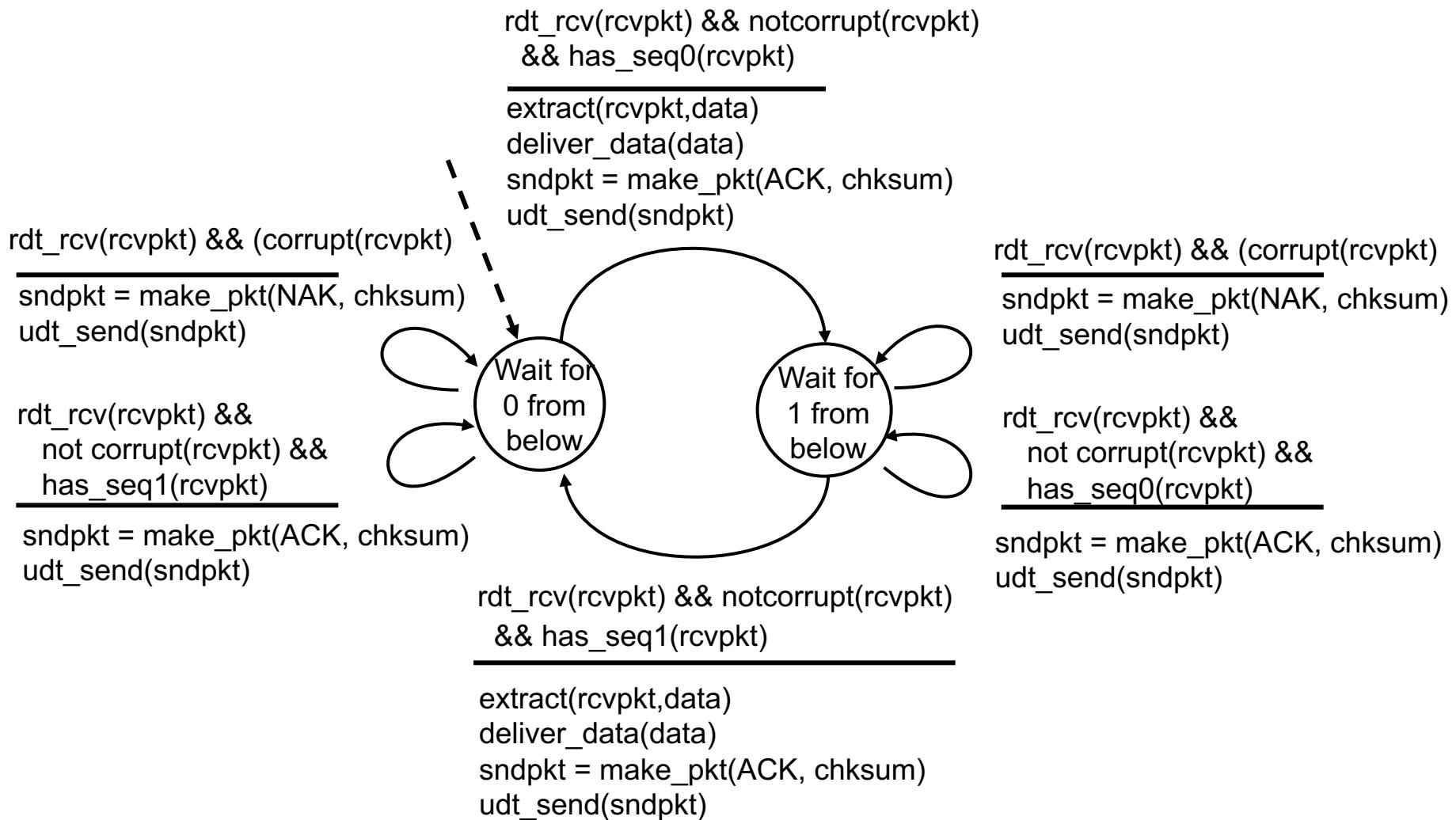
- sender retransmits current pkt if ACK/NAK corrupted
- sender adds *sequence number* to each pkt
- receiver discards (doesn't deliver up) duplicate pkt

stop and wait
sender sends one packet,
then waits for receiver
response

rdt2.1: sender, handles garbled ACK/NAKs



rdt2.1: receiver, handles garbled ACK/NAKs



rdt2.1: discussion

sender:

- seq # added to pkt
- two seq. #'s (0,1) will suffice. Why?
- must check if received ACK/NAK corrupted
- twice as many states
 - state must “remember” whether “expected” pkt should have seq # of 0 or 1

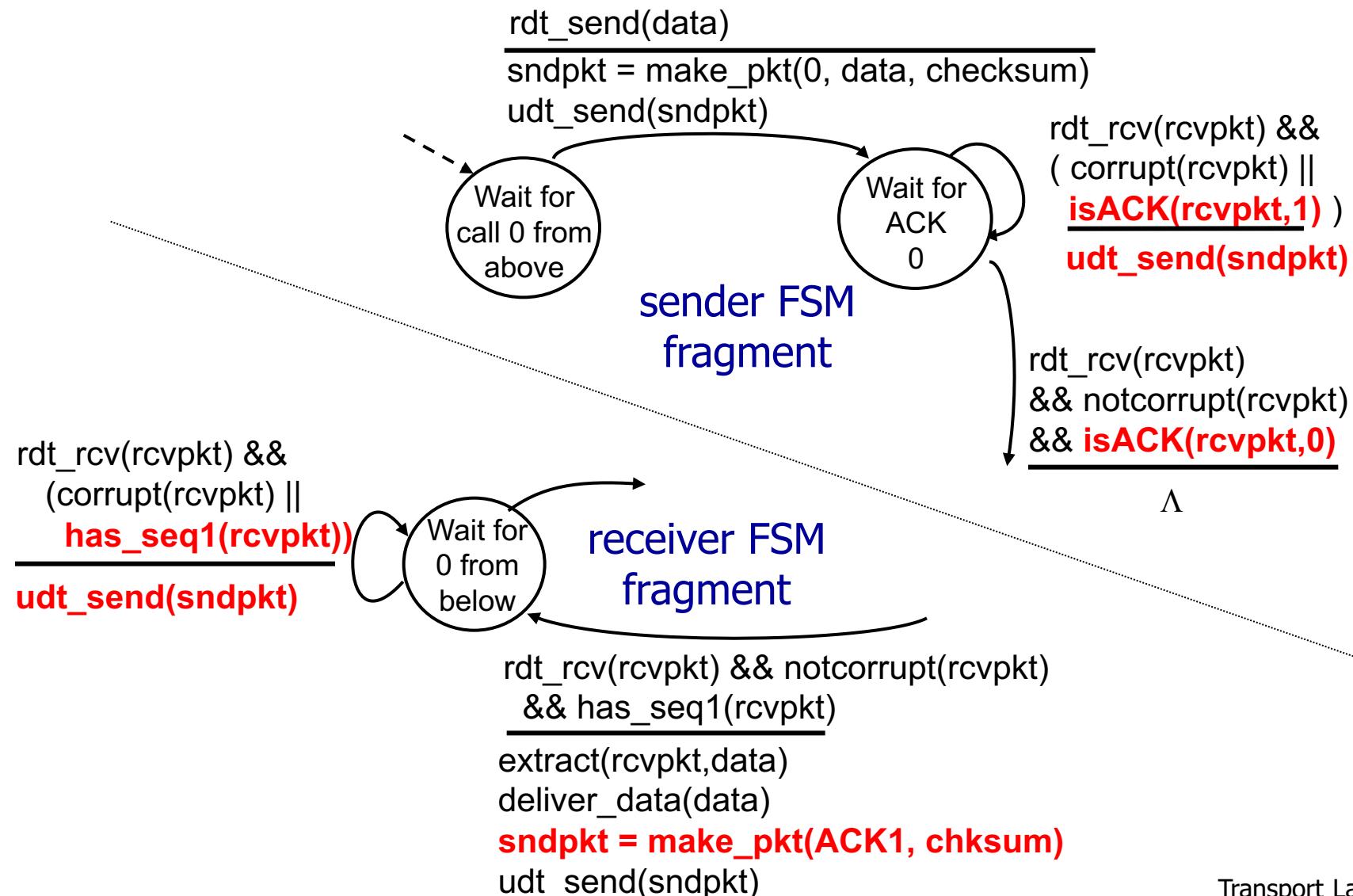
receiver:

- must check if received packet is duplicate
 - state indicates whether 0 or 1 is expected pkt seq #
- note: receiver can *not* know if its last ACK/NAK received OK at sender

rdt2.2: a NAK-free protocol

- same functionality as rdt2.1, using ACKs only
- instead of NAK, receiver sends ACK for last pkt received OK
 - receiver must *explicitly* include seq # of pkt being ACKed
- duplicate ACK at sender results in same action as NAK: *retransmit current pkt*

rdt2.2: sender, receiver fragments



rdt3.0: channels with errors and loss

new assumption:

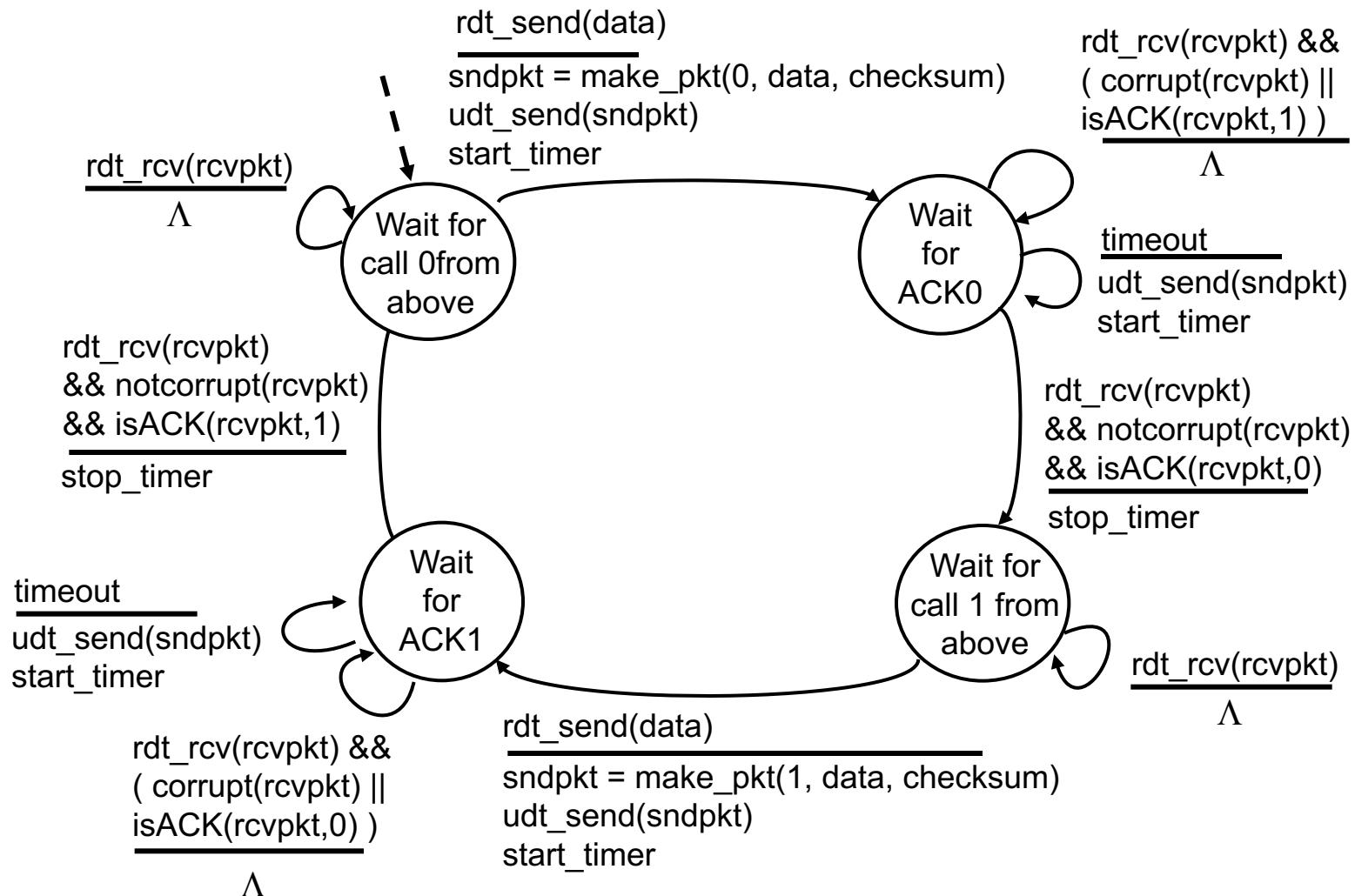
underlying channel can also lose packets (data, ACKs)

- checksum, seq. #, ACKs, retransmissions will be of help ... but not enough

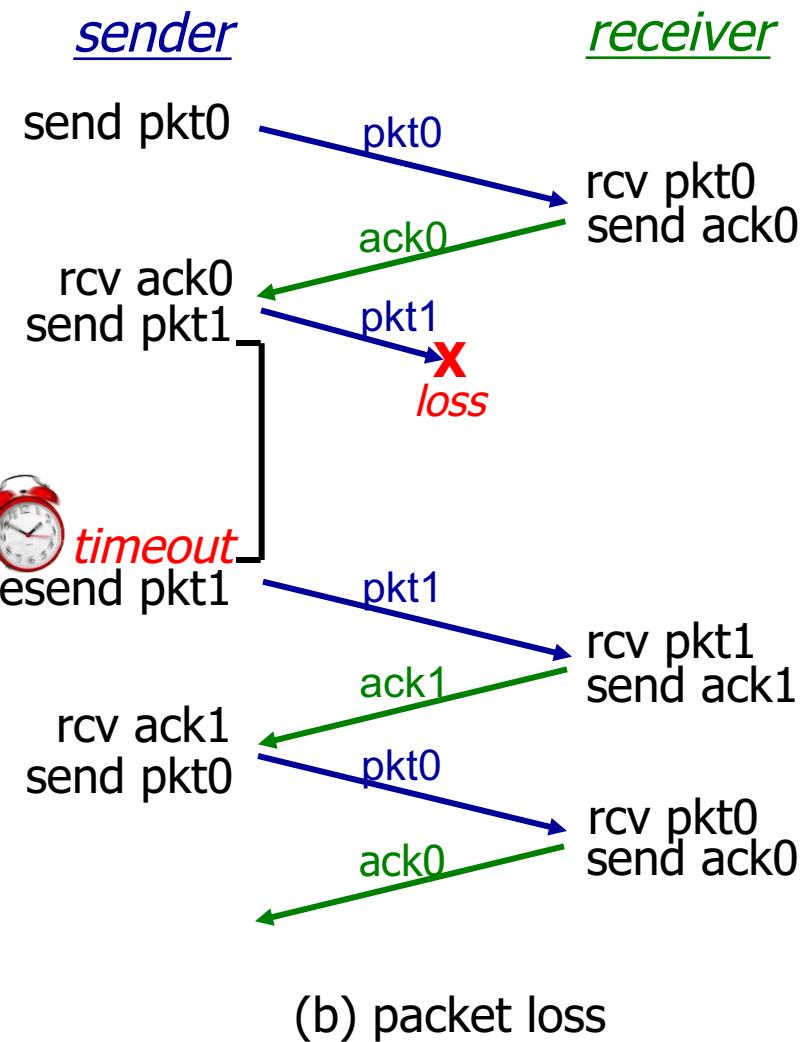
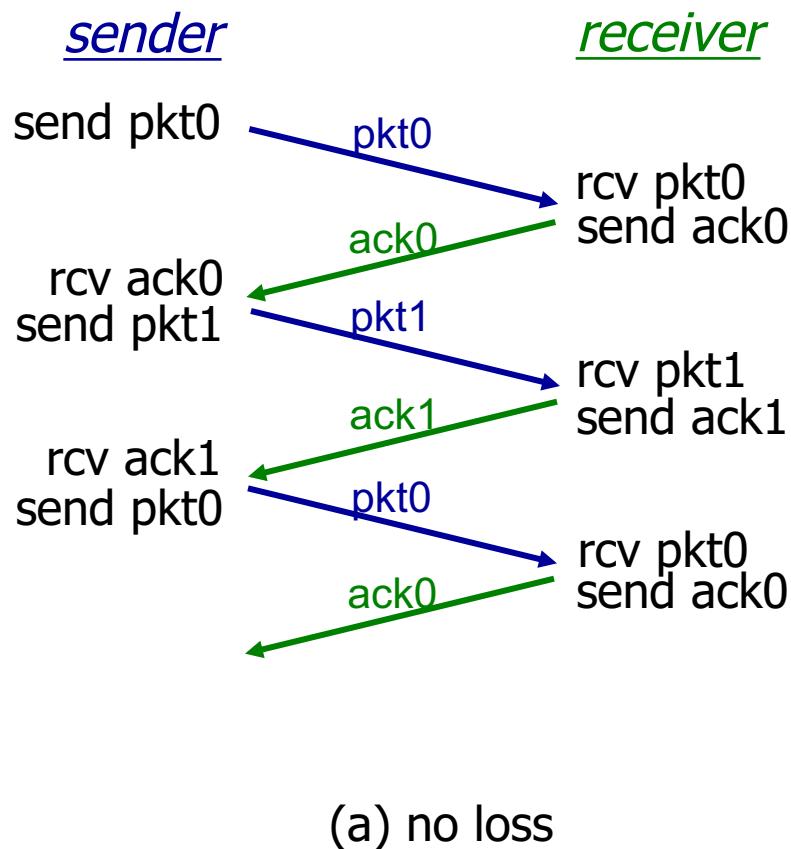
approach: sender waits “reasonable” amount of time for ACK

- retransmits if no ACK received in this time
- if pkt (or ACK) just delayed (not lost):
 - retransmission will be duplicate, but seq. #'s already handles this
 - receiver must specify seq # of pkt being ACKed
- requires countdown timer

rdt3.0 sender

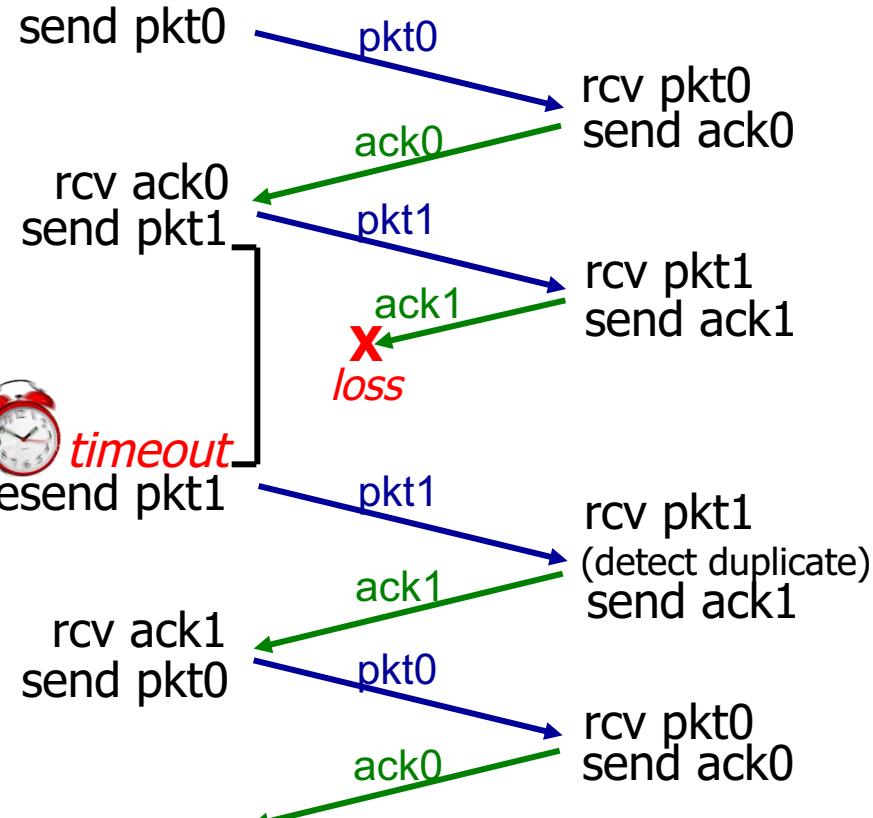


rdt3.0 in action



rdt3.0 in action

sender



(c) ACK loss

sender

send pkt0

rcv ack0
send pkt1

resend pkt1

rcv ack1
send pkt0

rcv ack1
send pkt0

rcv ack1
send pkt0

rcv ack0
send pkt0

rcv ack0
send pkt0

receiver

rcv pkt0
send ack0

rcv pkt1
send ack1

rcv pkt1
(detect duplicate)
send ack1

rcv pkt0
send ack0

rcv pkt0
(detect duplicate)
send ack0



timeout

pkt0

pkt1

pkt0

pkt1

pkt0

pkt0

ack1

ack0

Performance of rdt3.0

- rdt3.0 is correct, but performance stinks
- e.g.: 1 Gbps link, 15 ms prop. delay, 8000 bit packet:

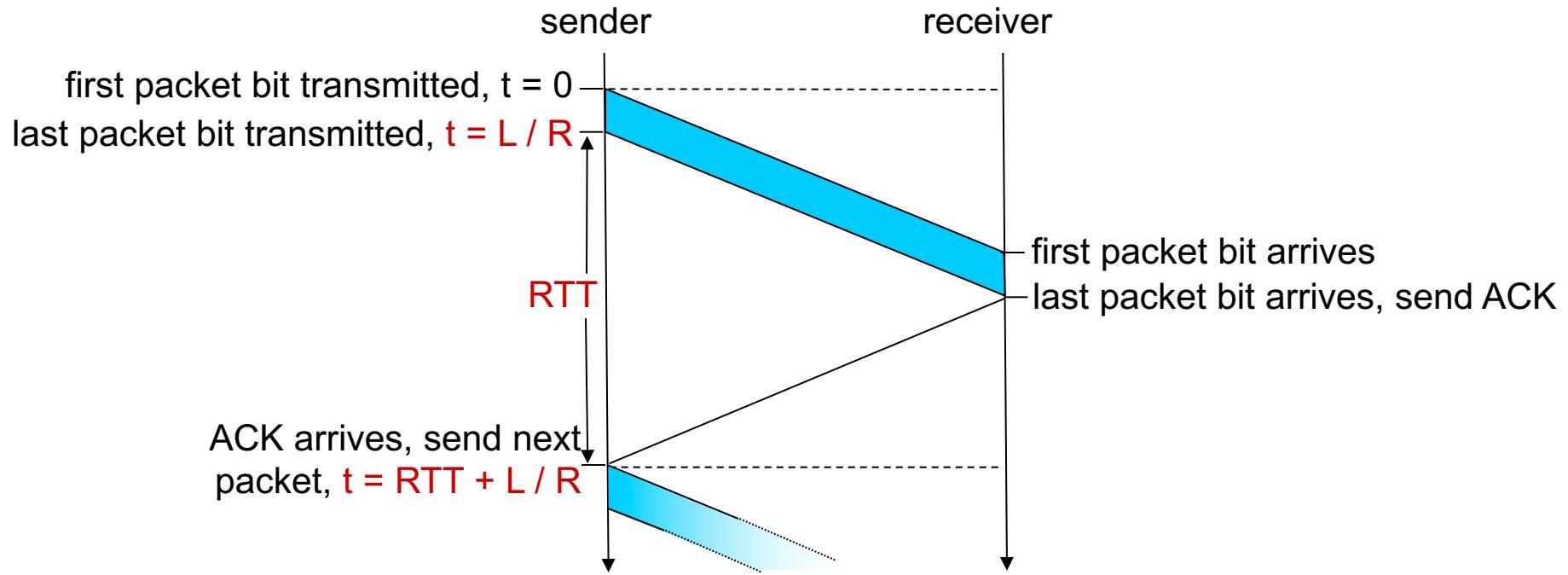
$$D_{trans} = \frac{L}{R} = \frac{8000 \text{ bits}}{10^9 \text{ bits/sec}} = 8 \text{ microsecs}$$

- U_{sender} : *utilization* – fraction of time sender busy sending

$$U_{\text{sender}} = \frac{L/R}{RTT + L/R} = \frac{.008}{30.008} = 0.00027$$

- if RTT=30 msec, 1KB pkt every 30 msec: 33kB/sec thruput over 1 Gbps link
- network protocol limits use of physical resources!

rdt3.0: stop-and-wait operation

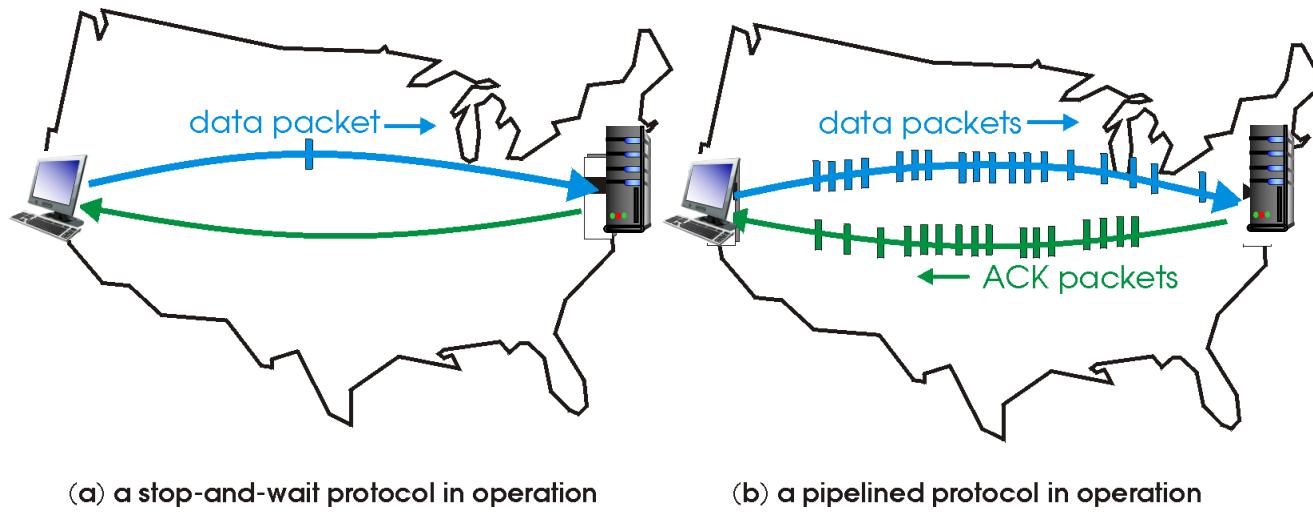


$$U_{\text{sender}} = \frac{L / R}{RTT + L / R} = \frac{.008}{30.008} = 0.00027$$

Pipelined protocols

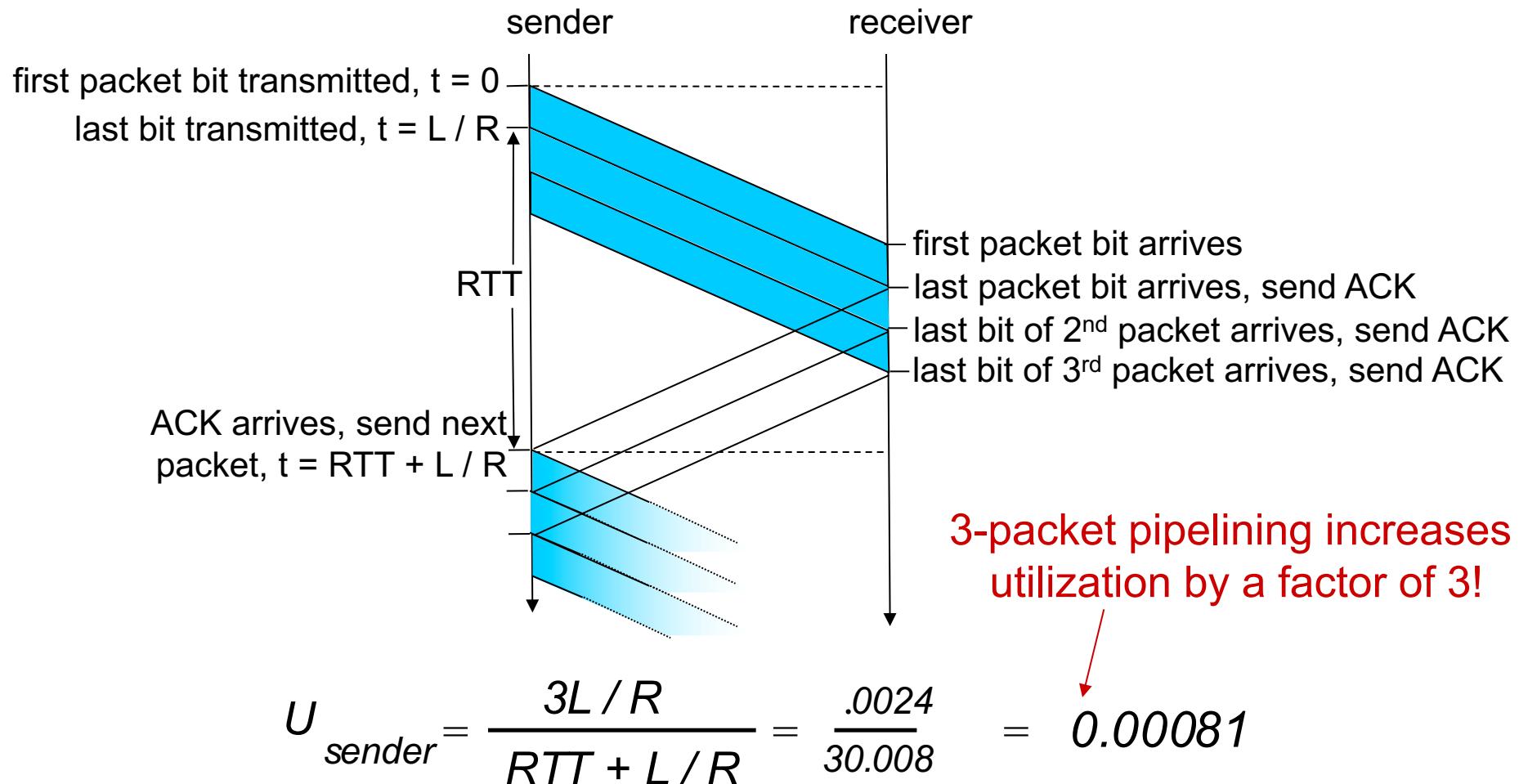
pipelining: sender allows multiple, “in-flight”, yet-to-be-acknowledged pkts

- range of sequence numbers must be increased
- buffering at sender and/or receiver



- two generic forms of pipelined protocols: *go-Back-N*, *selective repeat*

Pipelining: increased utilization



Pipelined protocols: overview

Go-back-N:

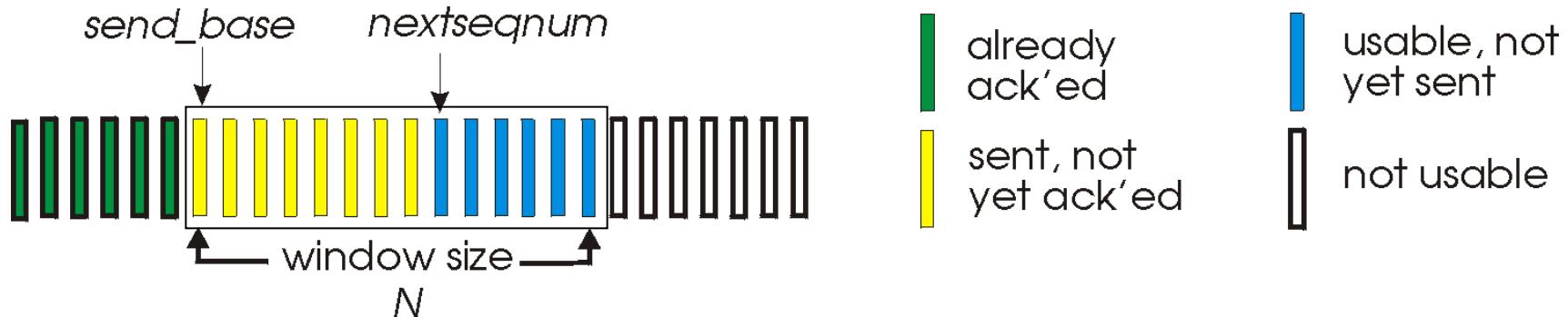
- sender can have up to N unacked packets in pipeline
- receiver only sends *cumulative ack*
 - doesn't ack packet if there's a gap
- sender has timer for oldest unacked packet
 - when timer expires, retransmit *all* unacked packets

Selective Repeat:

- sender can have up to N unacked packets in pipeline
- rcvr sends *individual ack* for each packet
- sender maintains timer for each unacked packet
 - when timer expires, retransmit only that unacked packet

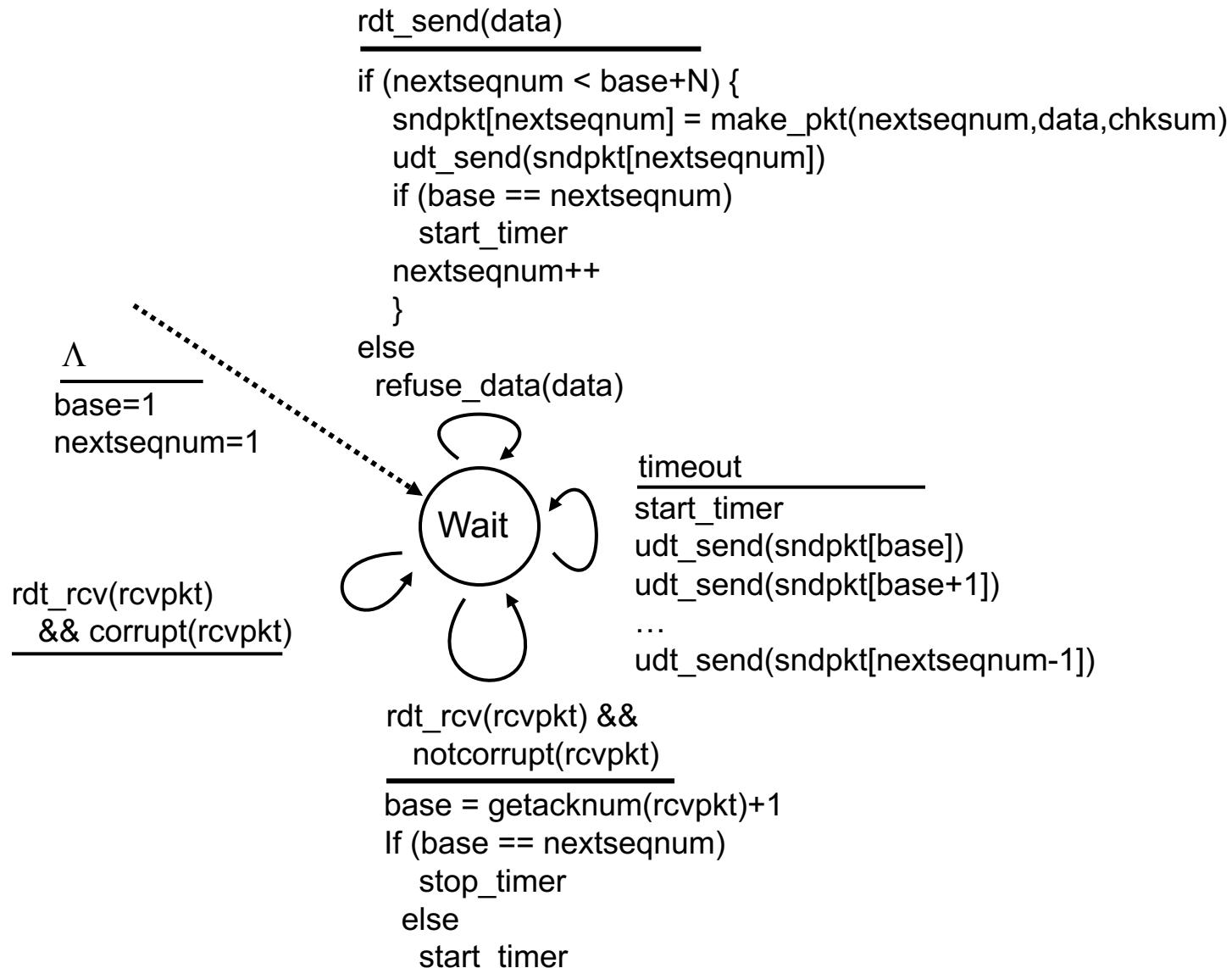
Go-Back-N: sender

- k-bit seq # in pkt header
- “window” of up to N, consecutive unack’ ed pkts allowed

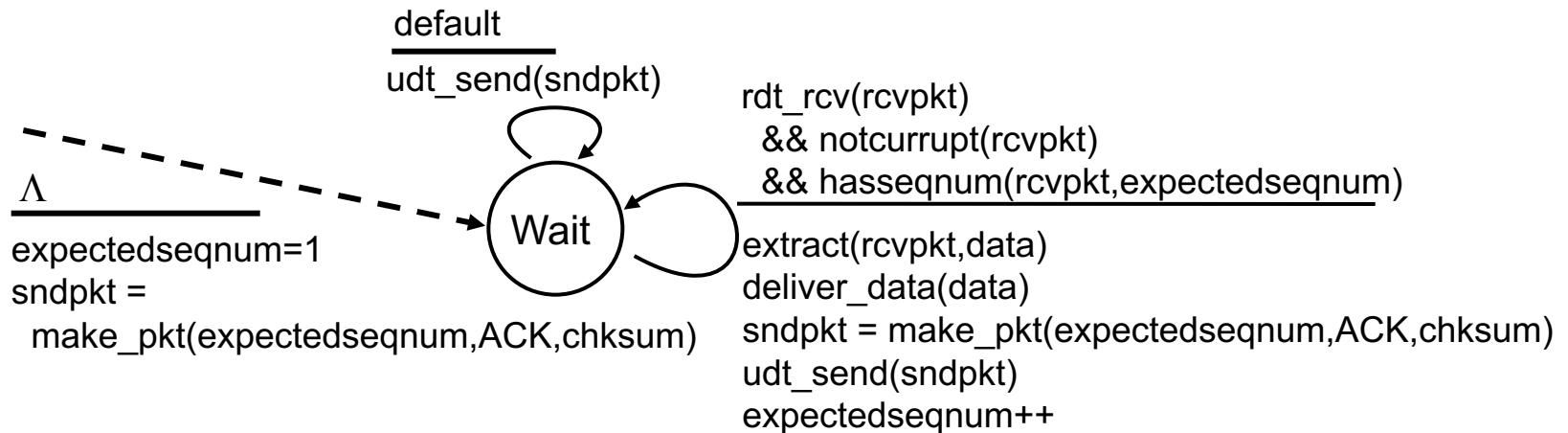


- ACK(n):ACKs all pkts up to, including seq # n - “*cumulative ACK*”
 - may receive duplicate ACKs (see receiver)
- timer for oldest in-flight pkt
- $\text{timeout}(n)$: retransmit packet n and all higher seq # pkts in window

GBN: sender extended FSM



GBN: receiver extended FSM



ACK-only: always send ACK for correctly-received
pkt with highest *in-order* seq #

- may generate duplicate ACKs
 - need only remember **expectedseqnum**
- **out-of-order pkt:**
 - discard (don't buffer): *no receiver buffering!*
 - re-ACK pkt with highest in-order seq #

GBN in action

sender window (N=4)

0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8

sender

send pkt0

send pkt1

send pkt2

send pkt3

(wait)

0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8

rcv ack0, send pkt4

rcv ack1, send pkt5

ignore duplicate ACK



pkt 2 timeout

send pkt2

send pkt3

send pkt4

send pkt5

receiver

receive pkt0, send ack0
receive pkt1, send ack1

receive pkt3, discard,
(re)send ack1

receive pkt4, discard,
(re)send ack1

receive pkt5, discard,
(re)send ack1

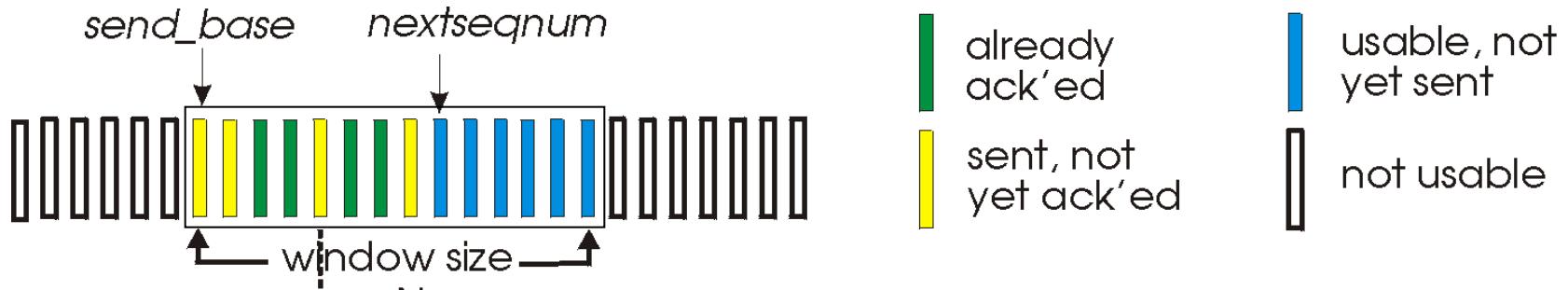
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8

rcv pkt2, deliver, send ack2
rcv pkt3, deliver, send ack3
rcv pkt4, deliver, send ack4
rcv pkt5, deliver, send ack5

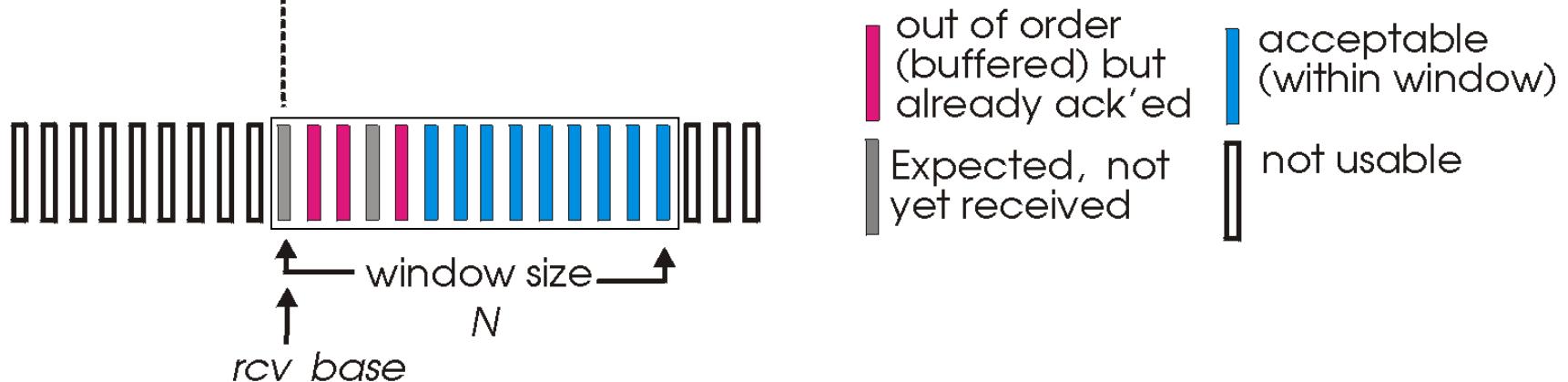
Selective repeat

- receiver *individually* acknowledges all correctly received pkts
 - buffers pkts, as needed, for eventual in-order delivery to upper layer
- sender only resends pkts for which ACK not received
 - sender timer for each unACKed pkt
- sender window
 - N consecutive seq #'s
 - limits seq #'s of sent, unACKed pkts

Selective repeat: sender, receiver windows



(a) sender view of sequence numbers



(b) receiver view of sequence numbers

Selective repeat

sender

data from above:

- if next available seq # in window, send pkt

timeout(n):

- resend pkt n, restart timer

ACK(n) in [sendbase,sendbase+N]:

- mark pkt n as received
- if n smallest unACKed pkt, advance window base to next unACKed seq #

receiver

pkt n in [rcvbase, rcvbase+N-1]

- send ACK(n)
- out-of-order: buffer
- in-order: deliver (also deliver buffered, in-order pkts), advance window to next not-yet-received pkt

pkt n in [rcvbase-N,rcvbase-1]

- ACK(n)

otherwise:

- ignore

Selective repeat in action

sender window (N=4)

0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8

sender

send pkt0
send pkt1
send pkt2
send pkt3
(wait)

receiver

receive pkt0, send ack0
receive pkt1, send ack1

receive pkt3, buffer,
send ack3

receive pkt4, buffer,
send ack4
receive pkt5, buffer,
send ack5

0 1 2 3 4 5 6 7 8 rcv ack0, send pkt4
0 1 2 3 4 5 6 7 8 rcv ack1, send pkt5

record ack3 arrived



pkt 2 timeout

send pkt2

record ack4 arrived

record ack5 arrived

0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8

rcv pkt2; deliver pkt2,
pkt3, pkt4, pkt5; send ack2

Q: what happens when ack2 arrives?

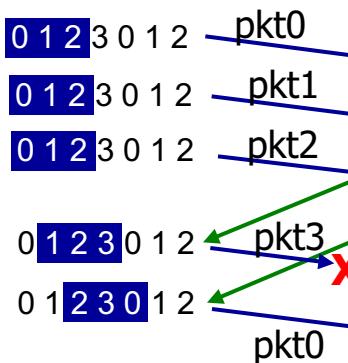
Selective repeat: dilemma

example:

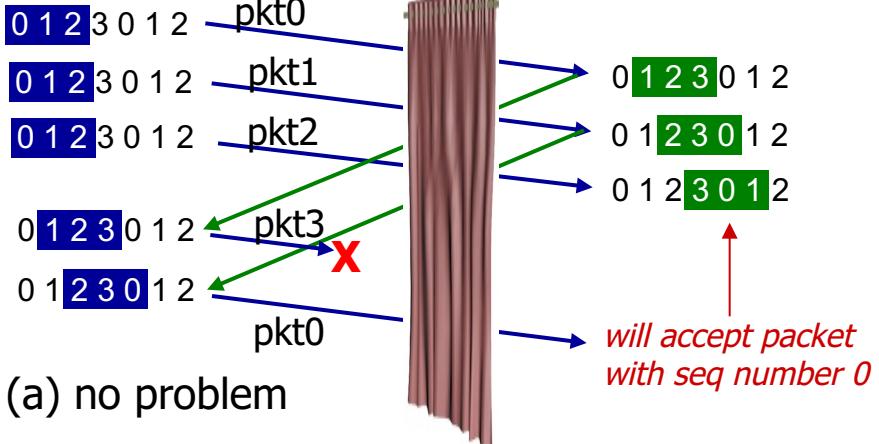
- seq #'s: 0, 1, 2, 3
- window size=3
- receiver sees no difference in two scenarios!
- duplicate data accepted as new in (b)

Q: what relationship between seq # size and window size to avoid problem in (b)?

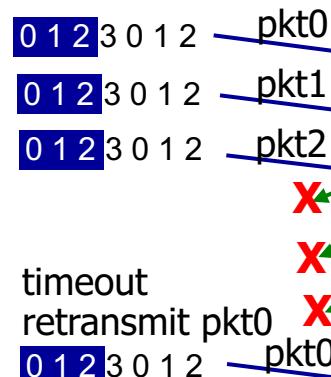
sender window
(after receipt)



receiver window
(after receipt)



*receiver can't see sender side.
receiver behavior identical in both cases!
something's (very) wrong!*



(b) oops!

Chapter 3 outline

3.1 transport-layer services

3.2 multiplexing and demultiplexing

3.3 connectionless transport: UDP

3.4 principles of reliable data transfer

3.5 connection-oriented transport: TCP

- segment structure
- reliable data transfer
- flow control
- connection management

3.6 principles of congestion control

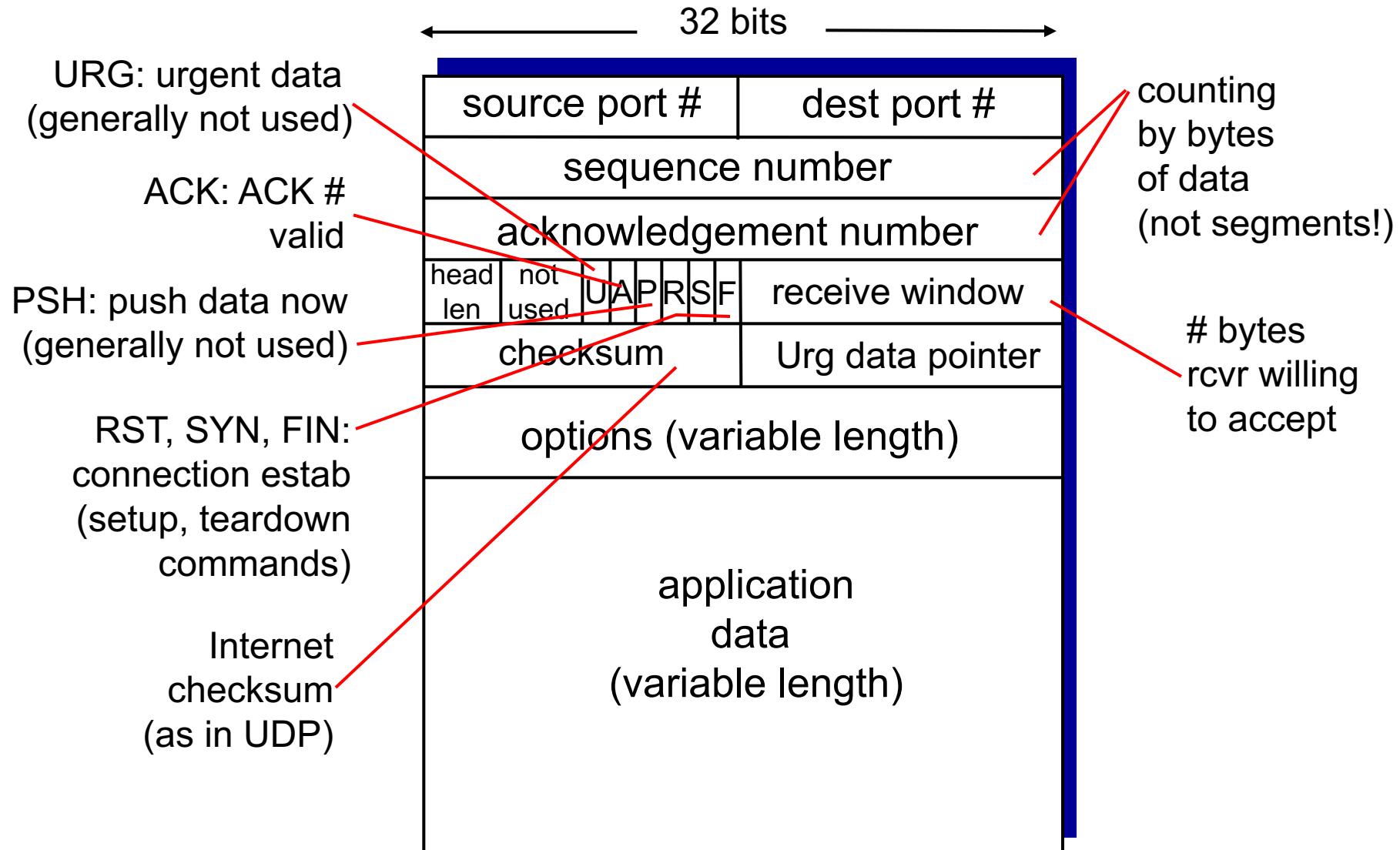
3.7 TCP congestion control

TCP: Overview

RFCs: 793, 1122, 1323, 2018, 2581

- **point-to-point:**
 - one sender, one receiver
- **reliable, in-order byte steam:**
 - no “message boundaries”
- **pipelined:**
 - TCP congestion and flow control set window size
- **full duplex data:**
 - bi-directional data flow in same connection
 - MSS: maximum segment size
- **connection-oriented:**
 - handshaking (exchange of control msgs) inits sender, receiver state before data exchange
- **flow controlled:**
 - sender will not overwhelm receiver

TCP segment structure



TCP seq. numbers, ACKs

sequence numbers:

- byte stream “number” of first byte in segment’s data

acknowledgements:

- seq # of next byte expected from other side
- cumulative ACK

Q: how receiver handles out-of-order segments

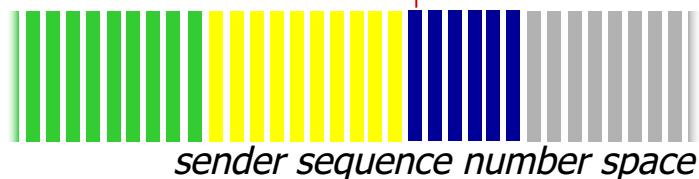
- A: TCP spec doesn’t say,
- up to implementor

outgoing segment from sender

source port #	dest port #
sequence number	
acknowledgement number	
	rwnd
checksum	urg pointer

window size

N



sent
ACKed

sent, not-
yet ACKed
("in-
flight")

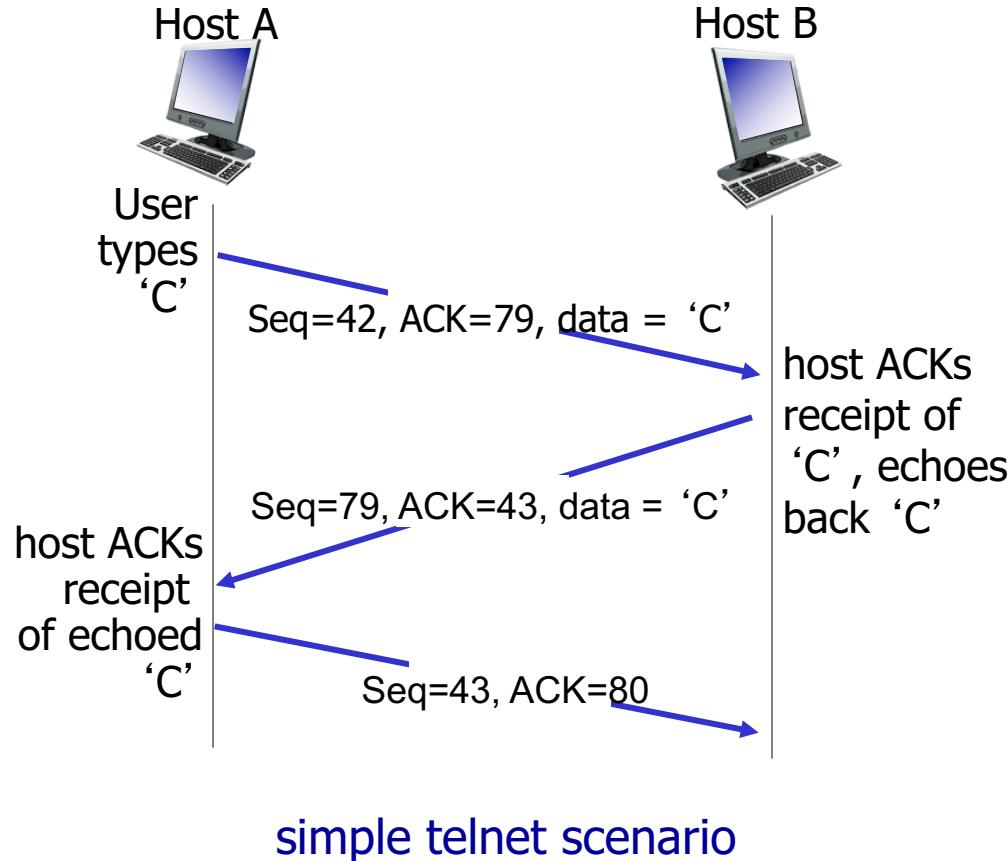
usable
but not
yet sent

not
usable

incoming segment to sender

source port #	dest port #
sequence number	
acknowledgement number	
	A
checksum	urg pointer

TCP seq. numbers, ACKs



TCP round trip time, timeout

Q: how to set TCP timeout value?

- longer than RTT
 - but RTT varies
- too short: premature timeout, unnecessary retransmissions
- too long: slow reaction to segment loss

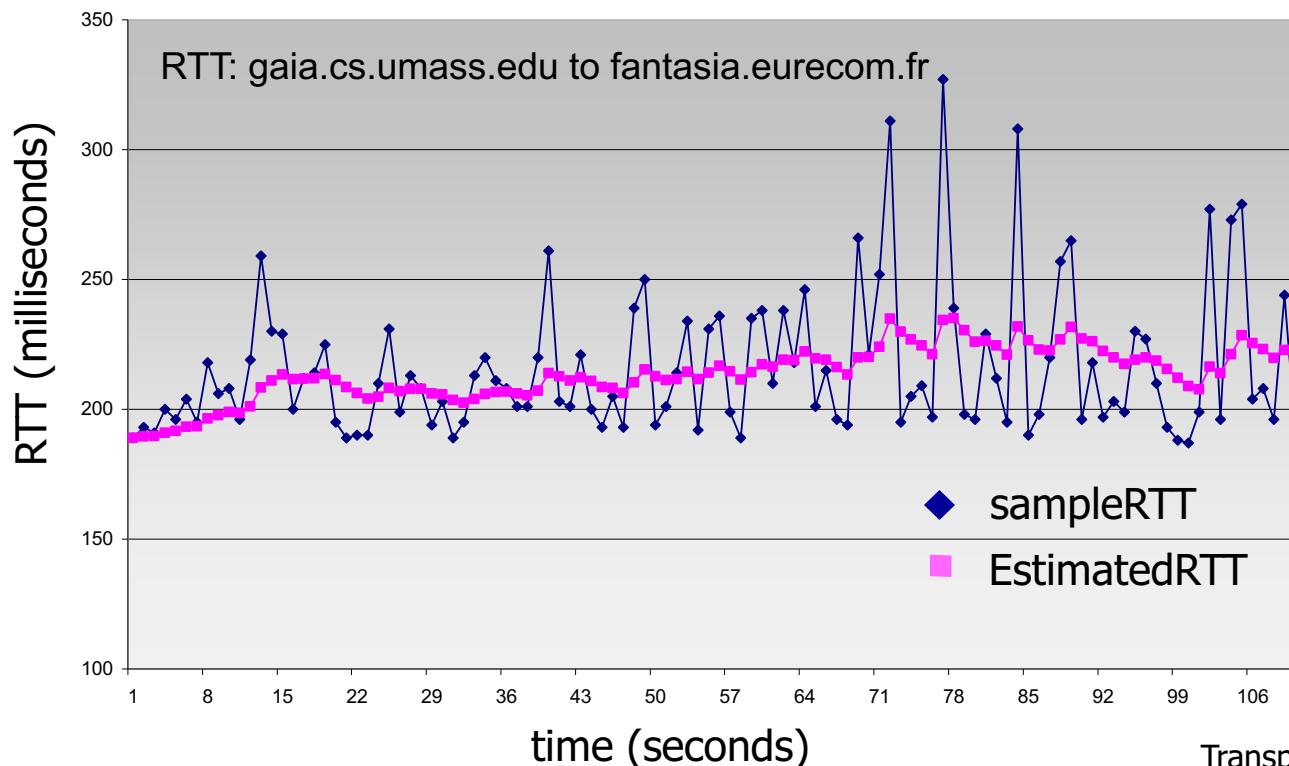
Q: how to estimate RTT?

- **SampleRTT**: measured time from segment transmission until ACK receipt
 - ignore retransmissions
- **SampleRTT** will vary, want estimated RTT “smoother”
 - average several *recent* measurements, not just current **SampleRTT**

TCP round trip time, timeout

$$\text{EstimatedRTT} = (1 - \alpha) * \text{EstimatedRTT} + \alpha * \text{SampleRTT}$$

- exponential weighted moving average
- influence of past sample decreases exponentially fast
- typical value: $\alpha = 0.125$



TCP round trip time, timeout

- **timeout interval:** **EstimatedRTT** plus “safety margin”
 - large variation in **EstimatedRTT** -> larger safety margin
- estimate SampleRTT deviation from EstimatedRTT:

$$\text{DevRTT} = (1-\beta) * \text{DevRTT} + \beta * |\text{SampleRTT} - \text{EstimatedRTT}|$$

(typically, $\beta = 0.25$)

$$\text{TimeoutInterval} = \text{EstimatedRTT} + 4 * \text{DevRTT}$$



↑
estimated RTT

↑
“safety margin”

* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

Chapter 3 outline

3.1 transport-layer services

3.2 multiplexing and demultiplexing

3.3 connectionless transport: UDP

3.4 principles of reliable data transfer

3.5 connection-oriented transport: TCP

- segment structure
- **reliable data transfer**
- flow control
- connection management

3.6 principles of congestion control

3.7 TCP congestion control

TCP reliable data transfer

- TCP creates rdt service on top of IP's unreliable service
 - pipelined segments
 - cumulative acks
 - single retransmission timer
- retransmissions triggered by:
 - timeout events
 - duplicate acks

let's initially consider simplified TCP sender:

- ignore duplicate acks
- ignore flow control, congestion control

TCP sender events:

data rcvd from app:

- create segment with seq #
- seq # is byte-stream number of first data byte in segment
- start timer if not already running
 - think of timer as for oldest unacked segment
 - expiration interval: `TimeOutInterval`

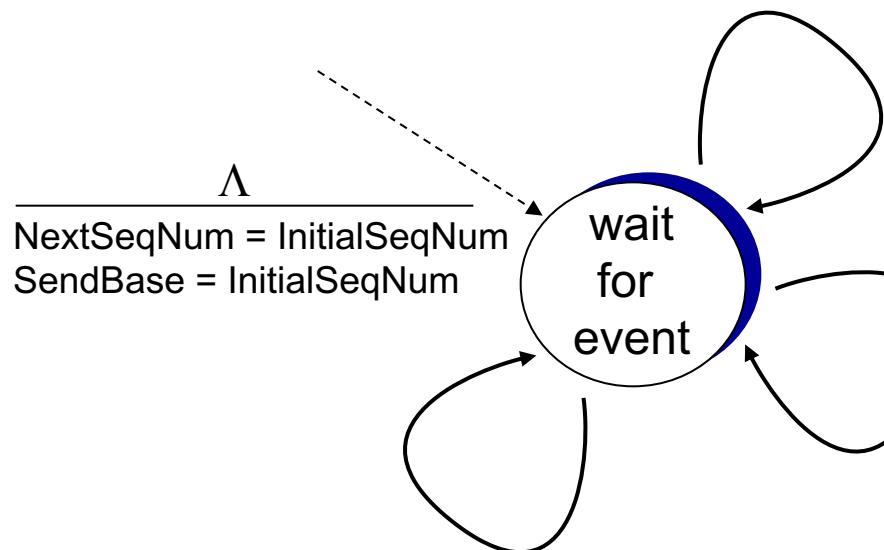
timeout:

- retransmit segment that caused timeout
- restart timer

ack rcvd:

- if ack acknowledges previously unacked segments
 - update what is known to be ACKed
 - start timer if there are still unacked segments

TCP sender (simplified)

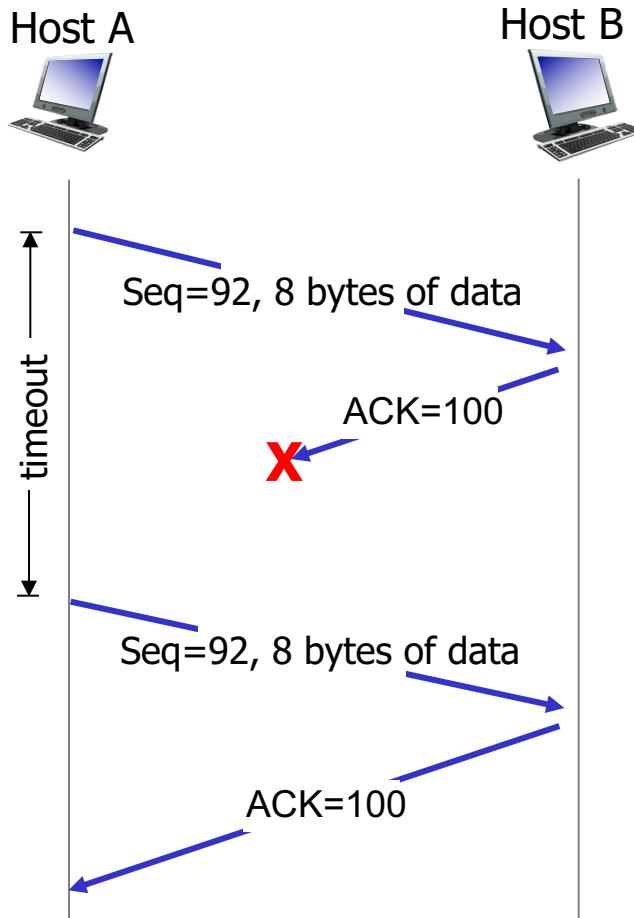


data received from application above
create segment, seq. #: NextSeqNum
pass segment to IP (i.e., “send”)
 $\text{NextSeqNum} = \text{NextSeqNum} + \text{length}(\text{data})$
if (timer currently not running)
start timer

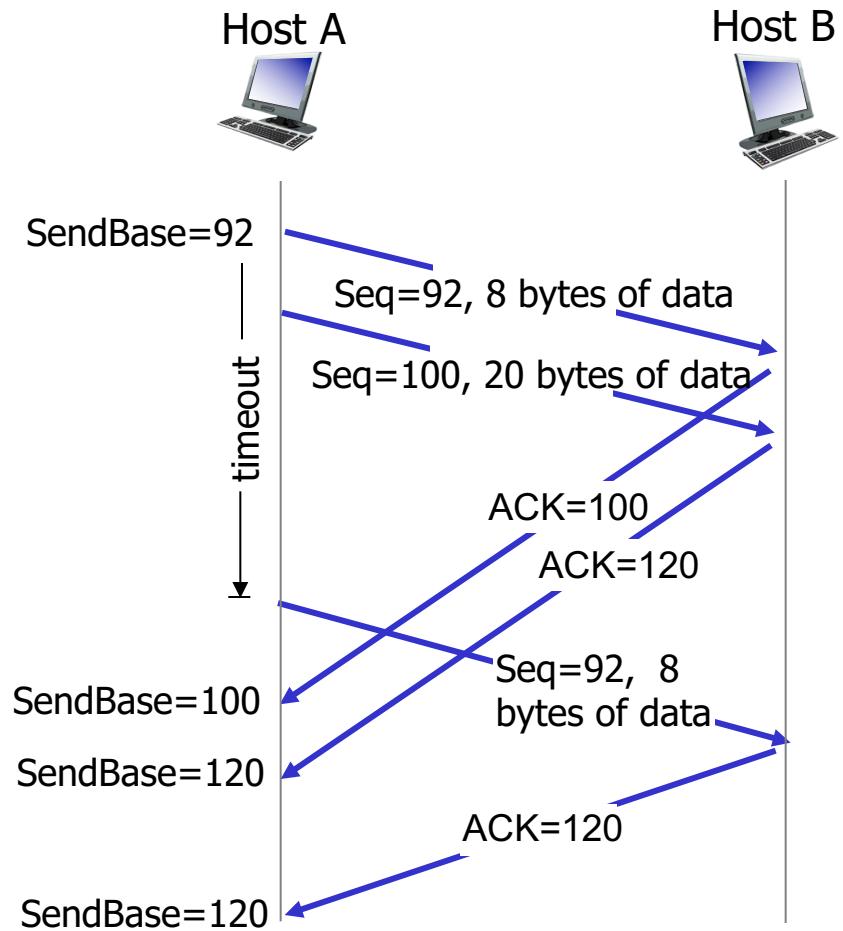
timeout
retransmit not-yet-acked segment
with smallest seq. #
start timer

```
if (y > SendBase) {
    SendBase = y
    /* SendBase-1: last cumulatively ACKed byte */
    if (there are currently not-yet-acked segments)
        start timer
    else stop timer
}
```

TCP: retransmission scenarios

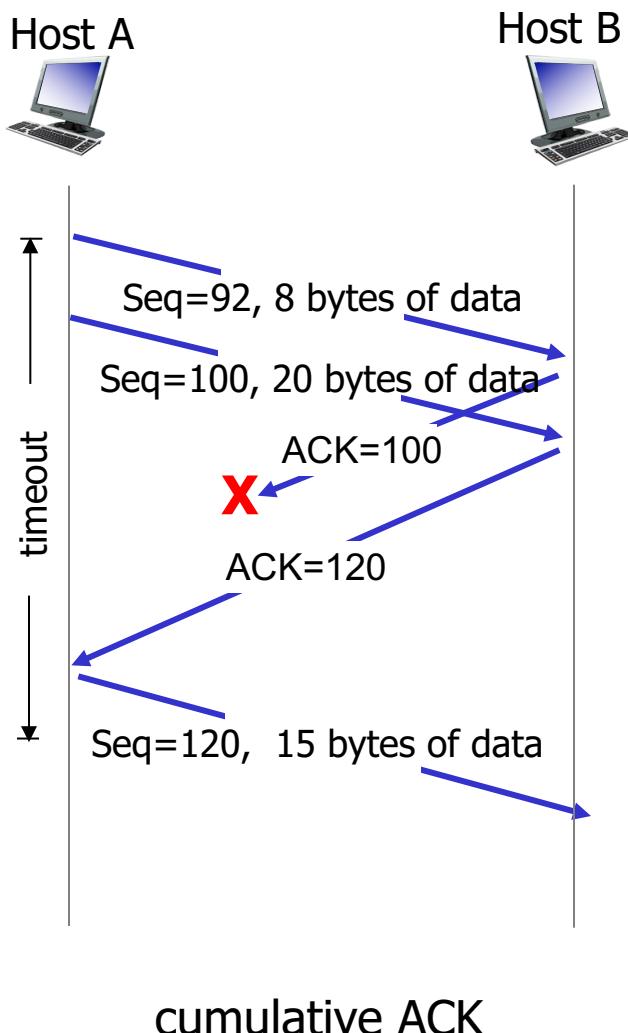


lost ACK scenario



premature timeout

TCP: retransmission scenarios



TCP ACK generation [RFC 1122, RFC 2581]

<i>event at receiver</i>	<i>TCP receiver action</i>
arrival of in-order segment with expected seq #. All data up to expected seq # already ACKed	delayed ACK. Wait up to 500ms for next segment. If no next segment, send ACK
arrival of in-order segment with expected seq #. One other segment has ACK pending	immediately send single cumulative ACK, ACKing both in-order segments
arrival of out-of-order segment higher-than-expect seq. # . Gap detected	immediately send <i>duplicate ACK</i> , indicating seq. # of next expected byte
arrival of segment that partially or completely fills gap	immediate send ACK, provided that segment starts at lower end of gap

TCP fast retransmit

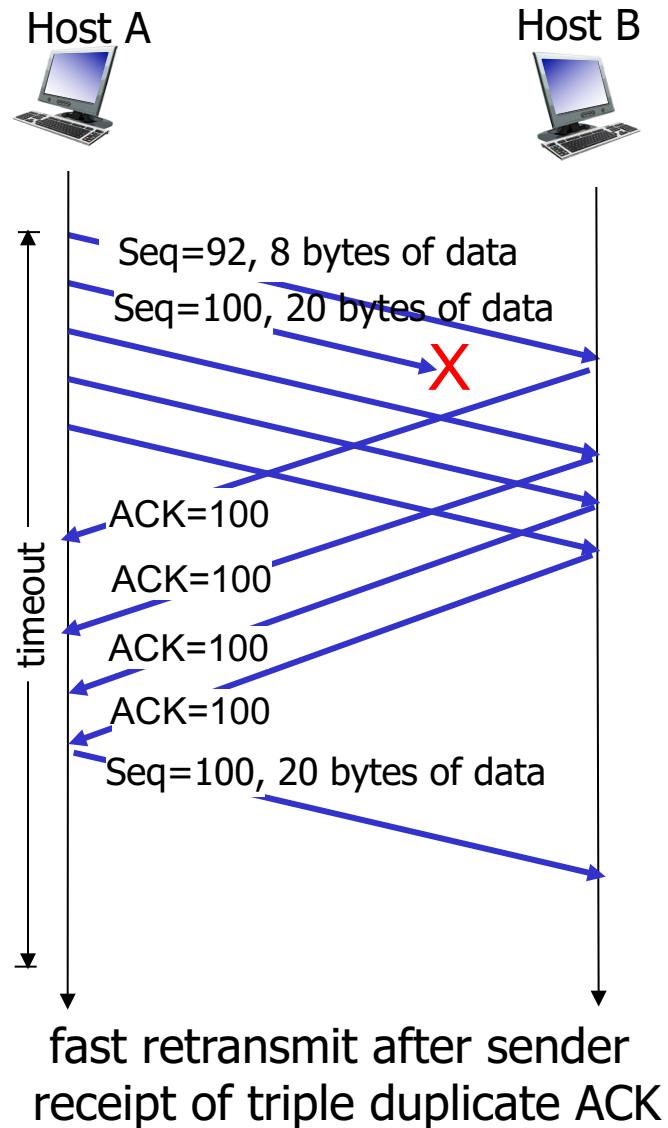
- time-out period often relatively long:
 - long delay before resending lost packet
- detect lost segments via duplicate ACKs.
 - sender often sends many segments back-to-back
 - if segment is lost, there will likely be many duplicate ACKs.

TCP fast retransmit

if sender receives 3 ACKs for same data (“triple duplicate ACKs”), resend unacked segment with smallest seq #

- likely that unacked segment lost, so don’t wait for timeout

TCP fast retransmit



Chapter 3 outline

3.1 transport-layer services

3.2 multiplexing and demultiplexing

3.3 connectionless transport: UDP

3.4 principles of reliable data transfer

3.5 connection-oriented transport: TCP

- segment structure
- reliable data transfer
- **flow control**
- connection management

3.6 principles of congestion control

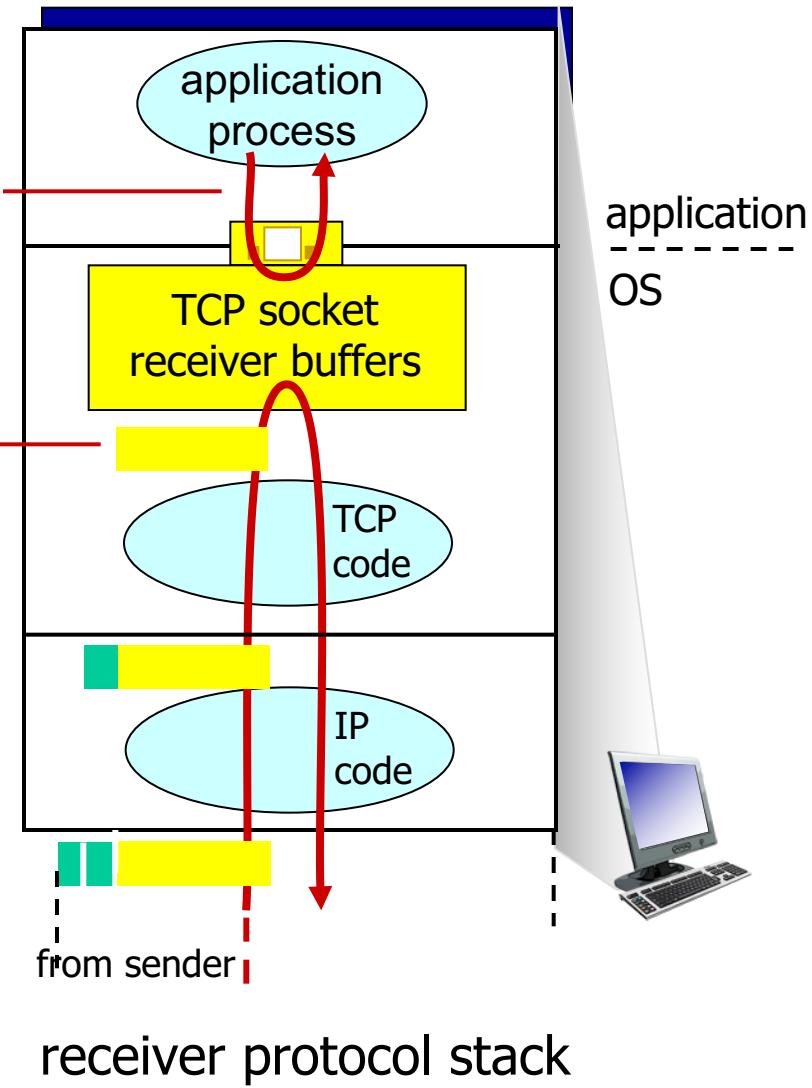
3.7 TCP congestion control

TCP flow control

application may
remove data from
TCP socket buffers

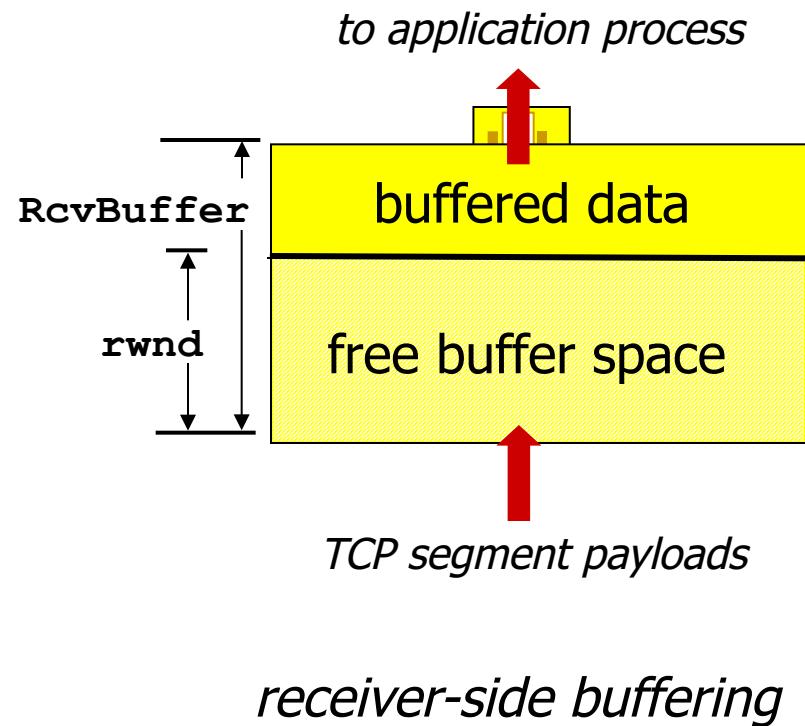
... slower than TCP
receiver is delivering
(sender is sending)

flow control
receiver controls sender, so
sender won't overflow
receiver's buffer by transmitting
too much, too fast



TCP flow control

- receiver “advertises” free buffer space by including **rwnd** value in TCP header of receiver-to-sender segments
 - **RcvBuffer** size set via socket options (typical default is 4096 bytes)
 - many operating systems autoadjust **RcvBuffer**
- sender limits amount of unacked (“in-flight”) data to receiver’s **rwnd** value
- guarantees receive buffer will not overflow



receiver-side buffering

Chapter 3 outline

3.1 transport-layer services

3.2 multiplexing and demultiplexing

3.3 connectionless transport: UDP

3.4 principles of reliable data transfer

3.5 connection-oriented transport: TCP

- segment structure
- reliable data transfer
- flow control
- connection management

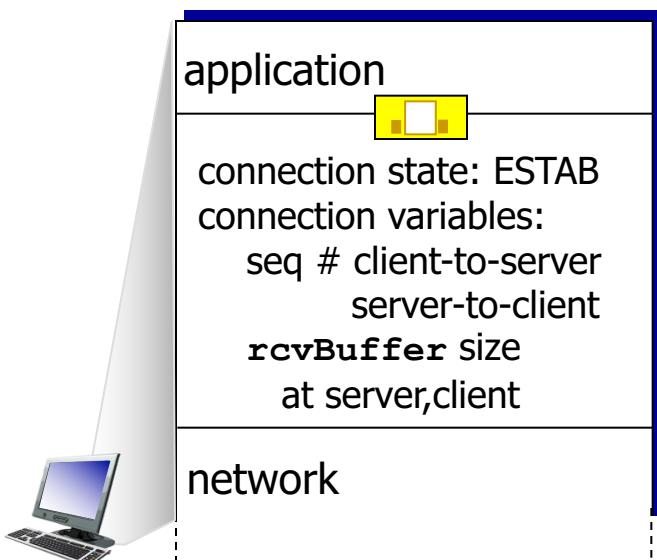
3.6 principles of congestion control

3.7 TCP congestion control

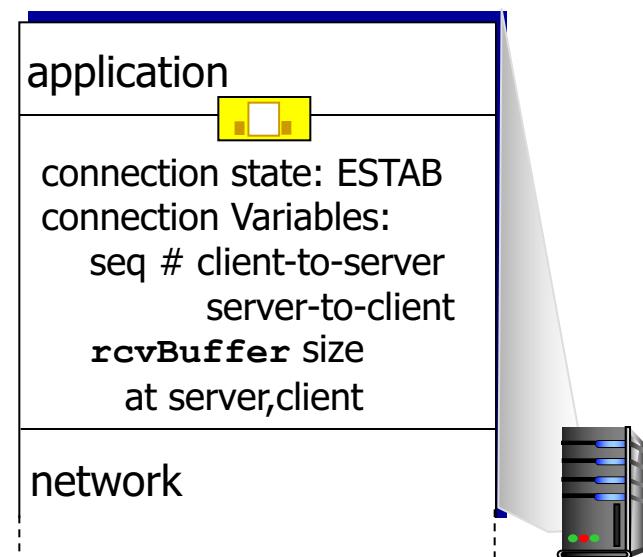
Connection Management

before exchanging data, sender/receiver “handshake”:

- agree to establish connection (each knowing the other willing to establish connection)
- agree on connection parameters



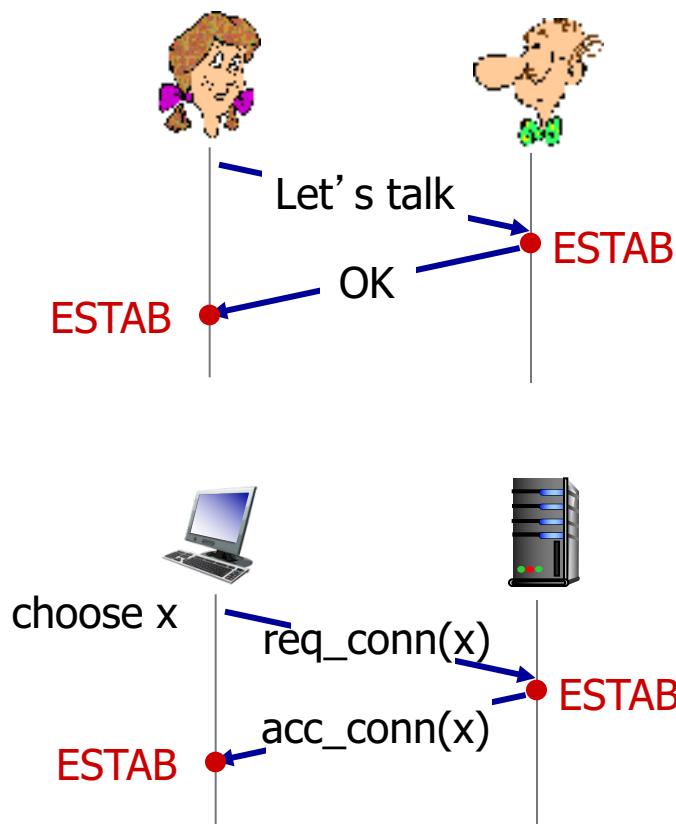
```
Socket clientSocket =  
    newSocket("hostname", "port  
    number");
```



```
Socket connectionSocket =  
    welcomeSocket.accept();
```

Agreeing to establish a connection

2-way handshake:

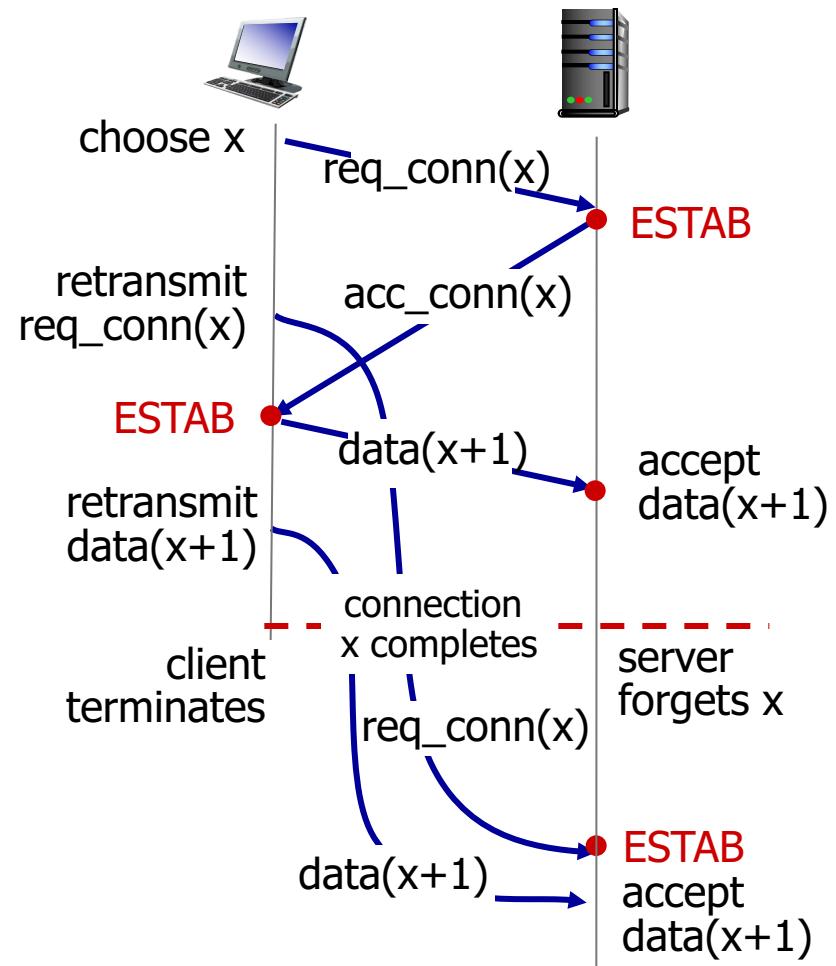
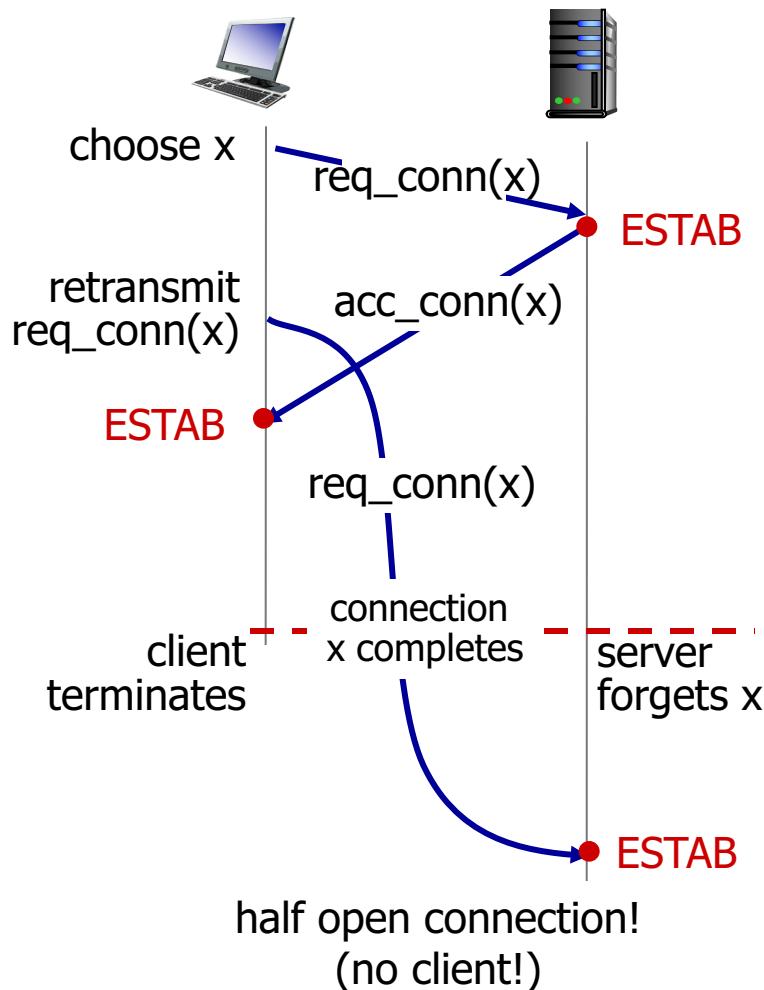


Q: will 2-way handshake always work in network?

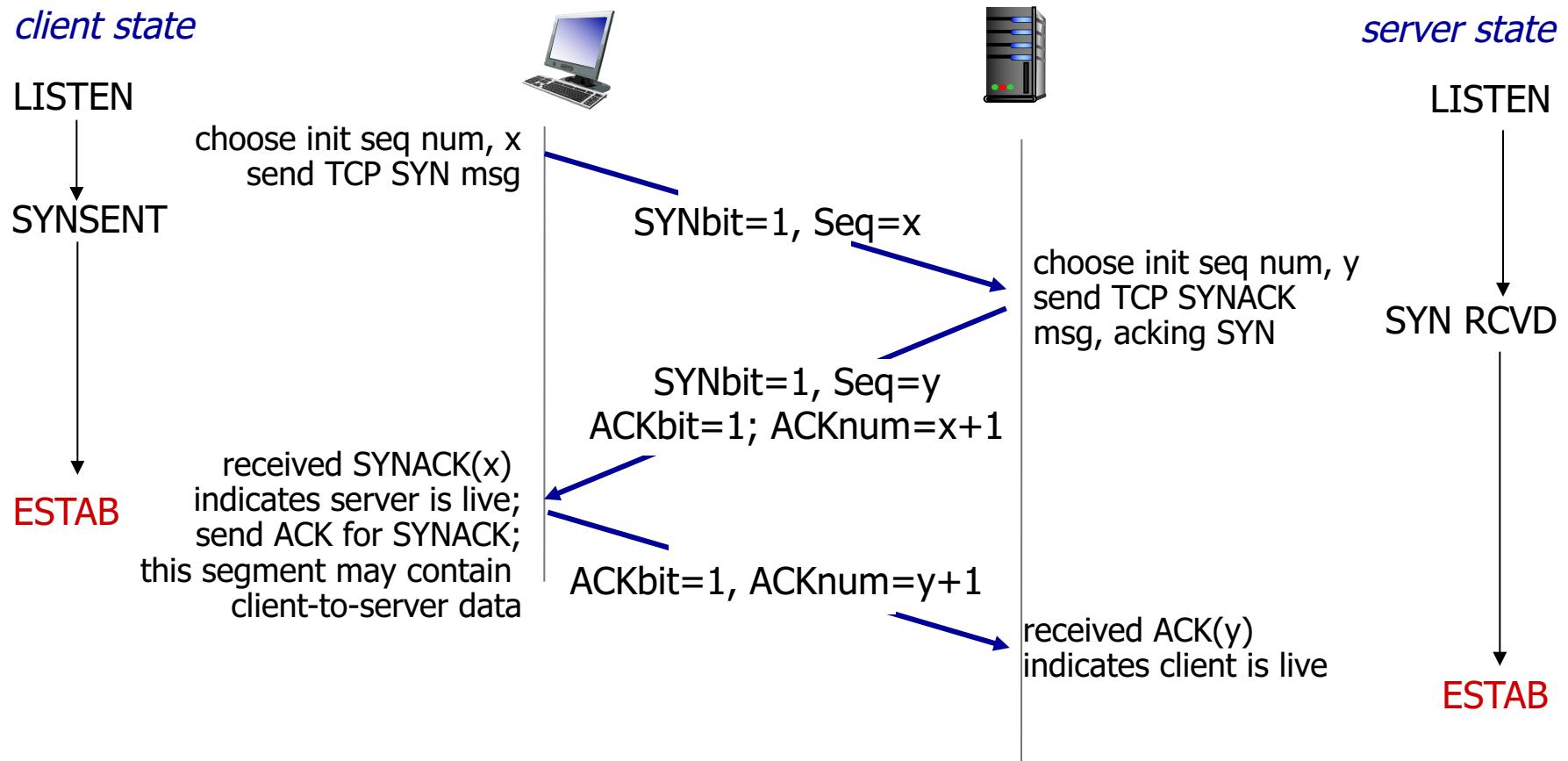
- variable delays
- retransmitted messages (e.g. `req_conn(x)`) due to message loss
- message reordering
- can't “see” other side

Agreeing to establish a connection

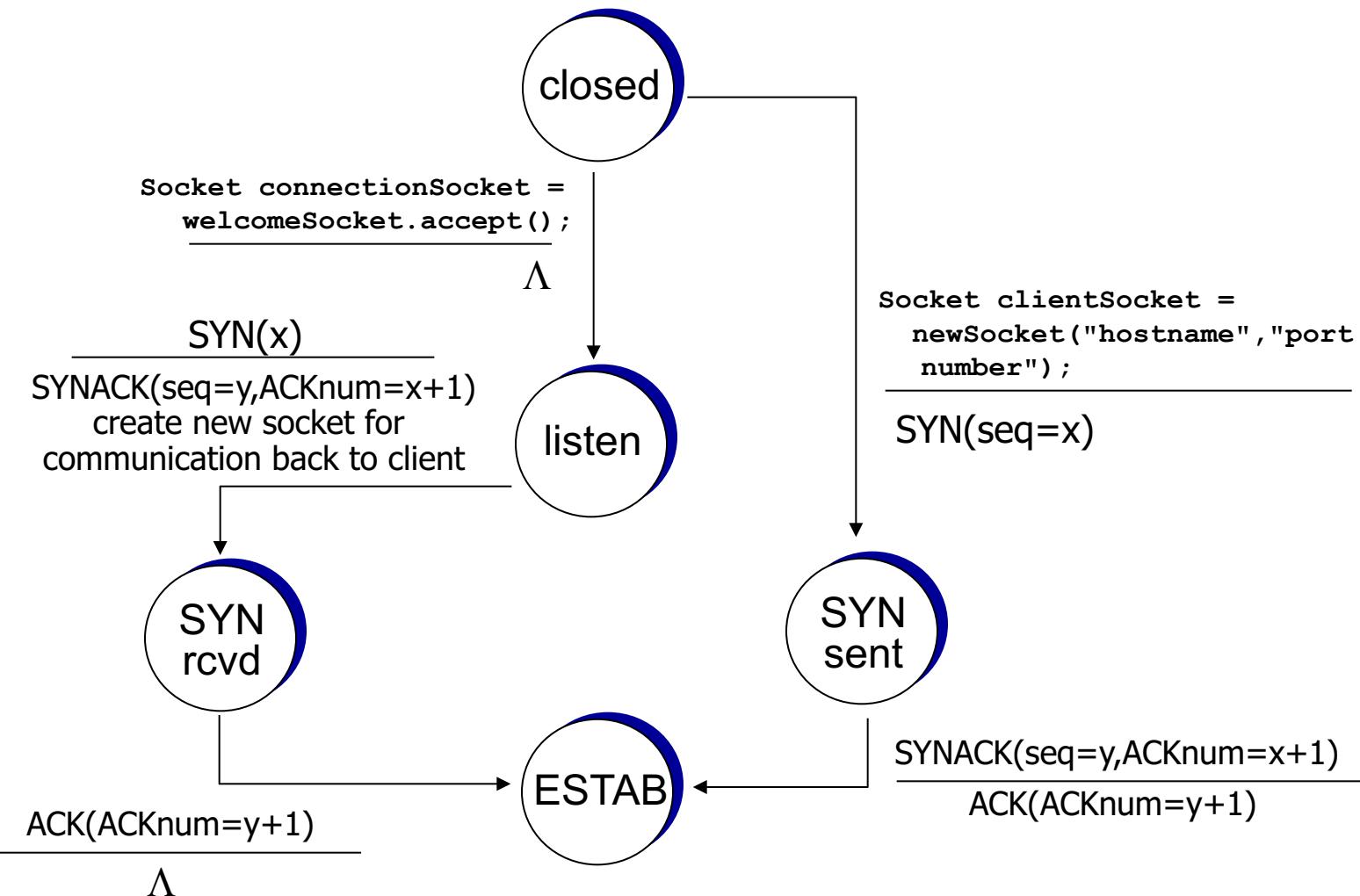
2-way handshake failure scenarios:



TCP 3-way handshake



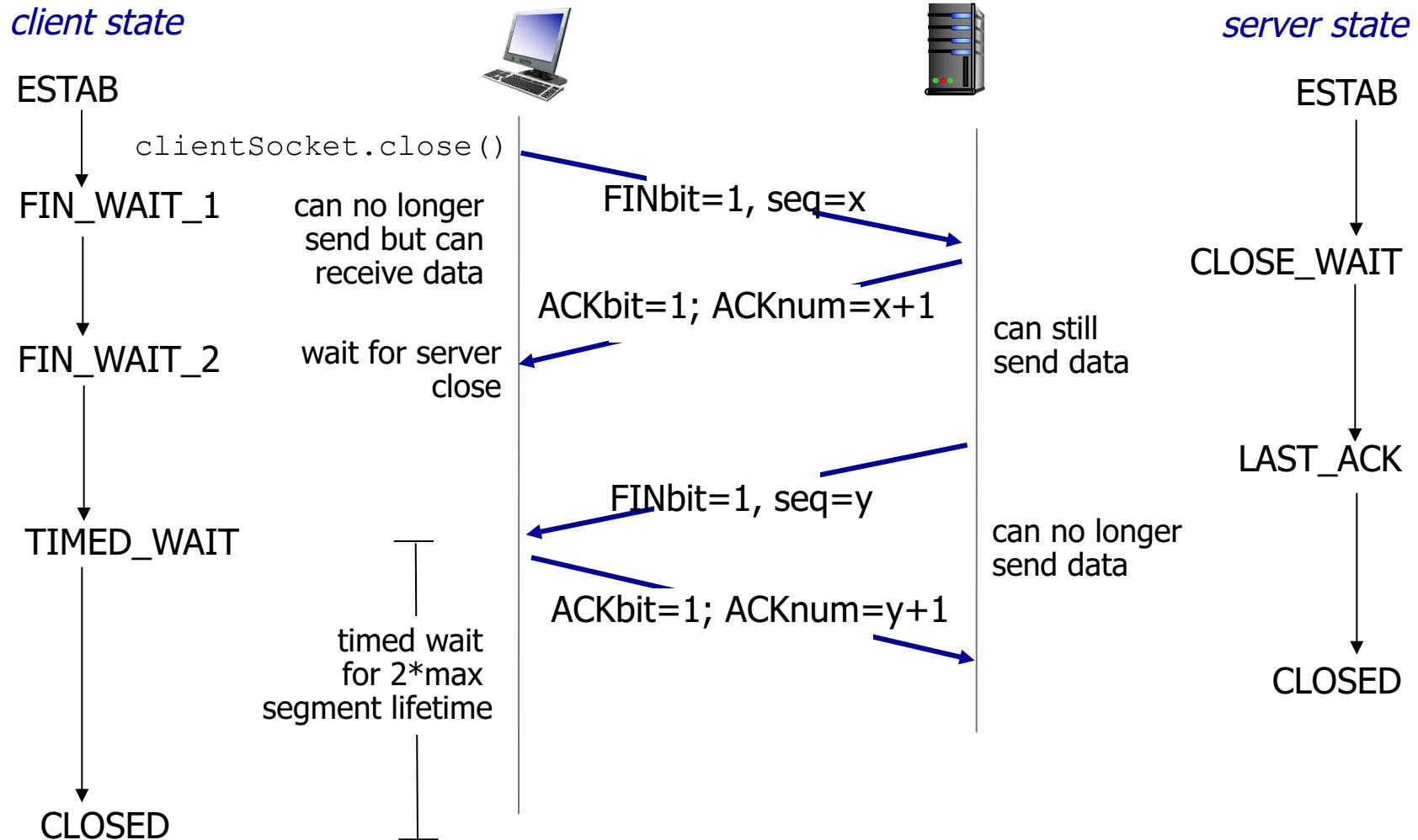
TCP 3-way handshake: FSM



TCP: closing a connection

- client, server each close their side of connection
 - send TCP segment with FIN bit = 1
- respond to received FIN with ACK
 - on receiving FIN, ACK can be combined with own FIN
- simultaneous FIN exchanges can be handled

TCP: closing a connection



Chapter 3 outline

3.1 transport-layer services

3.2 multiplexing and demultiplexing

3.3 connectionless transport: UDP

3.4 principles of reliable data transfer

3.5 connection-oriented transport: TCP

- segment structure
- reliable data transfer
- flow control
- connection management

3.6 principles of congestion control

3.7 TCP congestion control

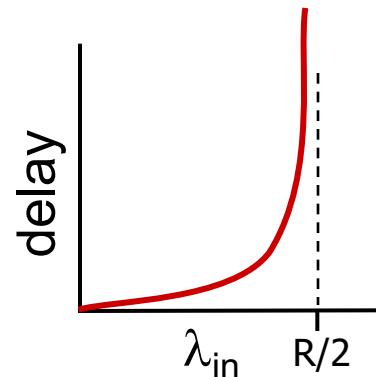
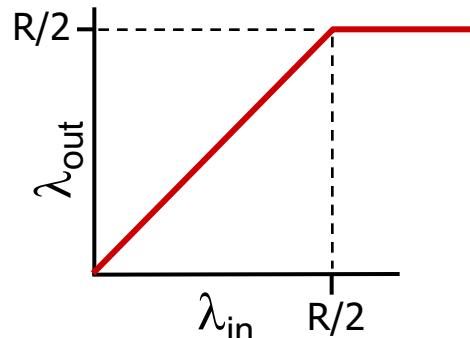
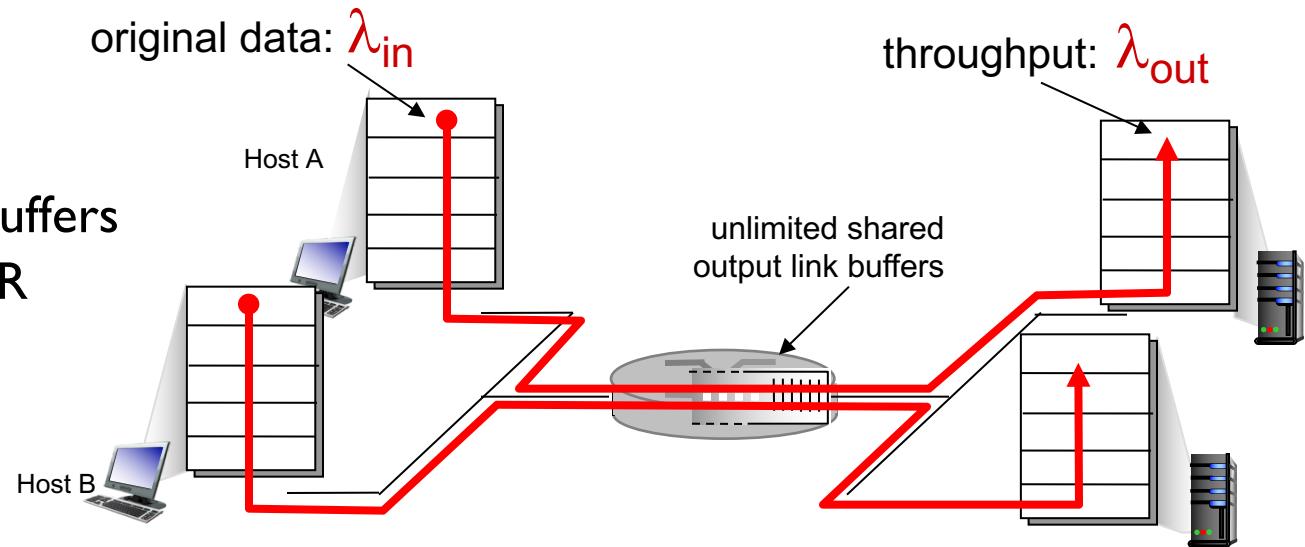
Principles of congestion control

congestion:

- informally: “too many sources sending too much data too fast for *network* to handle”
- different from flow control!
- manifestations:
 - lost packets (buffer overflow at routers)
 - long delays (queueing in router buffers)
- a top-10 problem!

Causes/costs of congestion: scenario I

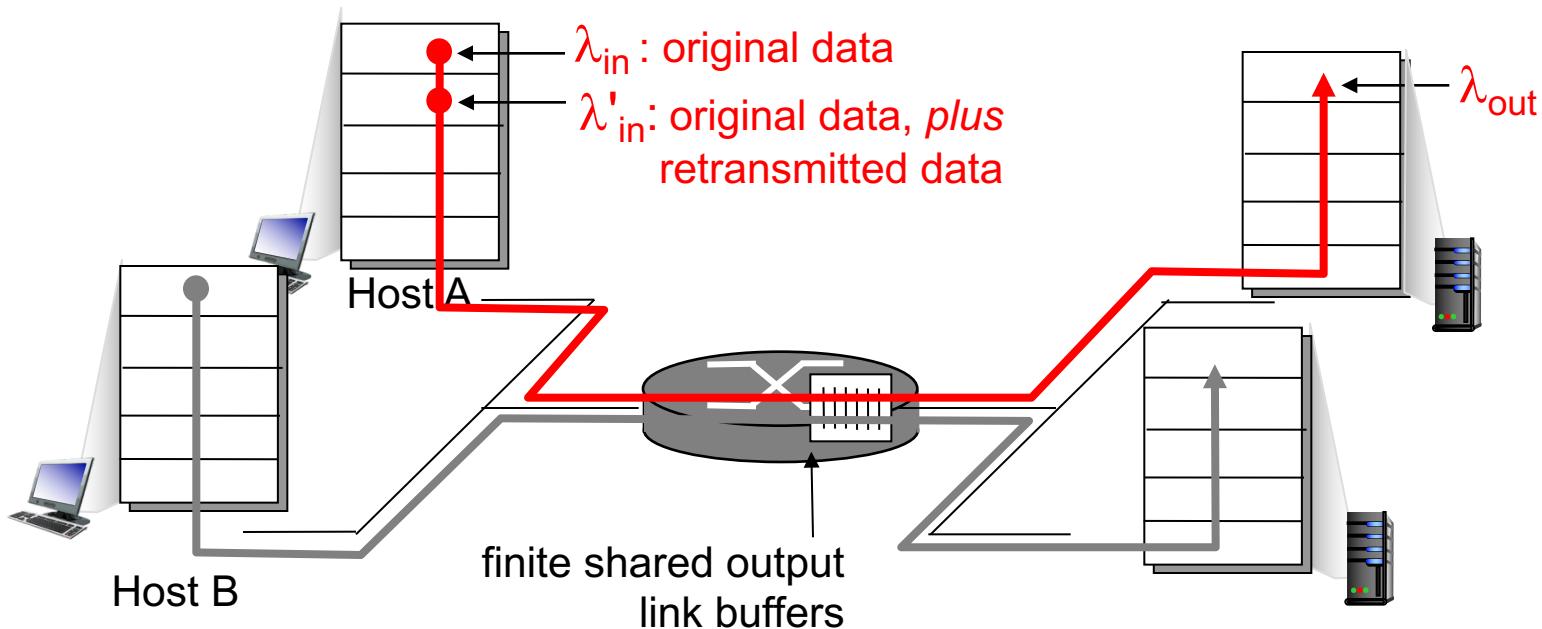
- two senders, two receivers
- one router, infinite buffers
- output link capacity: R
- no retransmission



- maximum per-connection throughput: $R/2$
- ❖ large delays as arrival rate, λ_{in} , approaches capacity

Causes/costs of congestion: scenario 2

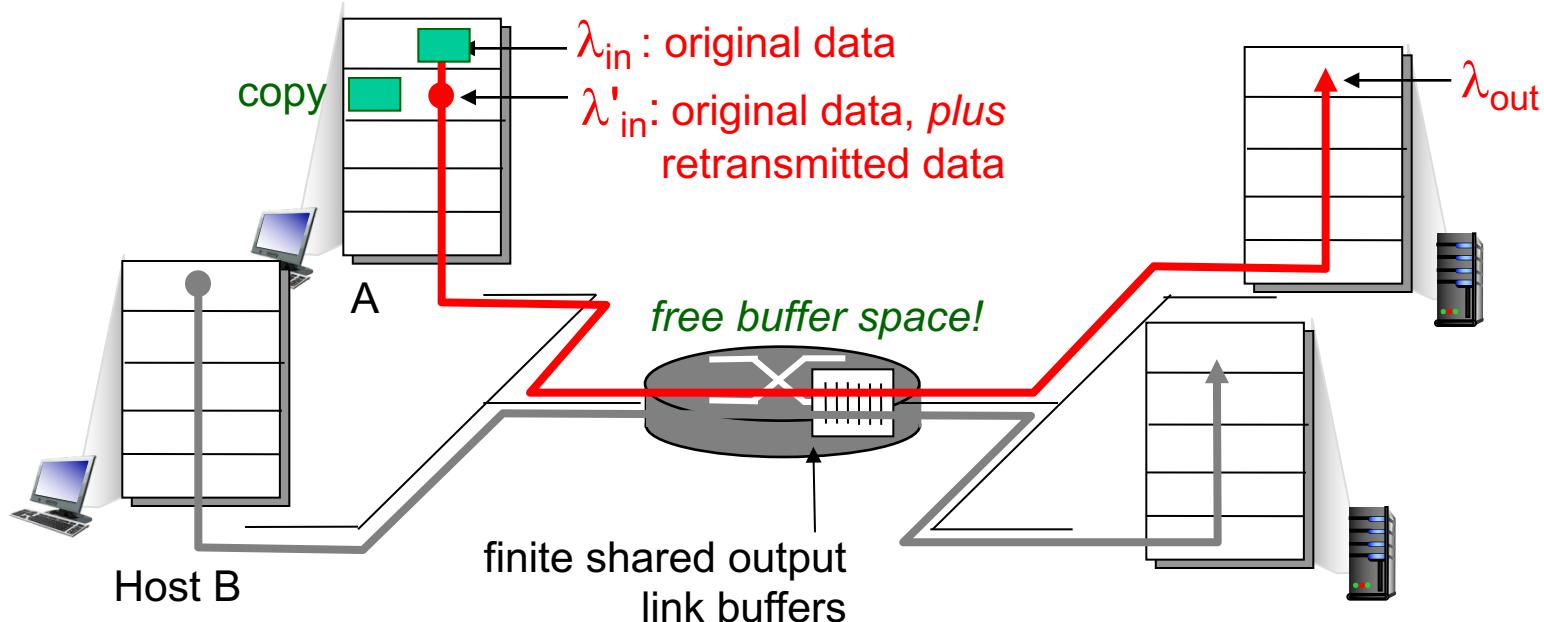
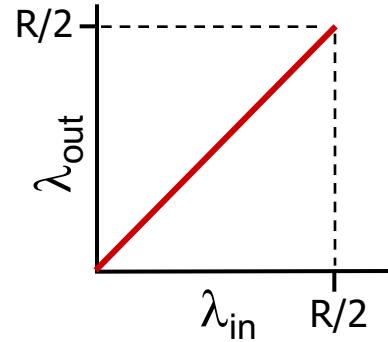
- one router, *finite* buffers
- sender retransmission of timed-out packet
 - application-layer input = application-layer output: $\lambda_{in} = \lambda_{out}$
 - transport-layer input includes *retransmissions* : $\lambda'_{in} \geq \lambda_{in}$



Causes/costs of congestion: scenario 2

idealization: perfect knowledge

- sender sends only when router buffers available

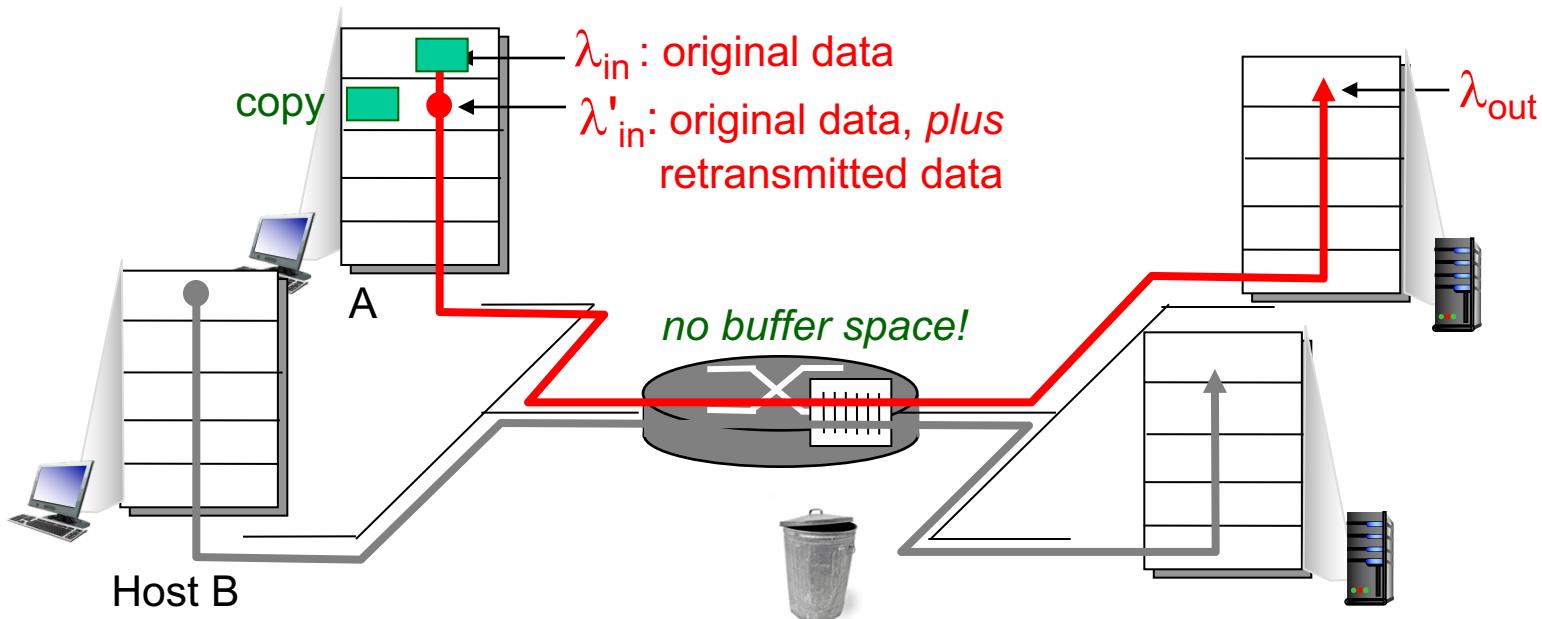


Causes/costs of congestion: scenario 2

Idealization: *known loss*

packets can be lost,
dropped at router due
to full buffers

- sender only resends if
packet *known* to be lost

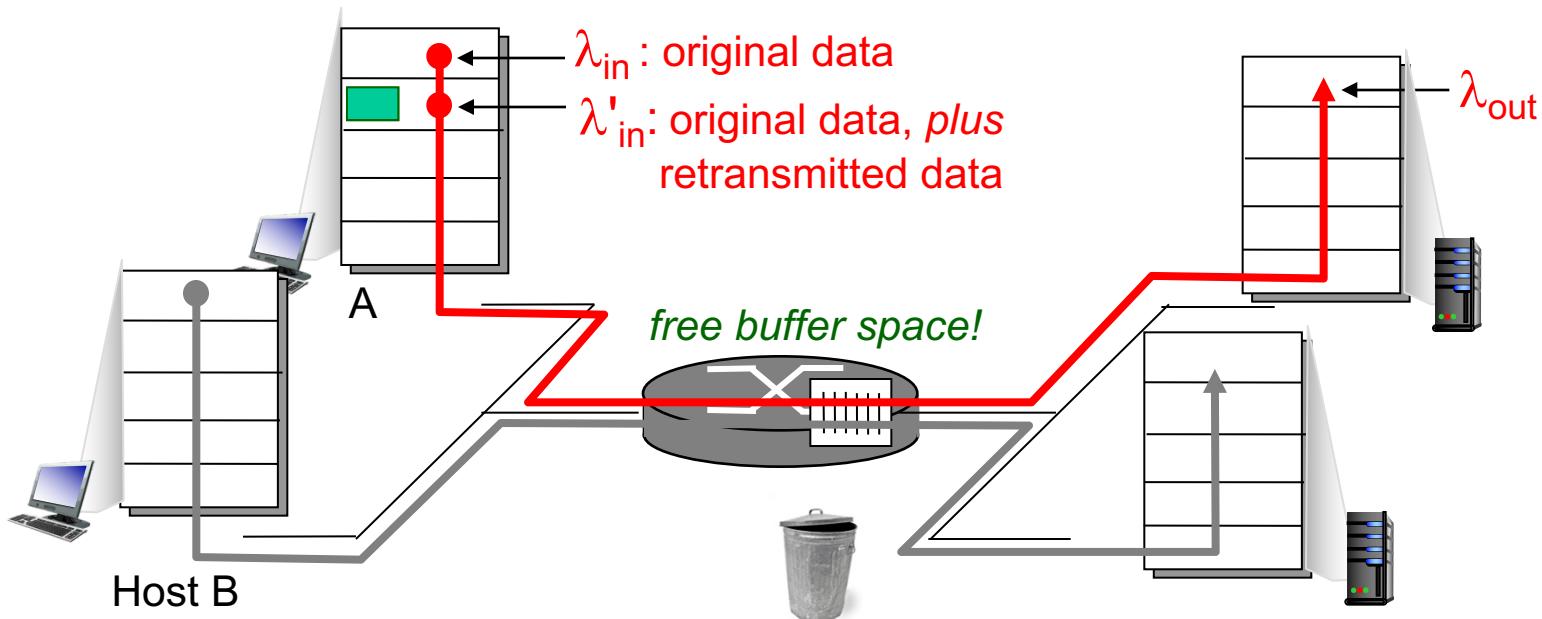
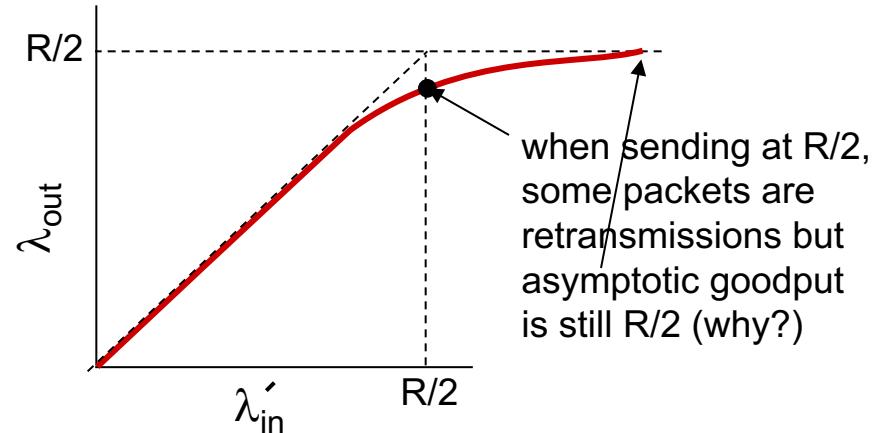


Causes/costs of congestion: scenario 2

Idealization: *known loss*

packets can be lost,
dropped at router due
to full buffers

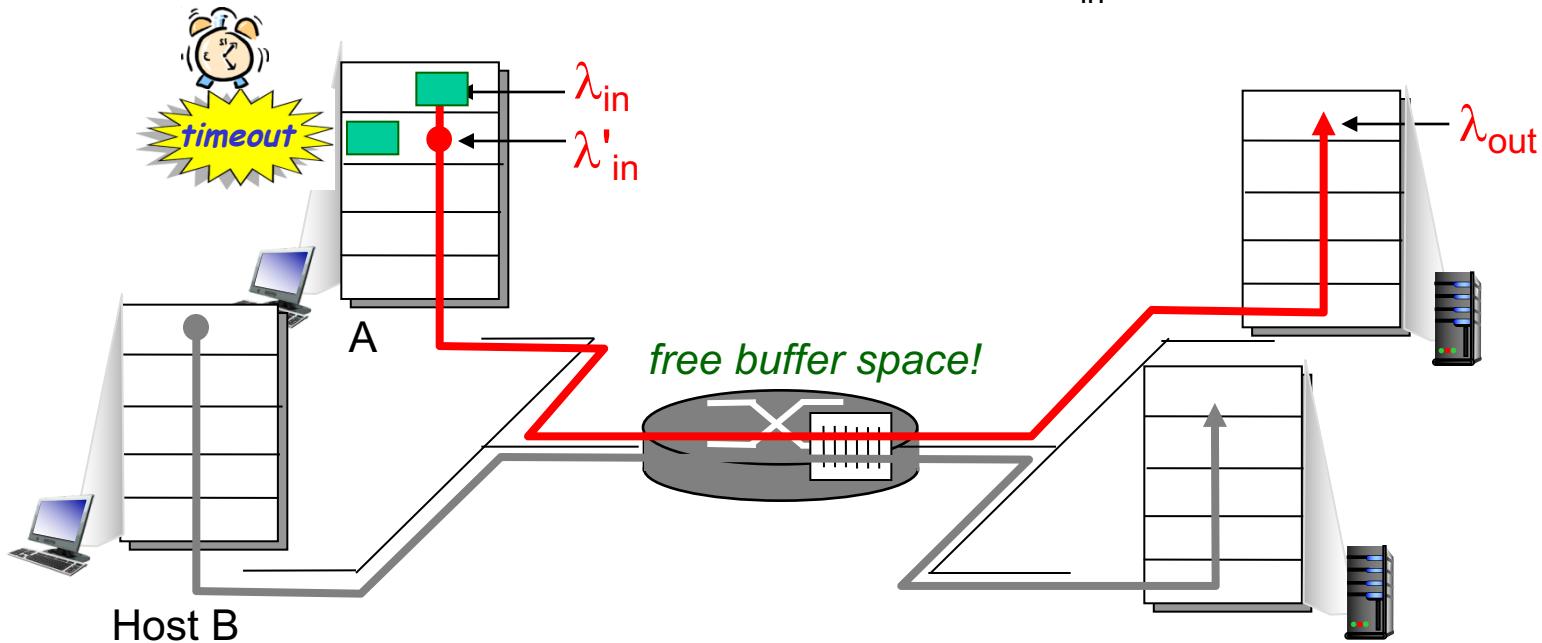
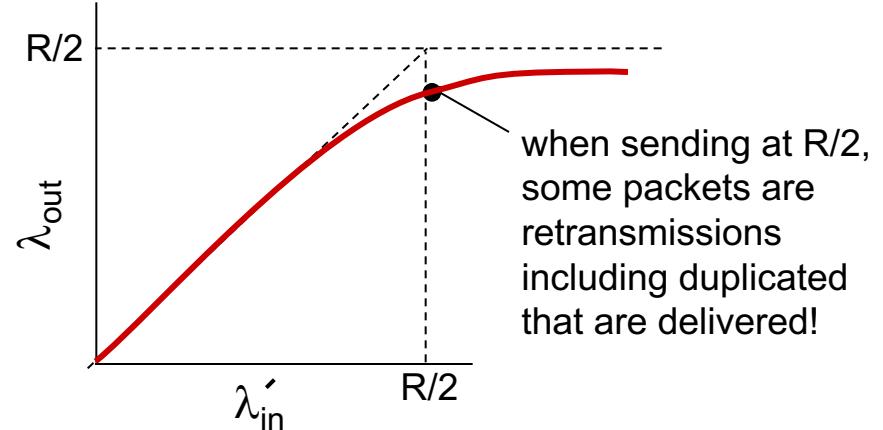
- sender only resends if
packet *known* to be lost



Causes/costs of congestion: scenario 2

Realistic: *duplicates*

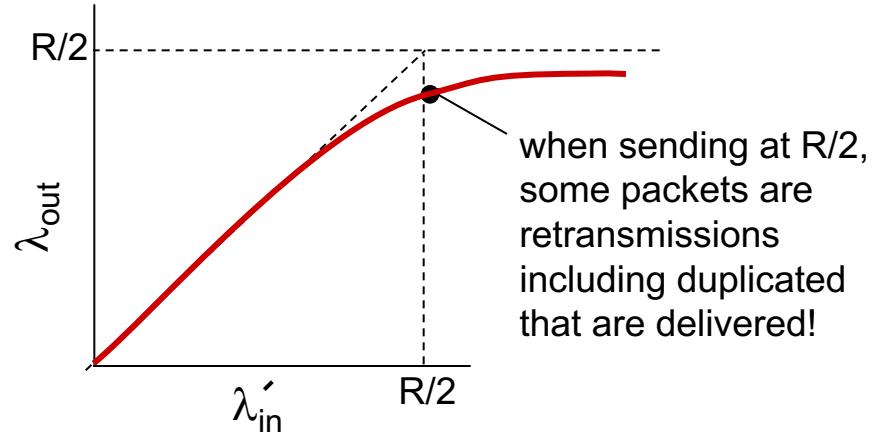
- packets can be lost, dropped at router due to full buffers
- sender times out prematurely, sending *two* copies, both of which are delivered



Causes/costs of congestion: scenario 2

Realistic: *duplicates*

- packets can be lost, dropped at router due to full buffers
- sender times out prematurely, sending *two* copies, both of which are delivered



“costs” of congestion:

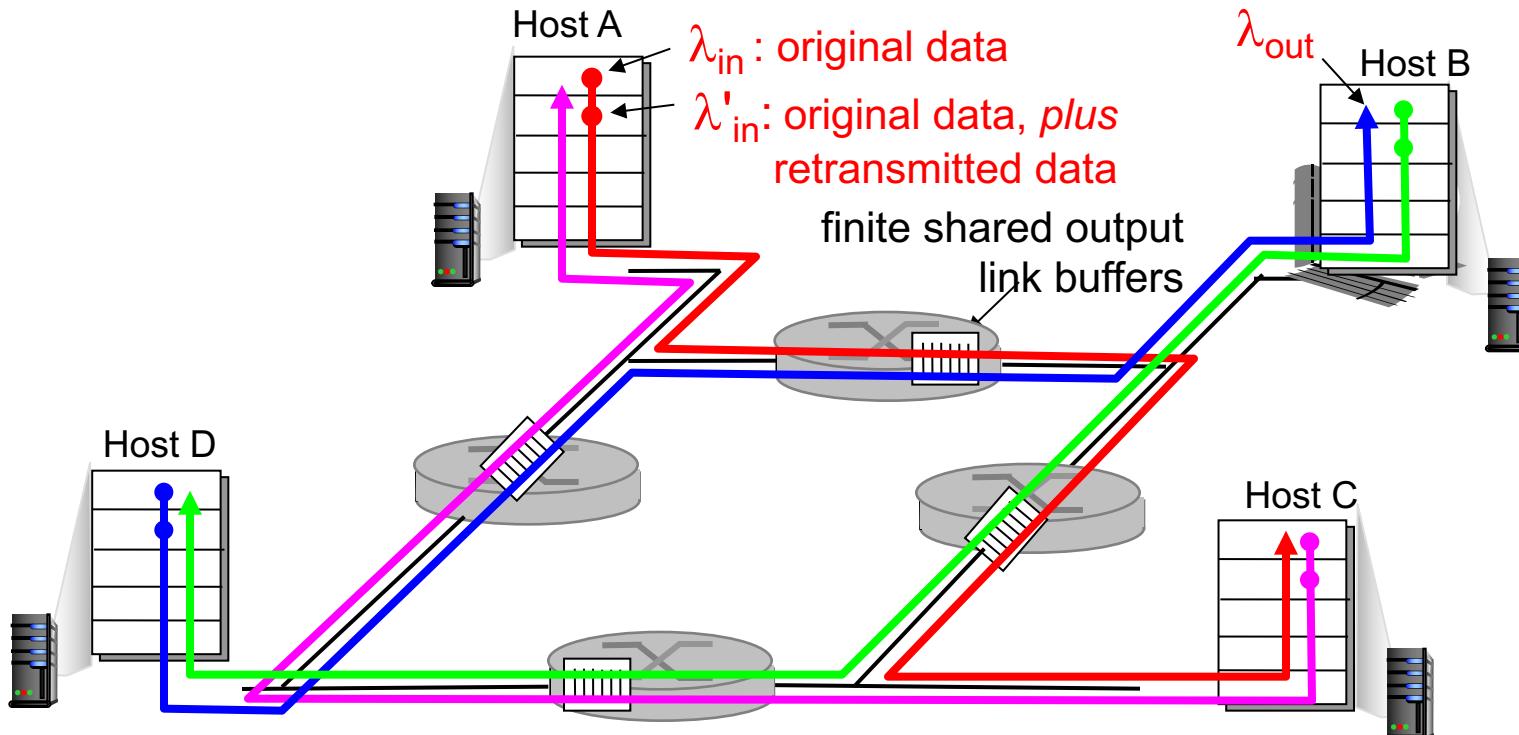
- more work (retrans) for given “goodput”
- unneeded retransmissions: link carries multiple copies of pkt
 - decreasing goodput

Causes/costs of congestion: scenario 3

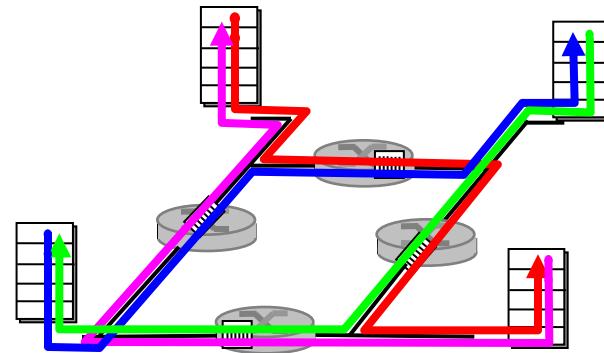
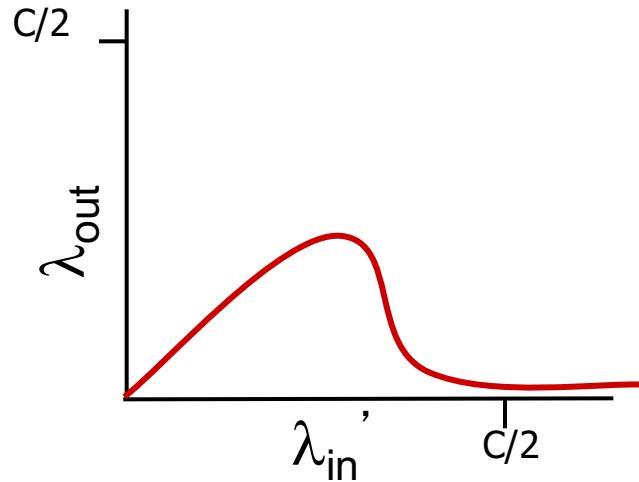
- four senders
- multihop paths
- timeout/retransmit

Q: what happens as λ_{in} and λ'_{in} increase ?

A: as red λ_{in} increases, all arriving blue pkts at upper queue are dropped, blue throughput $\rightarrow 0$



Causes/costs of congestion: scenario 3



another “cost” of congestion:

- when packet dropped, any “upstream transmission capacity used for that packet was wasted!

Chapter 3 outline

3.1 transport-layer services

3.2 multiplexing and demultiplexing

3.3 connectionless transport: UDP

3.4 principles of reliable data transfer

3.5 connection-oriented transport: TCP

- segment structure
- reliable data transfer
- flow control
- connection management

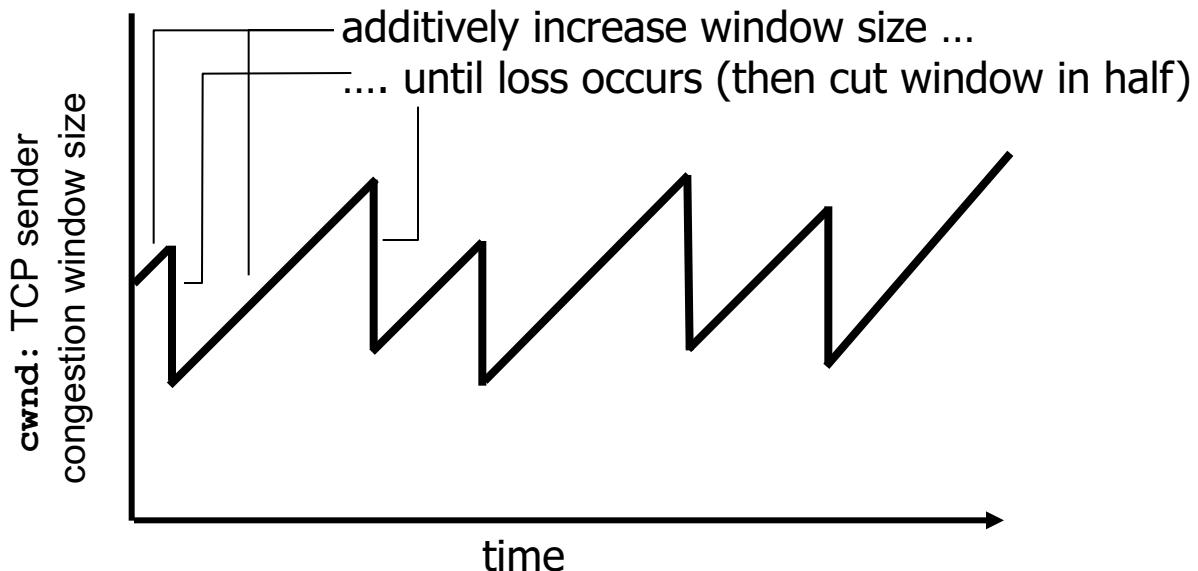
3.6 principles of congestion control

3.7 TCP congestion control

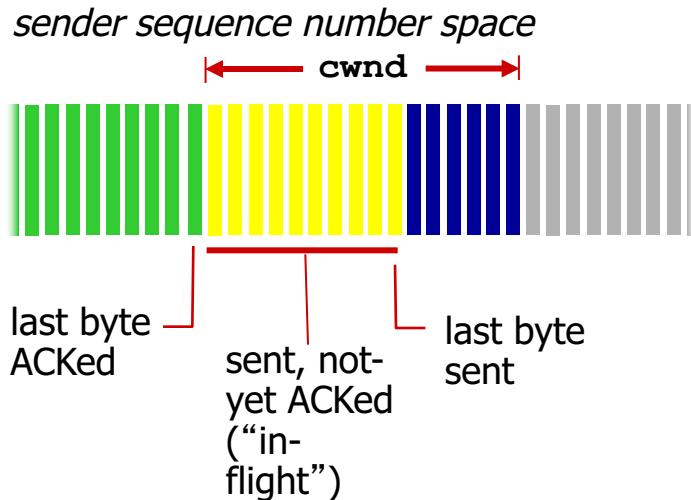
TCP congestion control: additive increase multiplicative decrease

- **approach:** sender increases transmission rate (window size), probing for usable bandwidth, until loss occurs
 - **additive increase:** increase **cwnd** by 1 MSS every RTT until loss detected
 - **multiplicative decrease:** cut **cwnd** in half after loss

AIMD saw tooth behavior: probing for bandwidth



TCP Congestion Control: details



- sender limits transmission:

$$\frac{\text{LastByteSent} - \text{LastByteAcked}}{\text{cwnd}} \leq 1$$

- **cwnd** is dynamic, function of perceived network congestion

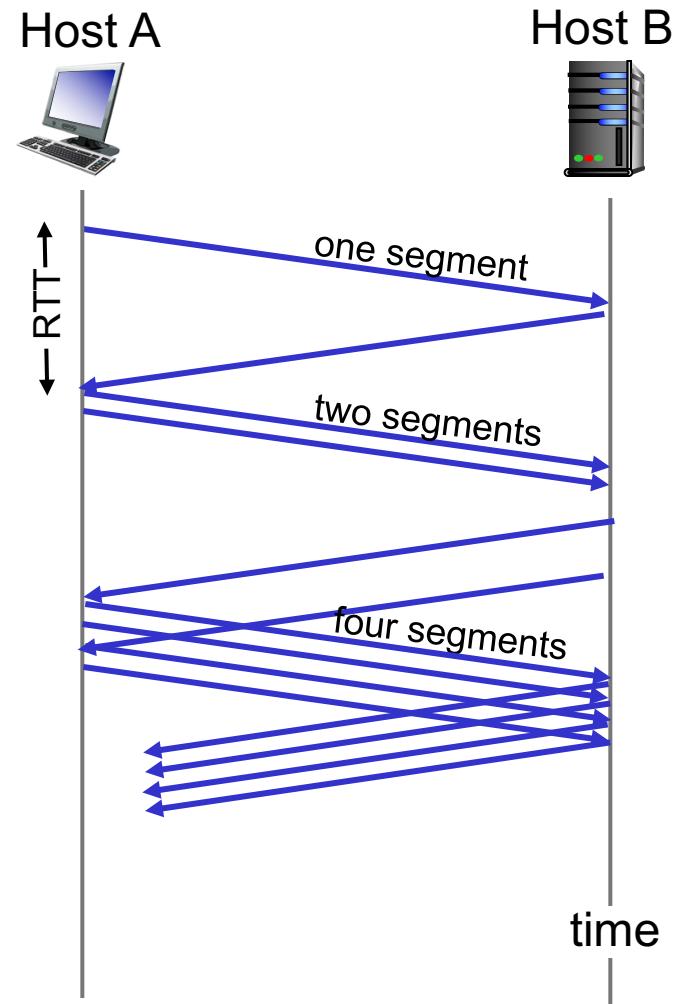
TCP sending rate:

- roughly: send cwnd bytes, wait RTT for ACKS, then send more bytes

$$\text{rate} \approx \frac{\text{cwnd}}{\text{RTT}} \text{ bytes/sec}$$

TCP Slow Start

- when connection begins, increase rate exponentially until first loss event:
 - initially **cwnd** = 1 MSS
 - double **cwnd** every RTT
 - done by incrementing **cwnd** for every ACK received
- summary: initial rate is slow but ramps up exponentially fast



TCP: detecting, reacting to loss

- loss indicated by timeout:
 - **cwnd** set to 1 MSS;
 - window then grows exponentially (as in slow start) to threshold, then grows linearly
- loss indicated by 3 duplicate ACKs: TCP RENO
 - dup ACKs indicate network capable of delivering some segments
 - **cwnd** is cut in half window then grows linearly
- TCP Tahoe always sets **cwnd** to 1 (timeout or 3 duplicate acks)

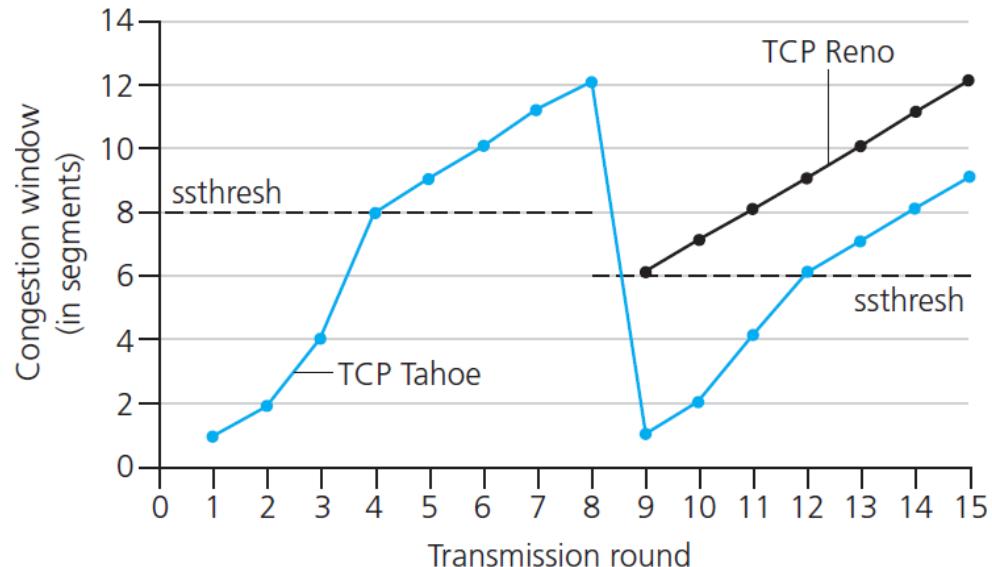
TCP: switching from slow start to CA

Q: when should the exponential increase switch to linear?

A: when **cwnd** gets to 1/2 of its value before timeout.

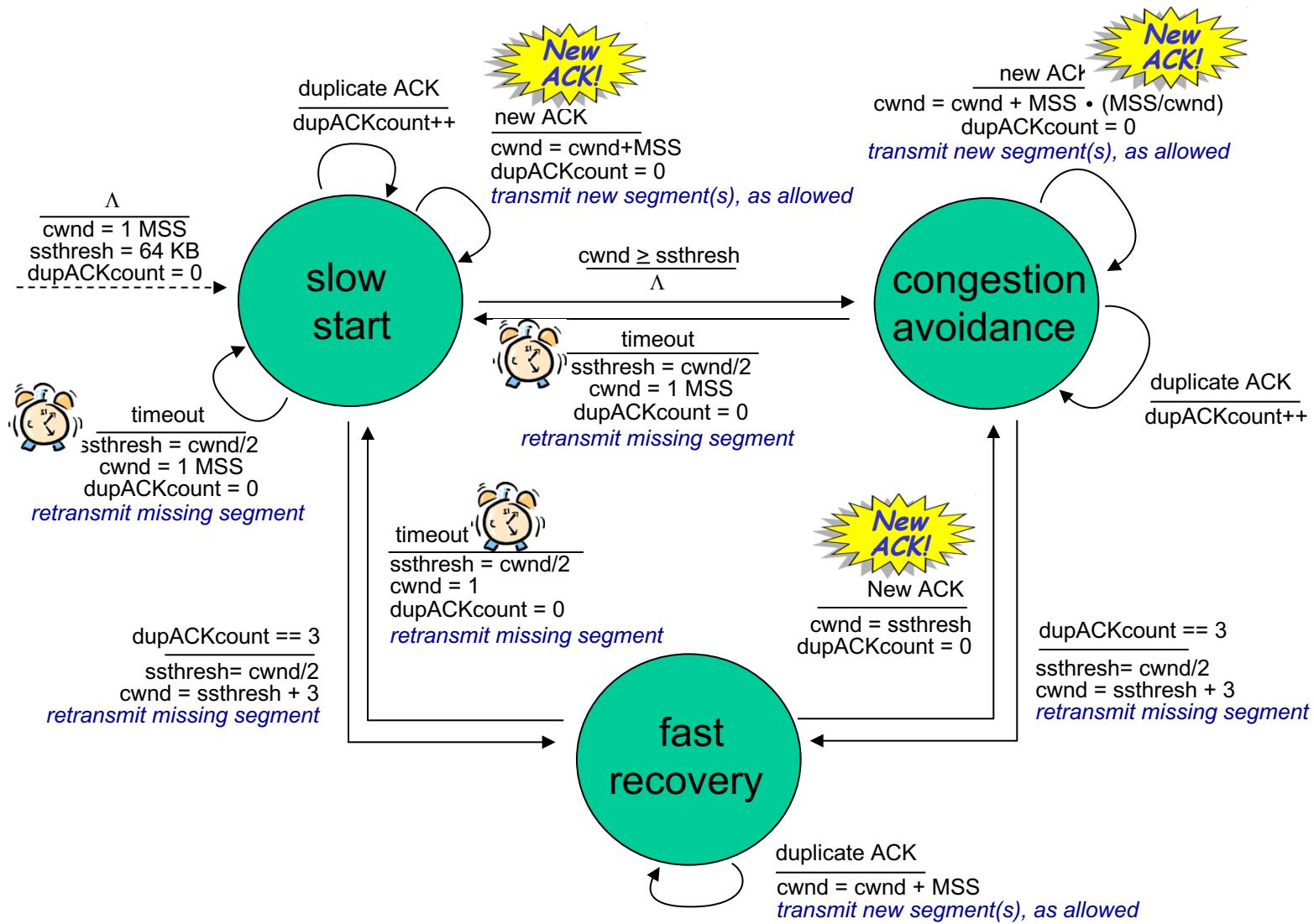
Implementation:

- variable **ssthresh**
- on loss event, **ssthresh** is set to 1/2 of **cwnd** just before loss event



* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

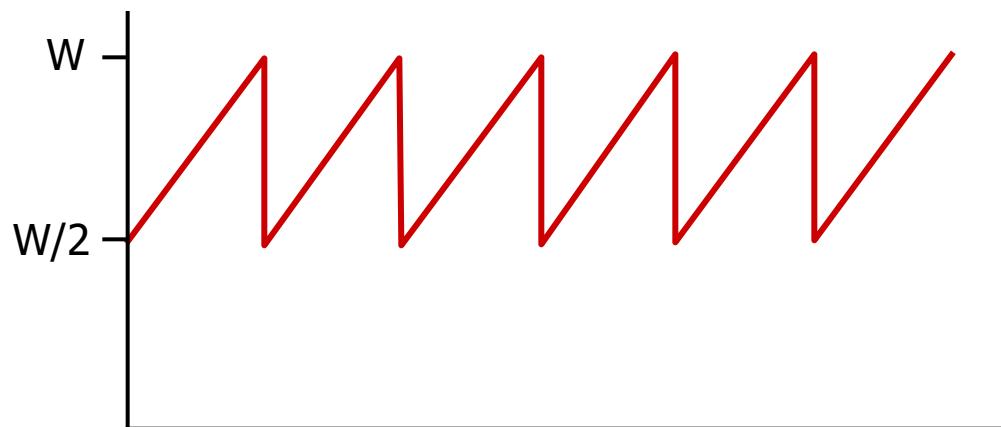
Summary: TCP Congestion Control



TCP throughput

- avg. TCP thruput as function of window size, RTT?
 - ignore slow start, assume always data to send
- W : window size (measured in bytes) where loss occurs
 - avg. window size (# in-flight bytes) is $\frac{3}{4} W$
 - avg. thruput is $\frac{3}{4}W$ per RTT

$$\text{avg TCP thruput} = \frac{3}{4} \frac{W}{\text{RTT}} \text{ bytes/sec}$$



TCP Futures: TCP over “long, fat pipes”

- example: 1500 byte segments, 100ms RTT, want 10 Gbps throughput
- requires $W = 83,333$ in-flight segments
- throughput in terms of segment loss probability, L [Mathis 1997]:

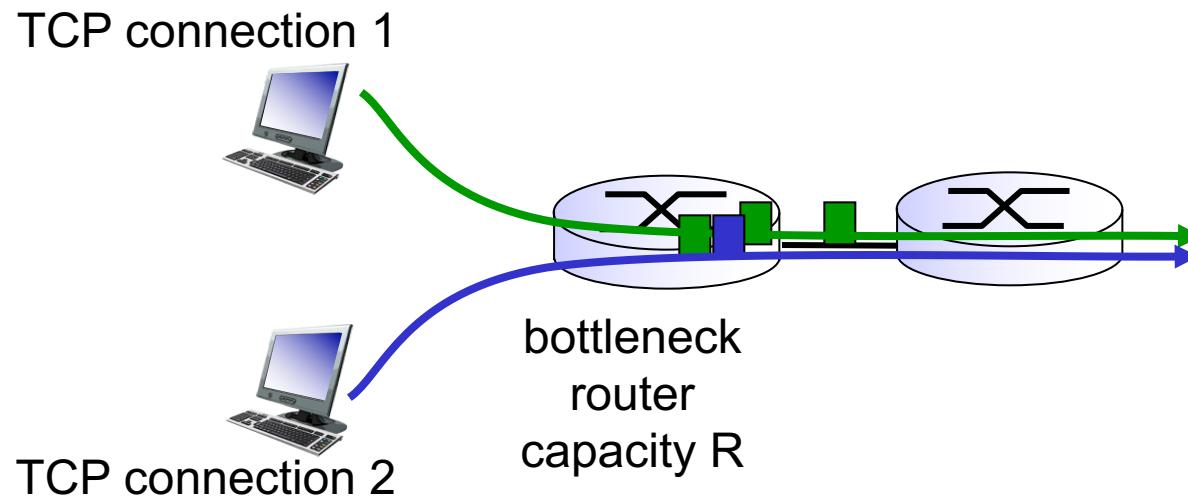
$$\text{TCP throughput} = \frac{1.22 \cdot \text{MSS}}{\text{RTT} \sqrt{L}}$$

→ to achieve 10 Gbps throughput, need a loss rate of $L = 2 \cdot 10^{-10}$ – *a very small loss rate!*

- new versions of TCP for high-speed

TCP Fairness

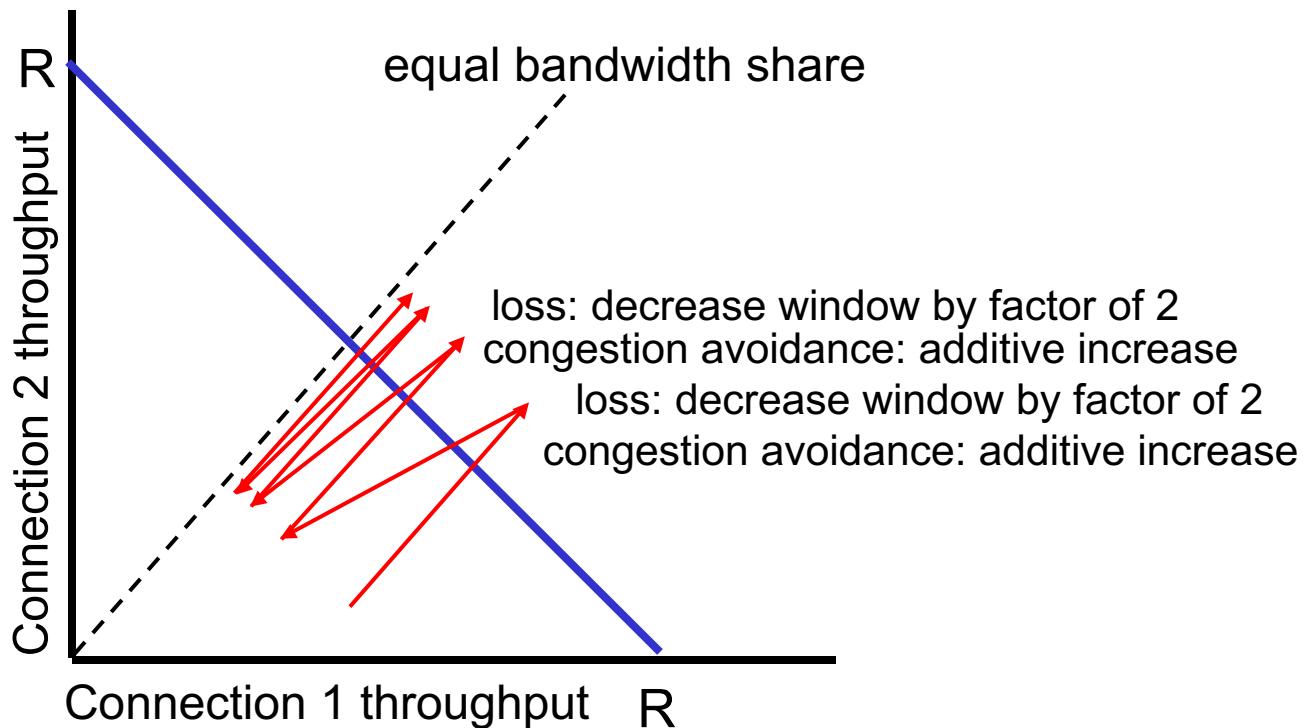
fairness goal: if K TCP sessions share same bottleneck link of bandwidth R , each should have average rate of R/K



Why is TCP fair?

two competing sessions:

- additive increase gives slope of 1, as throughput increases
- multiplicative decrease decreases throughput proportionally



Fairness (more)

Fairness and UDP

- multimedia apps often do not use TCP
 - do not want rate throttled by congestion control
- instead use UDP:
 - send audio/video at constant rate, tolerate packet loss

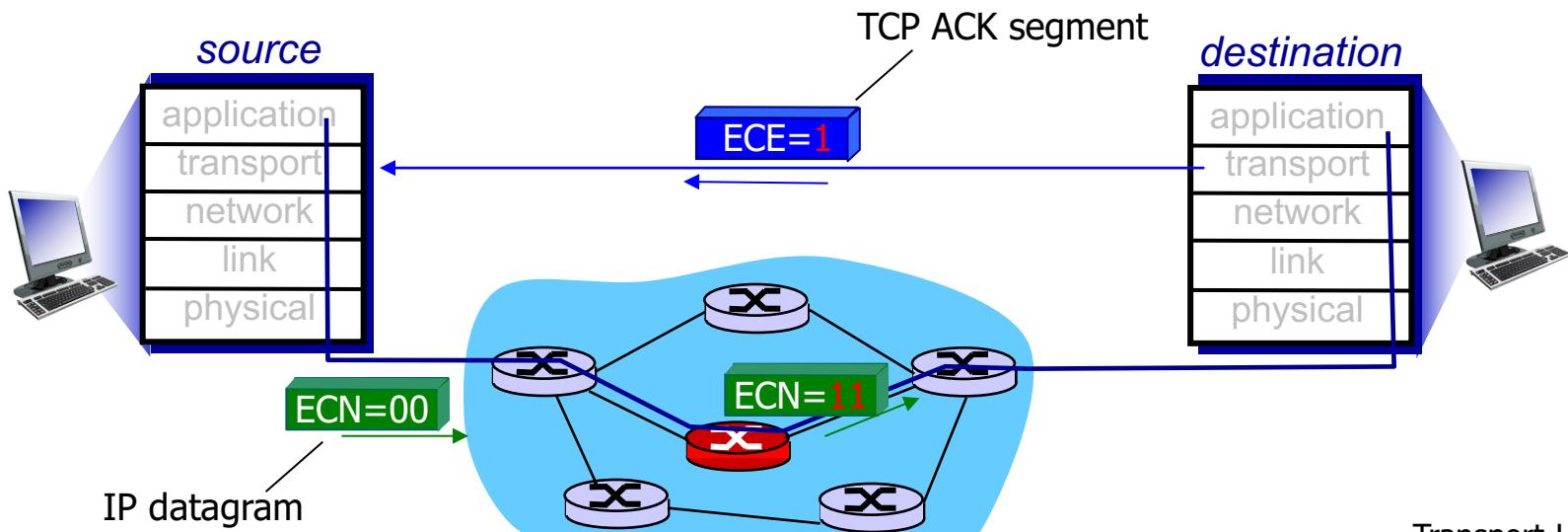
Fairness, parallel TCP connections

- application can open multiple parallel connections between two hosts
- web browsers do this
- e.g., link of rate R with 9 existing connections:
 - new app asks for 1 TCP, gets rate $R/10$
 - new app asks for 11 TCPs, gets $R/2$

Explicit Congestion Notification (ECN)

network-assisted congestion control:

- two bits in IP header (ToS field) marked *by network router* to indicate congestion
- congestion indication carried to receiving host
- receiver (seeing congestion indication in IP datagram)) sets ECE bit on receiver-to-sender ACK segment to notify sender of congestion



Chapter 3: summary

- principles behind transport layer services:
 - multiplexing, demultiplexing
 - reliable data transfer
 - flow control
 - congestion control
- instantiation, implementation in the Internet
 - UDP
 - TCP

next:

- leaving the network “edge” (application, transport layers)
- into the network “core”
- two network layer chapters:
 - data plane
 - control plane

Chapter 4

Network Layer:

The Data Plane

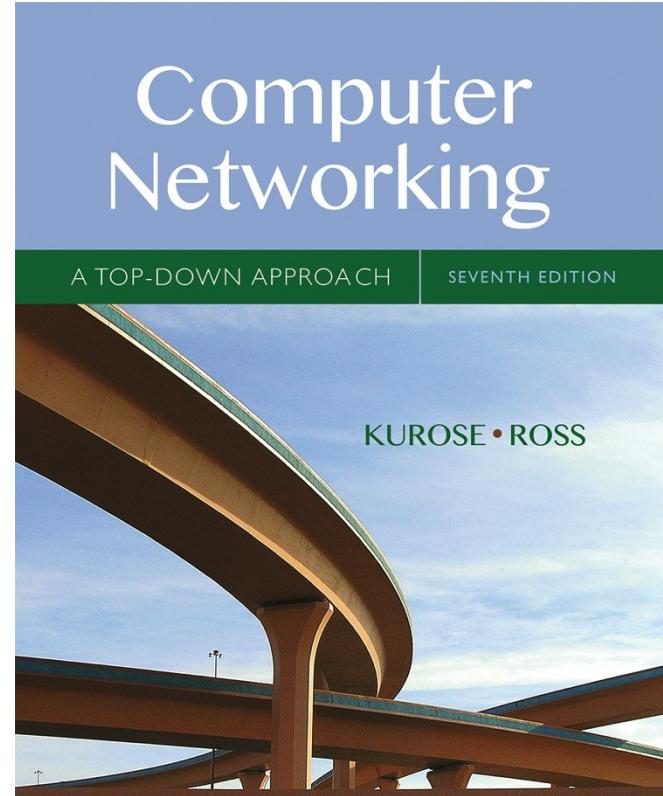
A note on the use of these Powerpoint slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR

© All material copyright 1996-2016
J.F Kurose and K.W. Ross, All Rights Reserved



*Computer
Networking: A Top
Down Approach*

7th edition

Jim Kurose, Keith Ross
Pearson/Addison Wesley
April 2016

Chapter 4: outline

4.1 Overview of Network layer

- data plane
- control plane

4.2 What's inside a router

4.3 IP: Internet Protocol

- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

4.4 Generalized Forward and SDN

- match
- action
- OpenFlow examples of match-plus-action in action

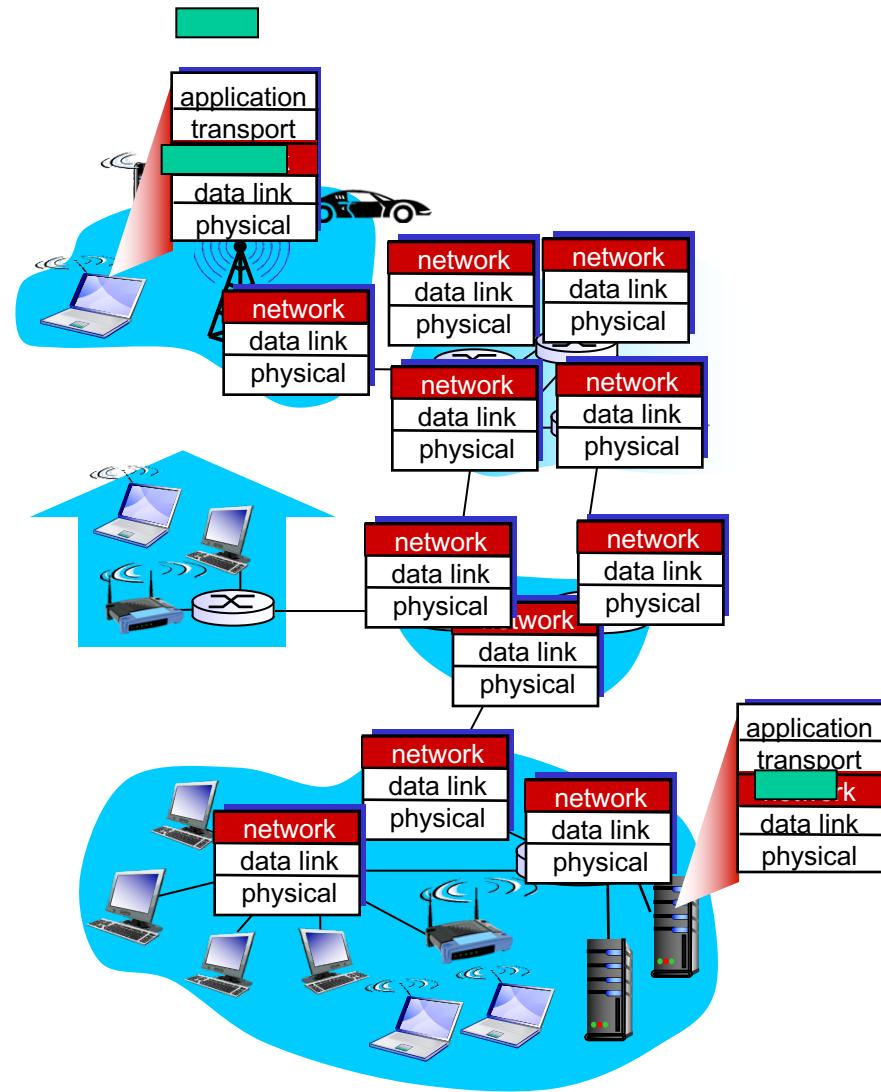
Chapter 4: network layer

chapter goals:

- understand principles behind network layer services, focusing on data plane:
 - network layer service models
 - forwarding versus routing
 - how a router works
 - generalized forwarding
- instantiation, implementation in the Internet

Network layer

- transport segment from sending to receiving host
- on sending side encapsulates segments into datagrams
- on receiving side, delivers segments to transport layer
- network layer protocols in **every** host, router
- router examines header fields in all IP datagrams passing through it



Two key network-layer functions

network-layer functions:

- **forwarding:** move packets from router's input to appropriate router output
- **routing:** determine route taken by packets from source to destination
 - *routing algorithms*

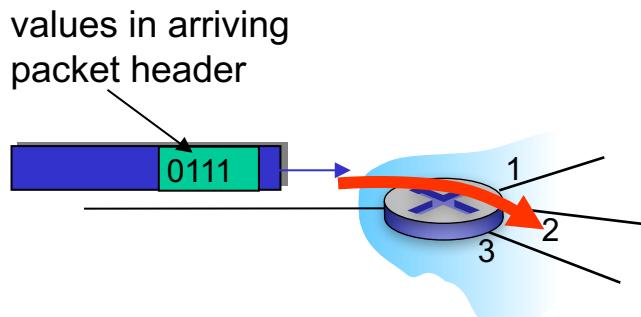
analogy: taking a trip

- **forwarding:** process of getting through single interchange
- **routing:** process of planning trip from source to destination

Network layer: data plane, control plane

Data plane

- local, per-router function
- determines how datagram arriving on router input port is forwarded to router output port
- forwarding function

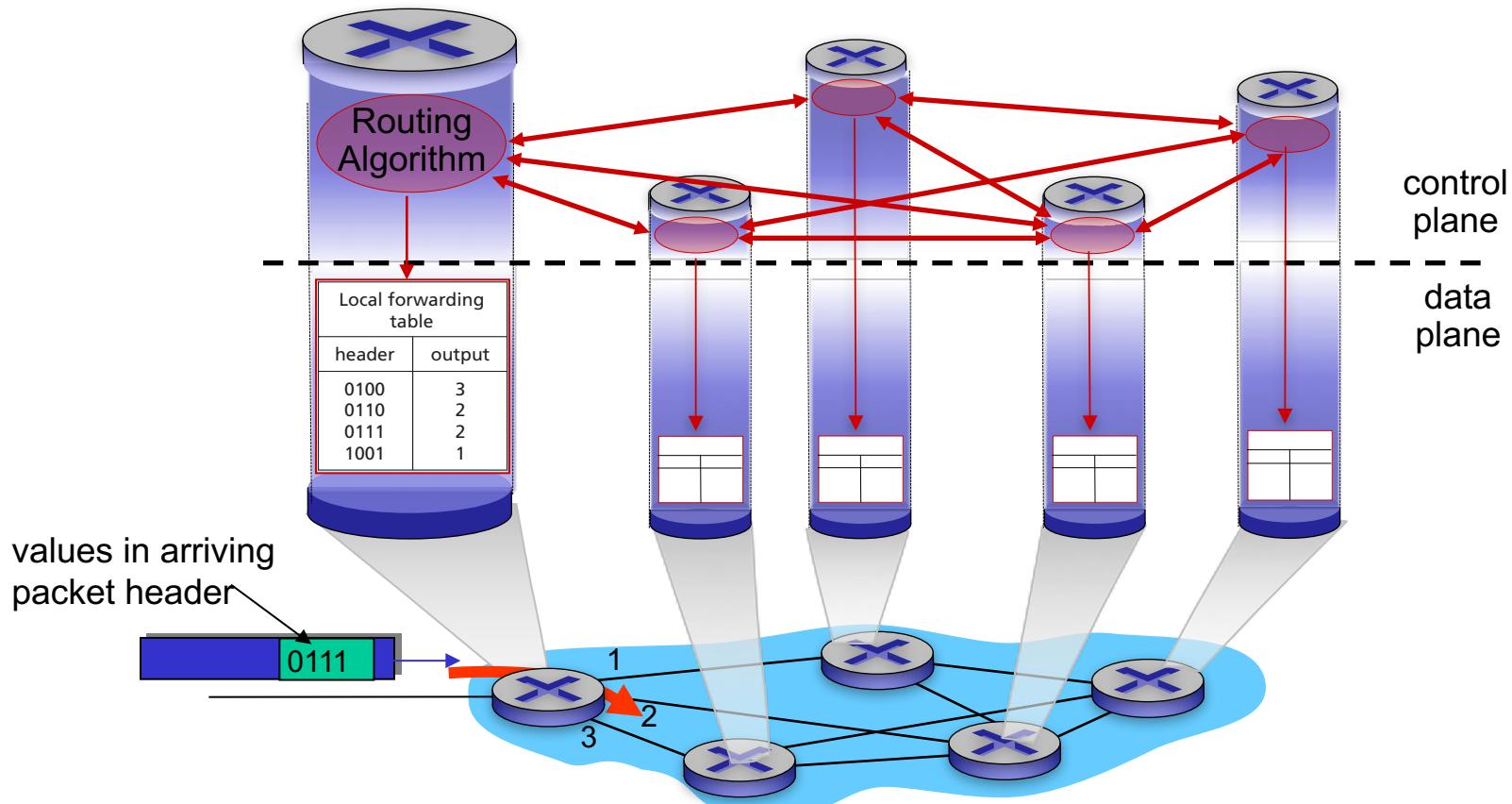


Control plane

- network-wide logic
- determines how datagram is routed among routers along end-end path from source host to destination host
- two control-plane approaches:
 - *traditional routing algorithms*: implemented in routers
 - *software-defined networking (SDN)*: implemented in (remote) servers

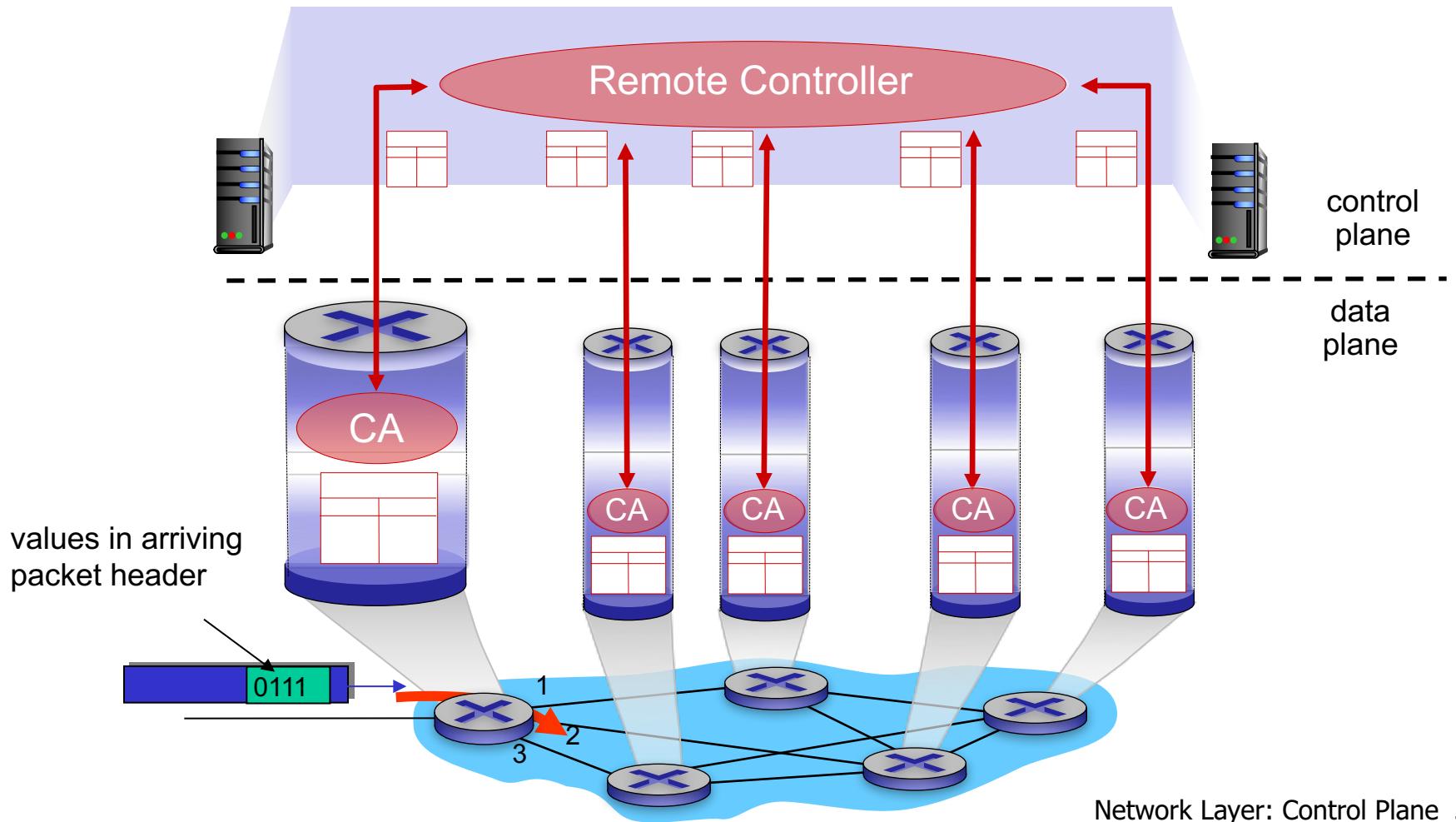
Per-router control plane

Individual routing algorithm components *in each and every router* interact in the control plane



Logically centralized control plane

A distinct (typically remote) controller interacts with local control agents (CAs)



Network service model

Q: What *service model* for “channel” transporting datagrams from sender to receiver?

example services for individual datagrams:

- guaranteed delivery
- guaranteed delivery with less than 40 msec delay

example services for a flow of datagrams:

- in-order datagram delivery
- guaranteed minimum bandwidth to flow
- restrictions on changes in inter-packet spacing

Network layer service models:

Network Architecture	Service Model	Guarantees ?				Congestion feedback
		Bandwidth	Loss	Order	Timing	
Internet	best effort	none	no	no	no	no (inferred via loss)
ATM	CBR	constant rate	yes	yes	yes	no congestion
ATM	VBR	guaranteed rate	yes	yes	yes	no congestion
ATM	ABR	guaranteed minimum	no	yes	no	yes
ATM	UBR	none	no	yes	no	no

Chapter 4: outline

4.1 Overview of Network layer

- data plane
- control plane

4.2 What's inside a router

4.3 IP: Internet Protocol

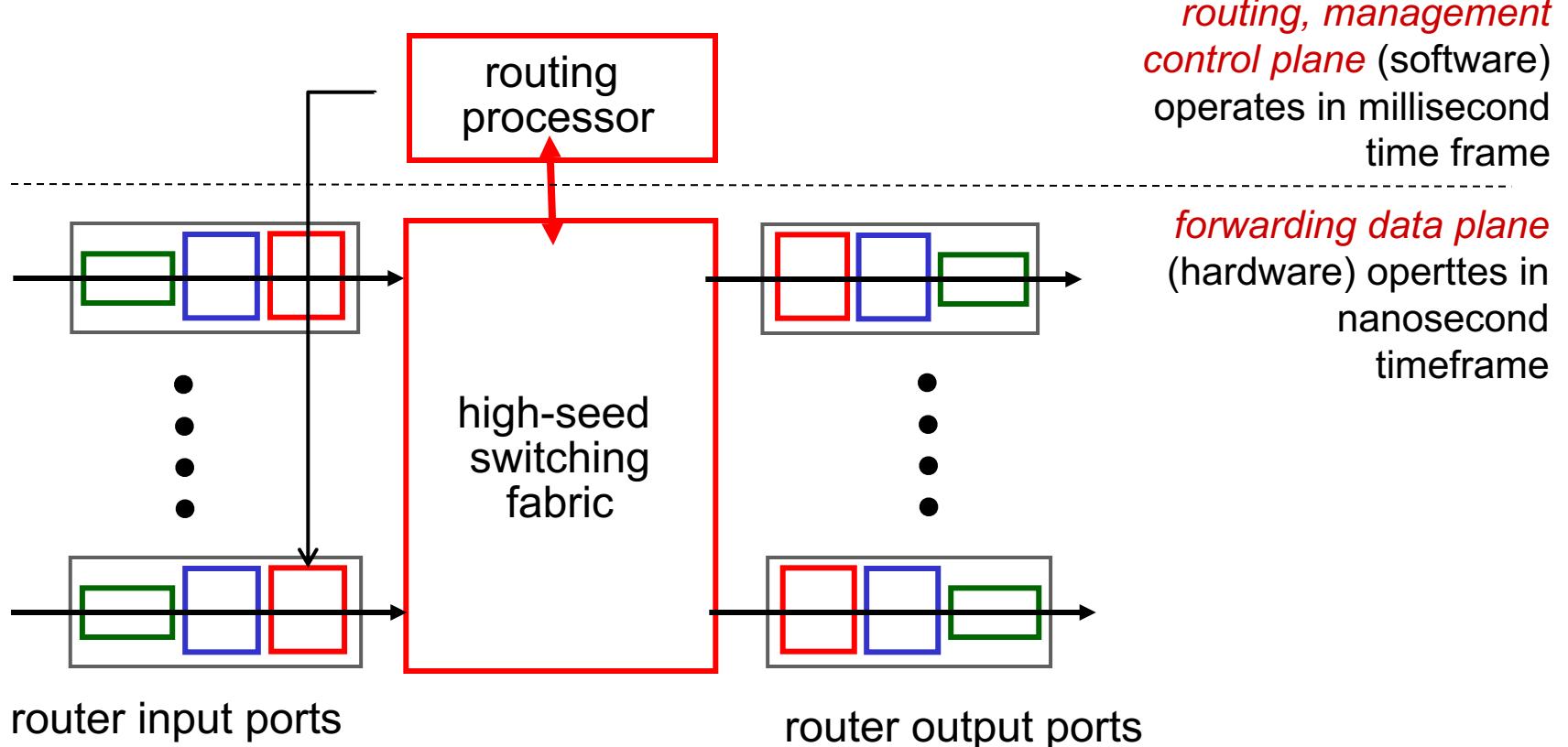
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

4.4 Generalized Forward and SDN

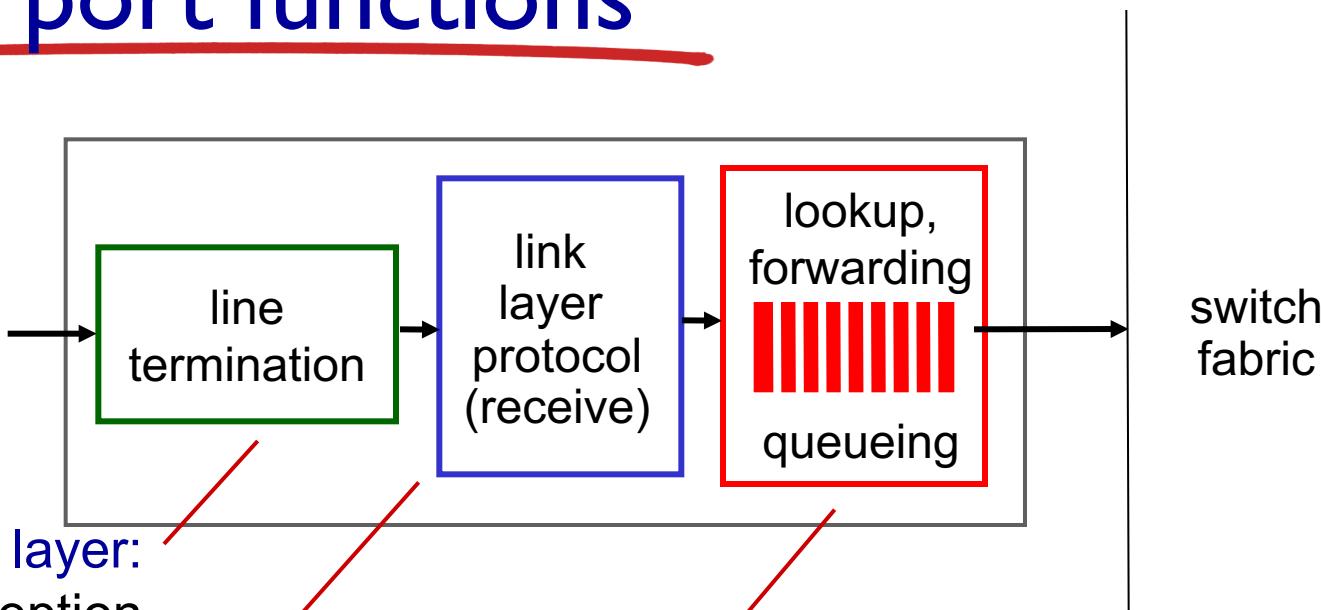
- match
- action
- OpenFlow examples of match-plus-action in action

Router architecture overview

- high-level view of generic router architecture:



Input port functions



physical layer:

bit-level reception

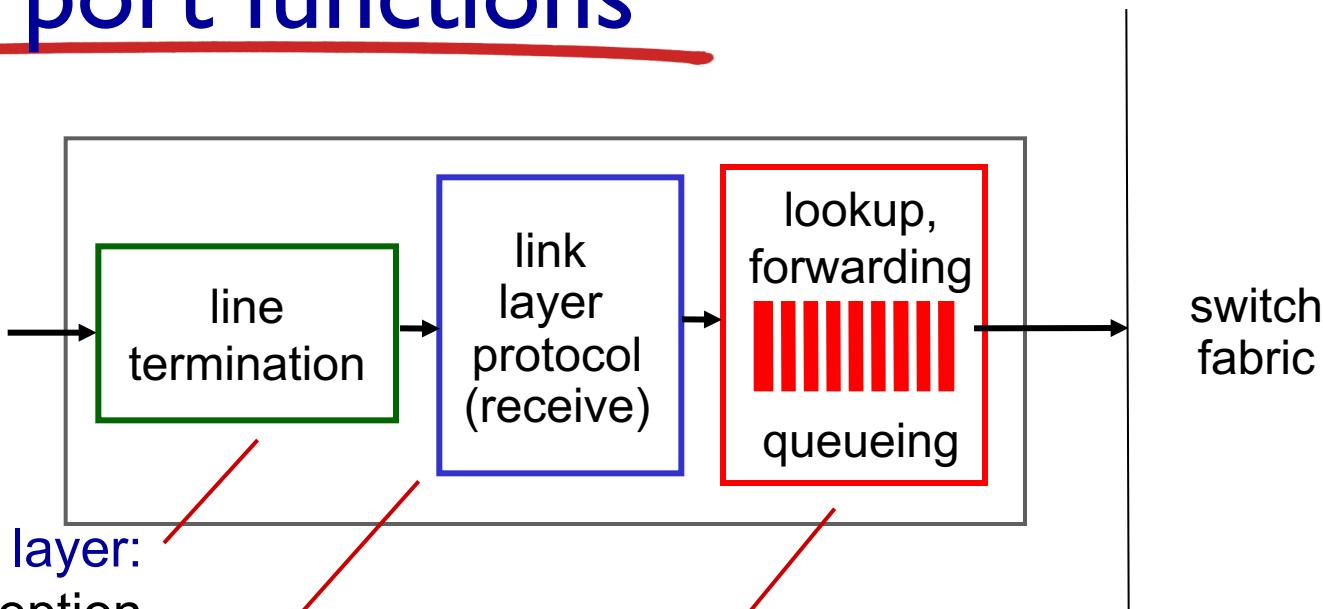
data link layer:

e.g., Ethernet
see chapter 5

decentralized switching:

- using header field values, lookup output port using forwarding table in input port memory (“*match plus action*”)
- goal: complete input port processing at ‘line speed’
- queuing: if datagrams arrive faster than forwarding rate into switch fabric

Input port functions



physical layer:

bit-level reception

data link layer:

e.g., Ethernet
see chapter 5

decentralized switching:

- using header field values, lookup output port using forwarding table in input port memory (“*match plus action*”)
- *destination-based forwarding*: forward based only on destination IP address (traditional)
- *generalized forwarding*: forward based on any set of header field values

Destination-based forwarding

forwarding table

Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111	2
otherwise	3

Q: but what happens if ranges don't divide up so nicely?

Longest prefix matching

longest prefix matching —

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range	Link interface
11001000 00010111 00010*** *****	0
11001000 00010111 00011000 *****	1
11001000 00010111 00011*** *****	2
otherwise	3

examples:

DA: 11001000 00010111 00010110 10100001

which interface?

DA: 11001000 00010111 00011000 10101010

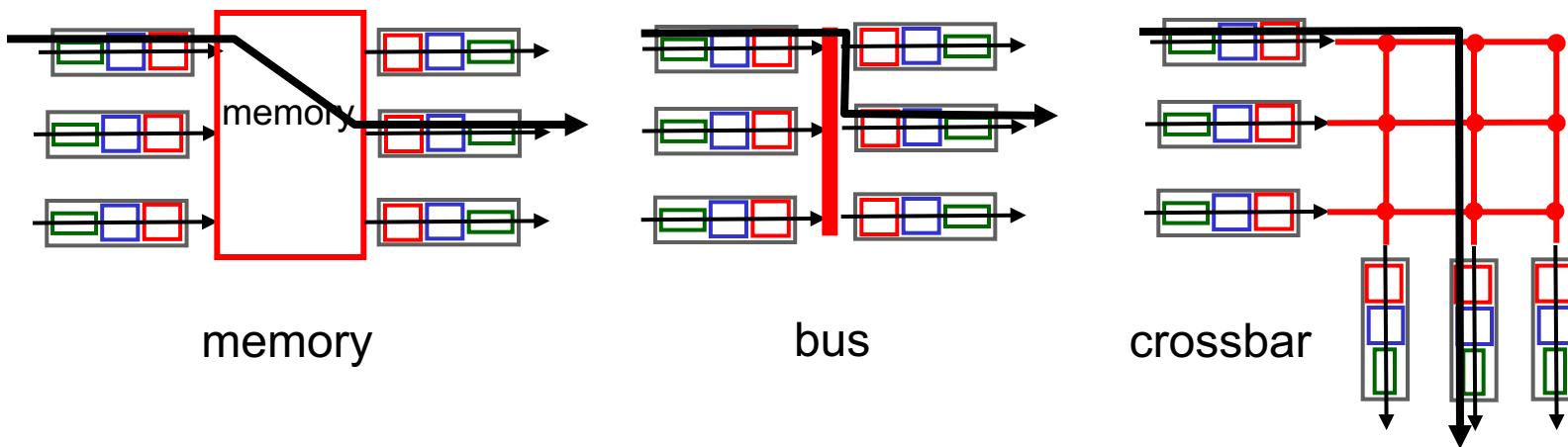
which interface?

Longest prefix matching

- we'll see *why* longest prefix matching is used shortly, when we study addressing
- longest prefix matching: often performed using ternary content addressable memories (TCAMs)
 - *content addressable*: present address to TCAM: retrieve address in one clock cycle, regardless of table size
 - Cisco Catalyst: can up ~1M routing table entries in TCAM

Switching fabrics

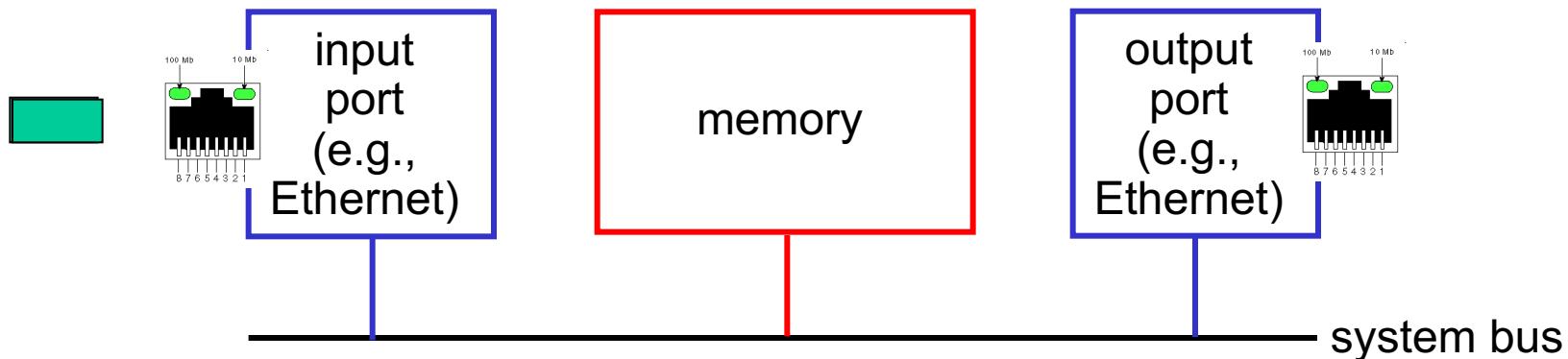
- transfer packet from input buffer to appropriate output buffer
- switching rate: rate at which packets can be transferred from inputs to outputs
 - often measured as multiple of input/output line rate
 - N inputs: switching rate N times line rate desirable
- three types of switching fabrics



Switching via memory

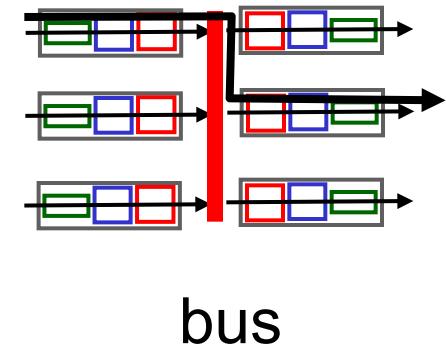
first generation routers:

- traditional computers with switching under direct control of CPU
- packet copied to system's memory
- speed limited by memory bandwidth (2 bus crossings per datagram)



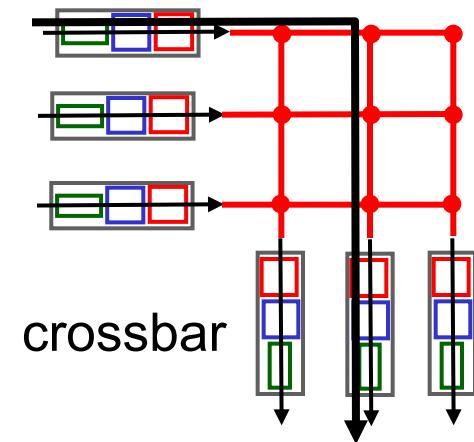
Switching via a bus

- datagram from input port memory to output port memory via a shared bus
- *bus contention*: switching speed limited by bus bandwidth
- 32 Gbps bus, Cisco 5600: sufficient speed for access and enterprise routers



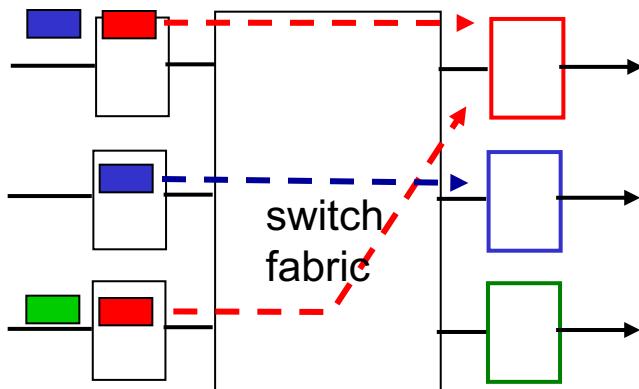
Switching via interconnection network

- overcome bus bandwidth limitations
- banyan networks, crossbar, other interconnection nets initially developed to connect processors in multiprocessor
- advanced design: fragmenting datagram into fixed length cells, switch cells through the fabric.
- Cisco 12000: switches 60 Gbps through the interconnection network

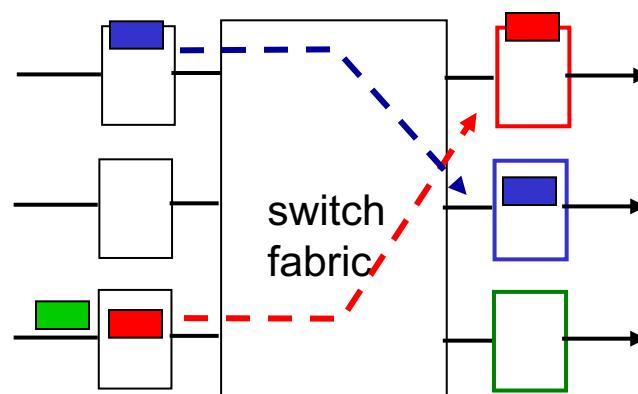


Input port queuing

- fabric slower than input ports combined -> queueing may occur at input queues
 - *queueing delay and loss due to input buffer overflow!*
- **Head-of-the-Line (HOL) blocking:** queued datagram at front of queue prevents others in queue from moving forward



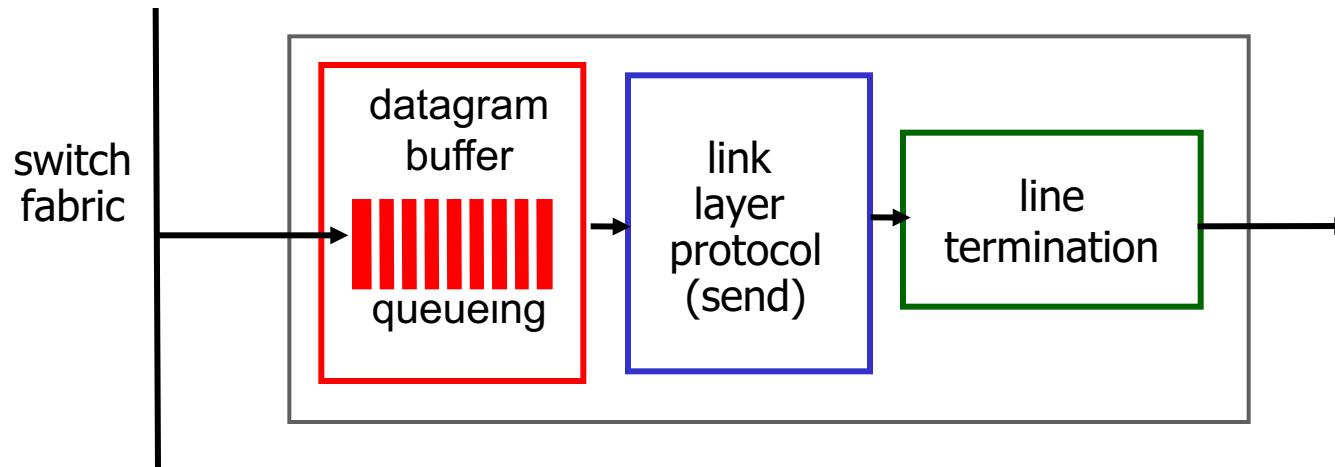
output port contention:
only one red datagram can be
transferred.
lower red packet is blocked



one packet time later:
green packet
experiences HOL
blocking

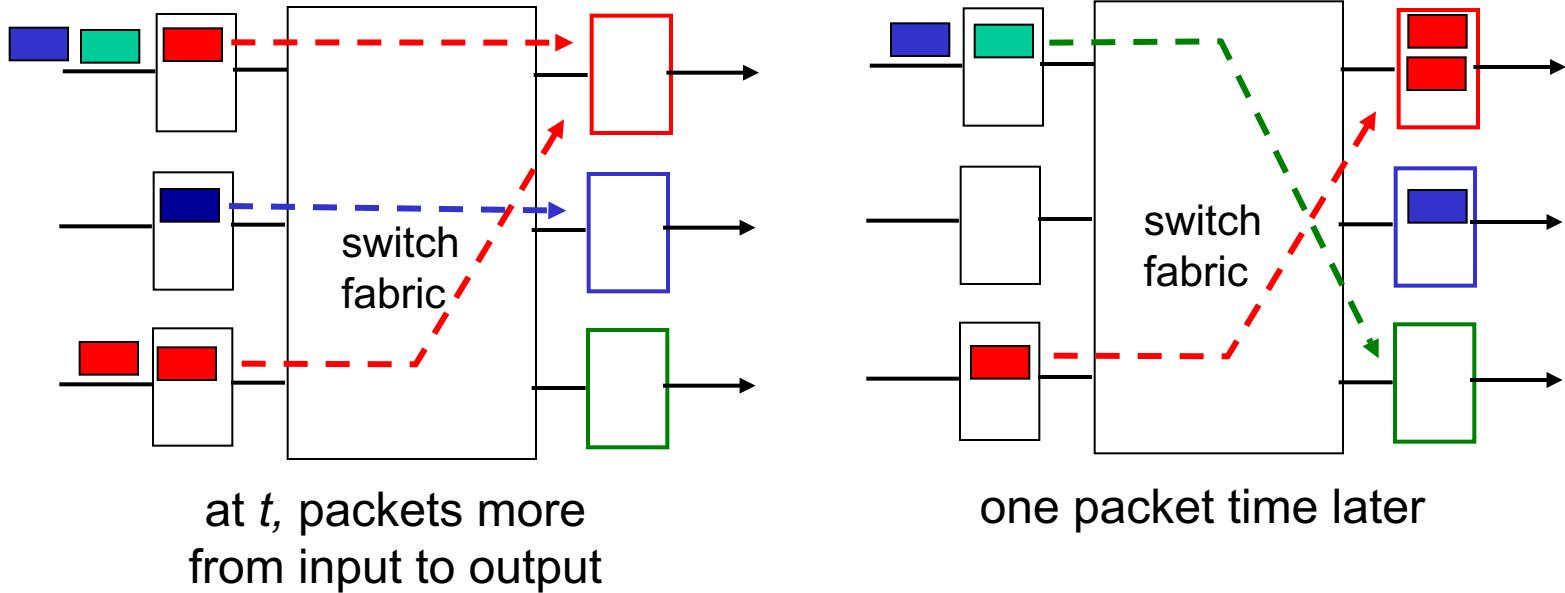
Output ports

This slide is HUGELY important!



- ***buffering*** required from fabric faster rate
 - Datagram (packets) can be lost due to congestion, lack of buffers
- ***scheduling*** datagrams
 - Priority scheduling – who gets best performance, network neutrality

Output port queueing



- buffering when arrival rate via switch exceeds output line speed
- *queueing (delay) and loss due to output port buffer overflow!*

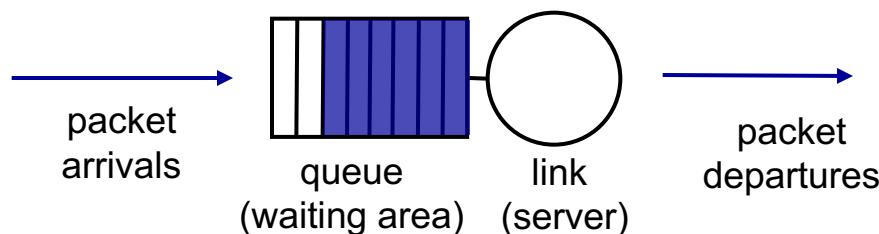
How much buffering?

- RFC 3439 rule of thumb: average buffering equal to “typical” RTT (say 250 msec) times link capacity C
 - e.g., $C = 10 \text{ Gpbs}$ link: 2.5 Gbit buffer
- recent recommendation: with N flows, buffering equal to

$$\frac{\text{RTT} \cdot C}{\sqrt{N}}$$

Scheduling mechanisms

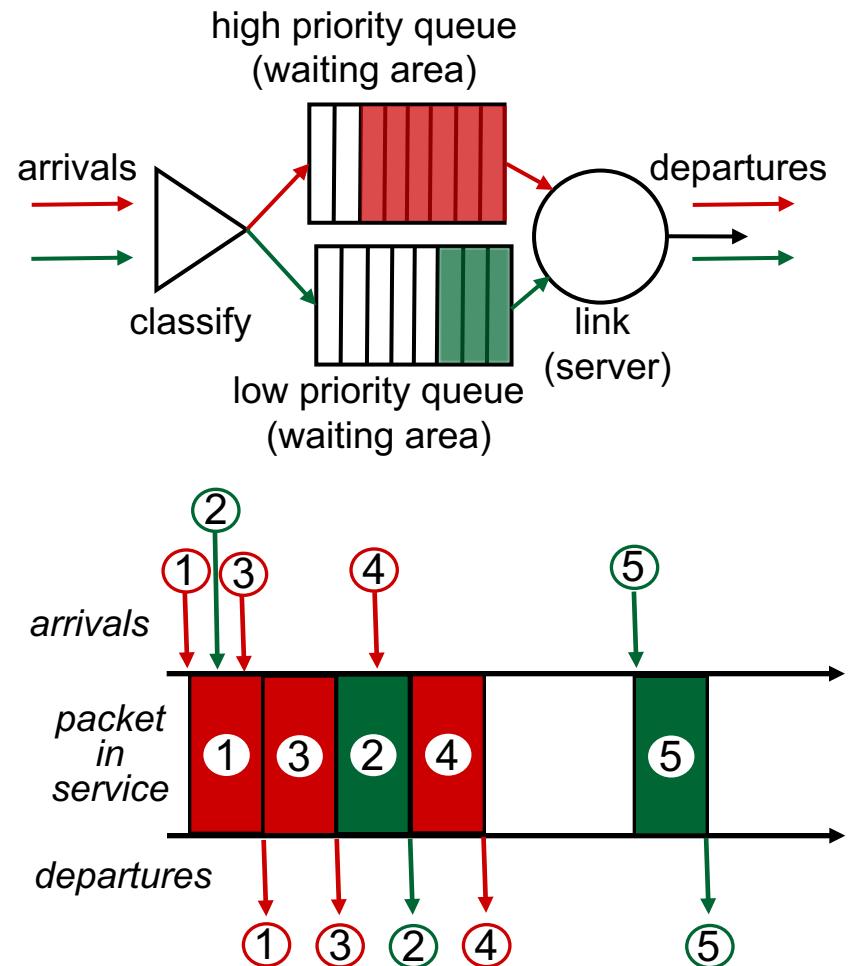
- *scheduling*: choose next packet to send on link
- *FIFO (first in first out) scheduling*: send in order of arrival to queue
 - real-world example?
 - *discard policy*: if packet arrives to full queue: who to discard?
 - *tail drop*: drop arriving packet
 - *priority*: drop/remove on priority basis
 - *random*: drop/remove randomly



Scheduling policies: priority

priority scheduling: send highest priority queued packet

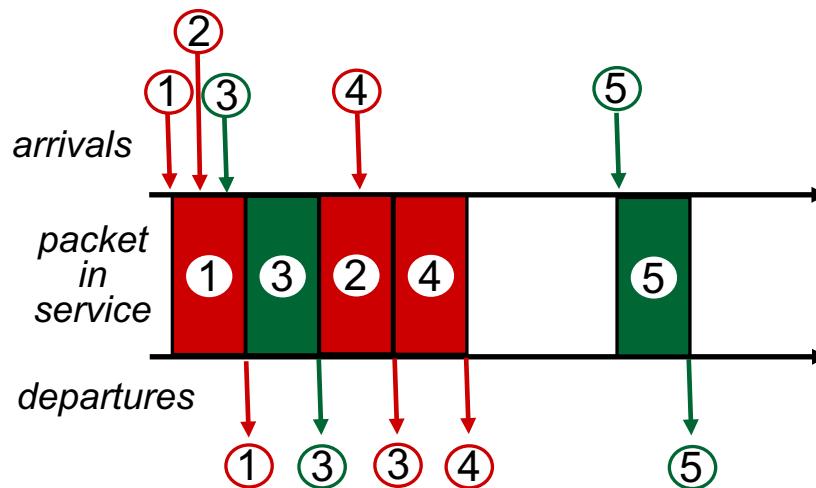
- multiple classes, with different priorities
 - class may depend on marking or other header info, e.g. IP source/dest, port numbers, etc.
 - real world example?



Scheduling policies: still more

Round Robin (RR) scheduling:

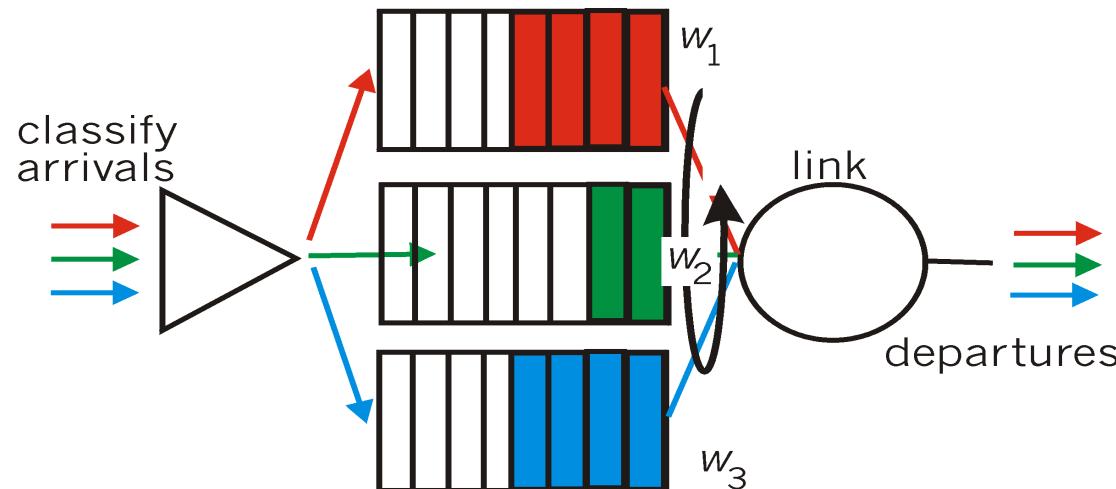
- multiple classes
- cyclically scan class queues, sending one complete packet from each class (if available)
- real world example?



Scheduling policies: still more

Weighted Fair Queuing (WFQ):

- generalized Round Robin
- each class gets weighted amount of service in each cycle
- real-world example?



Chapter 4: outline

4.1 Overview of Network layer

- data plane
- control plane

4.2 What's inside a router

4.3 IP: Internet Protocol

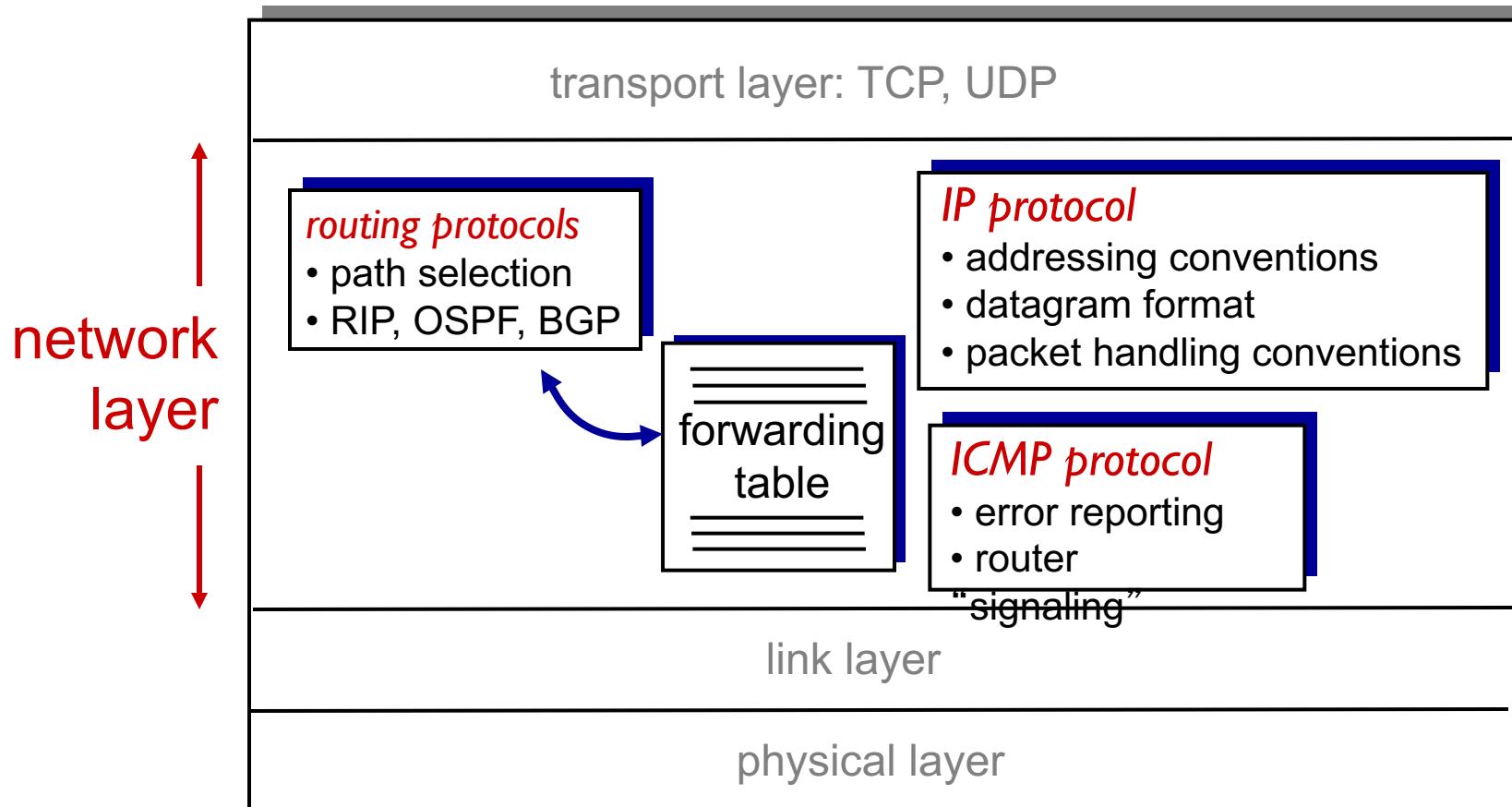
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

4.4 Generalized Forward and SDN

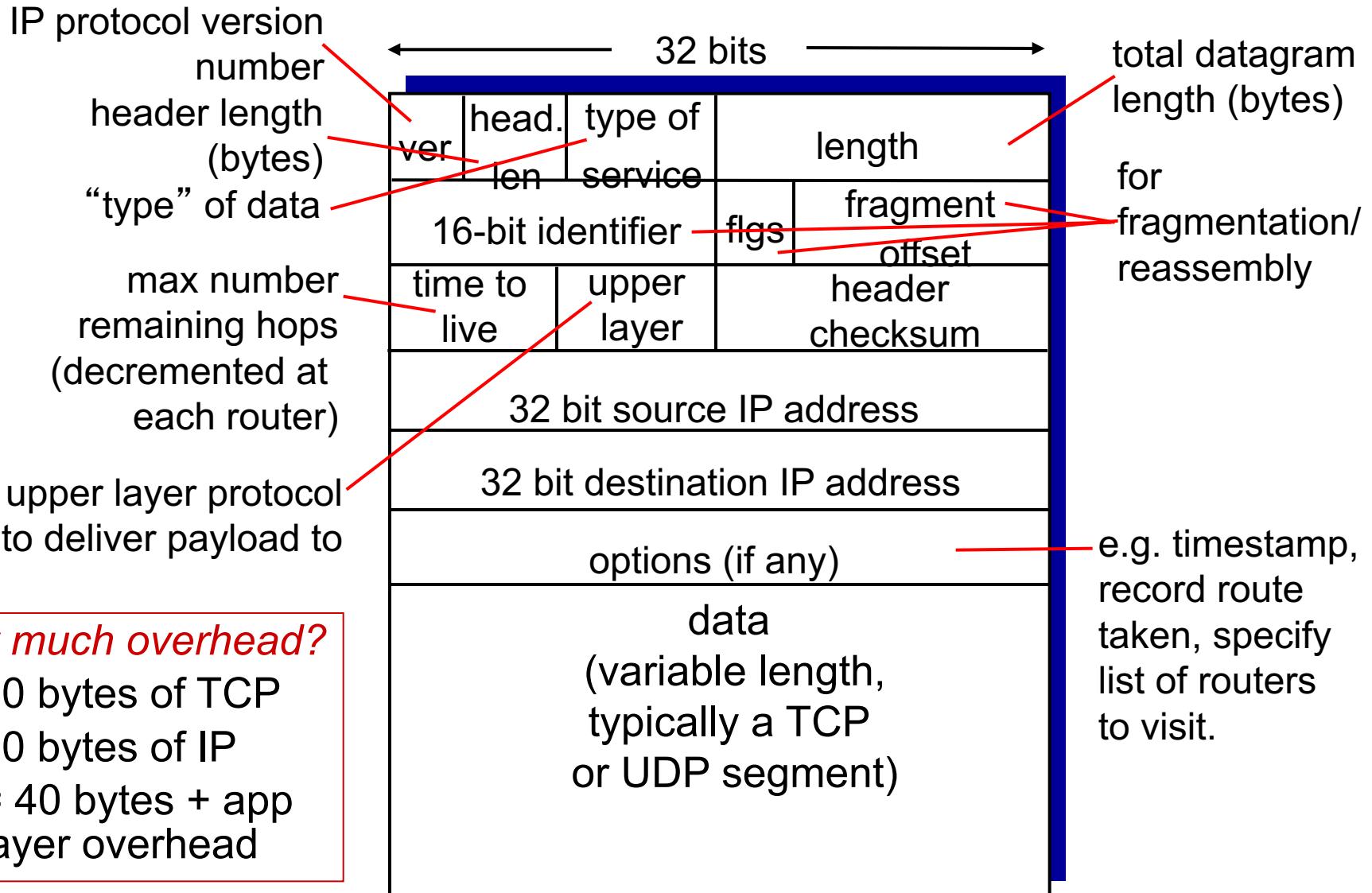
- match
- action
- OpenFlow examples of match-plus-action in action

The Internet network layer

host, router network layer functions:

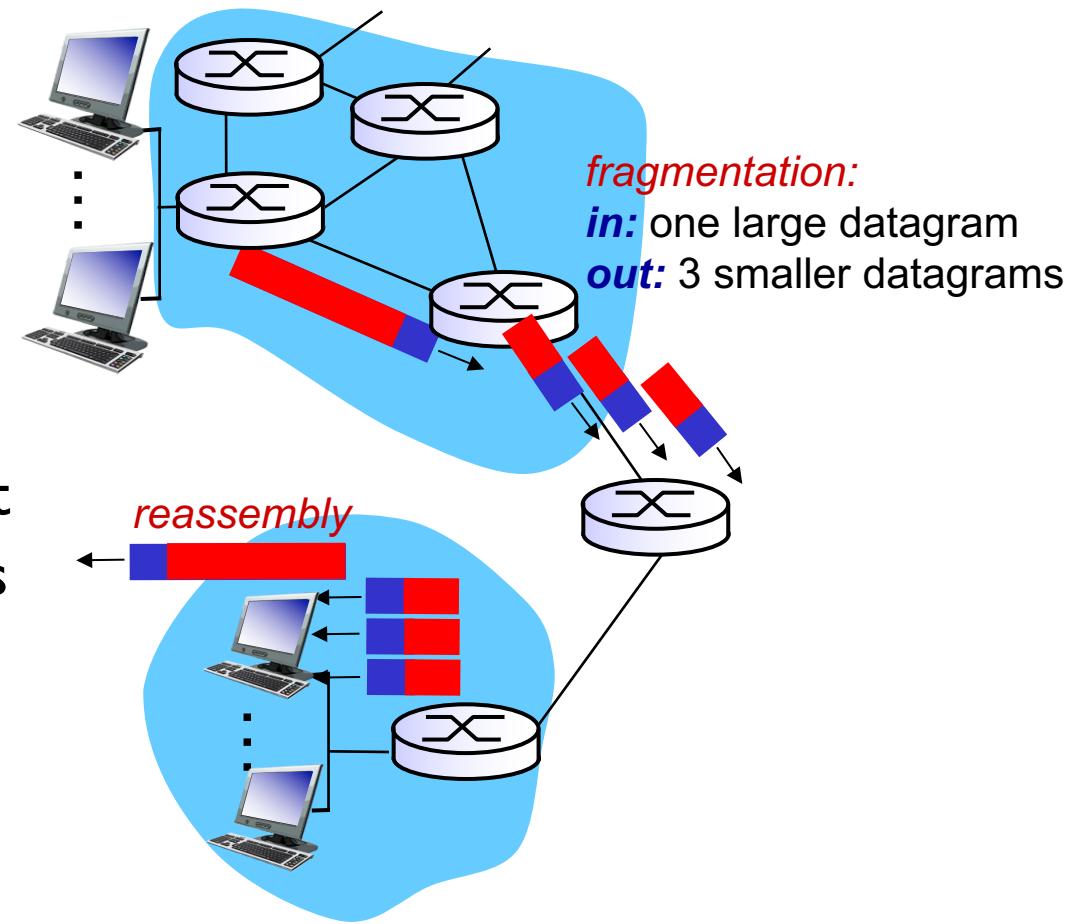


IP datagram format



IP fragmentation, reassembly

- network links have MTU (max.transfer size) - largest possible link-level frame
 - different link types, different MTUs
- large IP datagram divided (“fragmented”) within net
 - one datagram becomes several datagrams
 - “reassembled” only at final destination
 - IP header bits used to identify, order related fragments



IP fragmentation, reassembly

example:

- ❖ 4000 byte datagram
- ❖ MTU = 1500 bytes

1480 bytes in
data field

offset =
 $1480/8$

	length =4000	ID =x	fragflag =0	offset =0	
--	-----------------	----------	----------------	--------------	--

*one large datagram becomes
several smaller datagrams*

	length =1500	ID =x	fragflag =1	offset =0	
--	-----------------	----------	----------------	--------------	--

	length =1500	ID =x	fragflag =1	offset =185	
--	-----------------	----------	----------------	----------------	--

	length =1040	ID =x	fragflag =0	offset =370	
--	-----------------	----------	----------------	----------------	--

Chapter 4: outline

4.1 Overview of Network layer

- data plane
- control plane

4.2 What's inside a router

4.3 IP: Internet Protocol

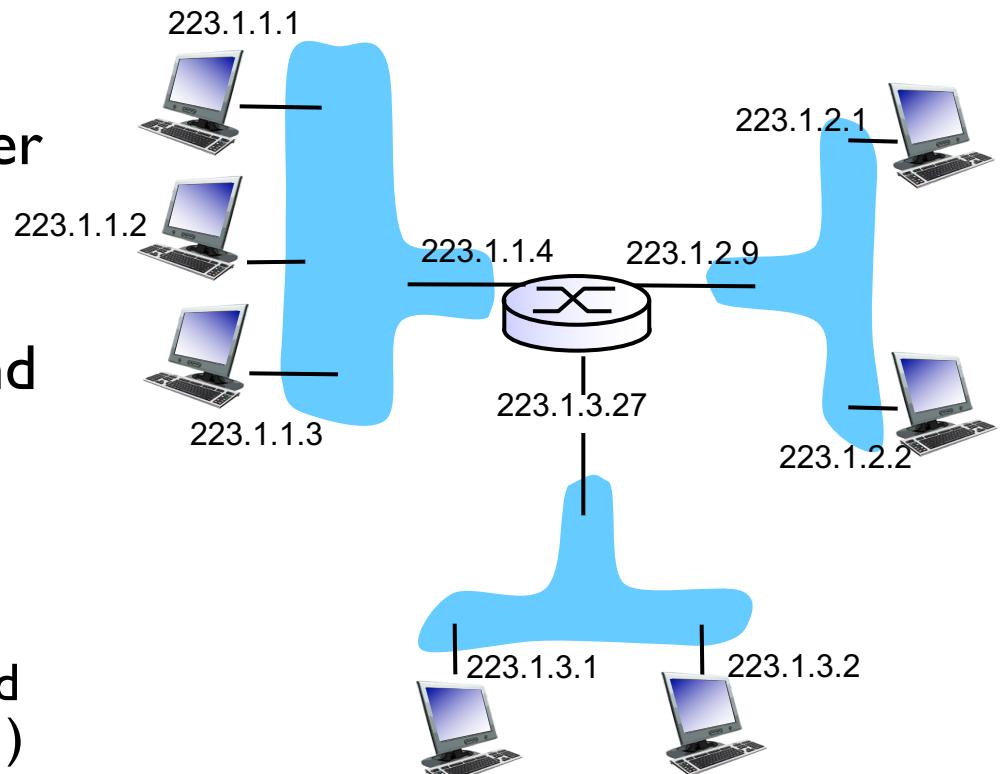
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

4.4 Generalized Forward and SDN

- match
- action
- OpenFlow examples of match-plus-action in action

IP addressing: introduction

- **IP address:** 32-bit identifier for host, router interface
- **interface:** connection between host/router and physical link
 - router's typically have multiple interfaces
 - host typically has one or two interfaces (e.g., wired Ethernet, wireless 802.11)
- **IP addresses associated with each interface**



$223.1.1.1 = \underbrace{11011111}_\text{223} \underbrace{00000001}_\text{1} \underbrace{00000001}_\text{1} \underbrace{00000001}_\text{1}$

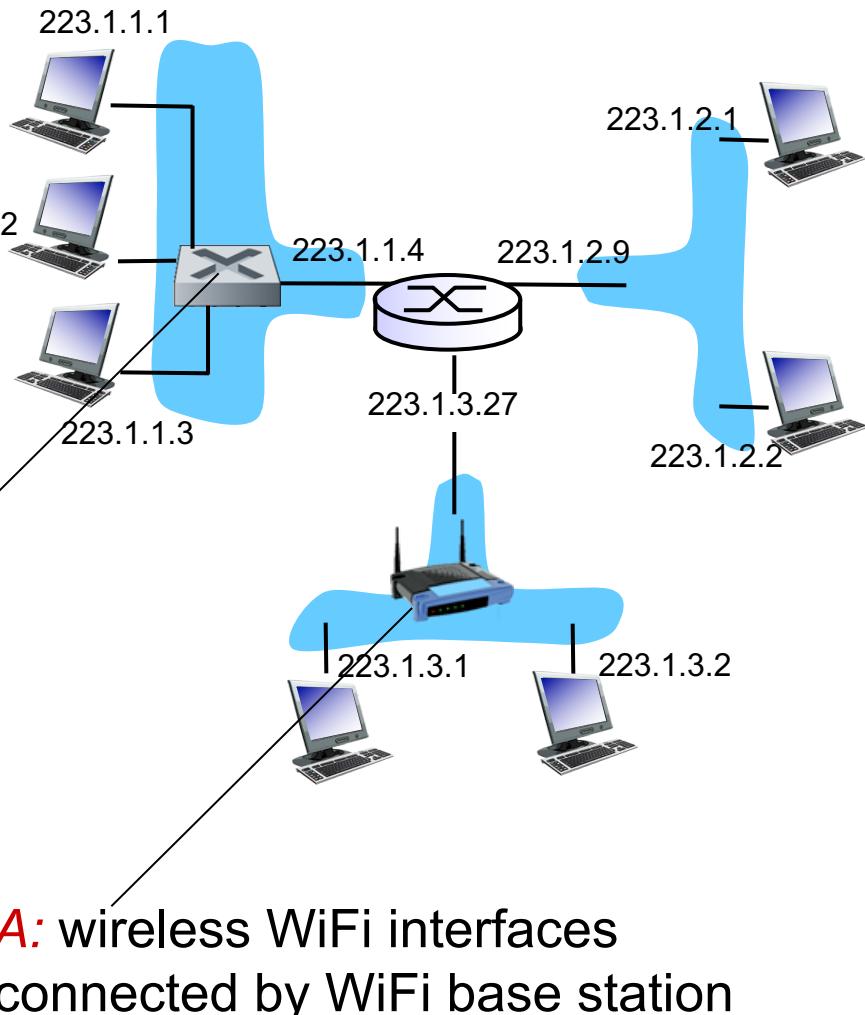
IP addressing: introduction

Q: how are interfaces actually connected?

A: we'll learn about that in chapter 5, 6.

A: wired Ethernet interfaces connected by Ethernet switches

For now: don't need to worry about how one interface is connected to another (with no intervening router)



A: wireless WiFi interfaces connected by WiFi base station

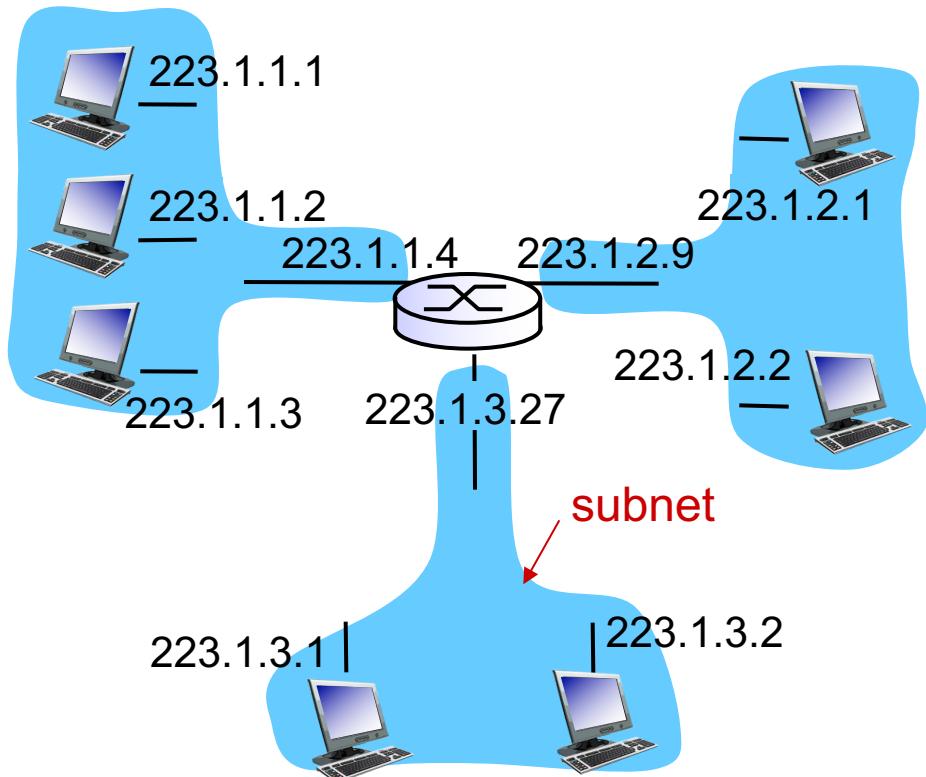
Subnets

■ IP address:

- subnet part - high order bits
- host part - low order bits

■ what's a subnet ?

- device interfaces with same subnet part of IP address
- can physically reach each other *without intervening router*

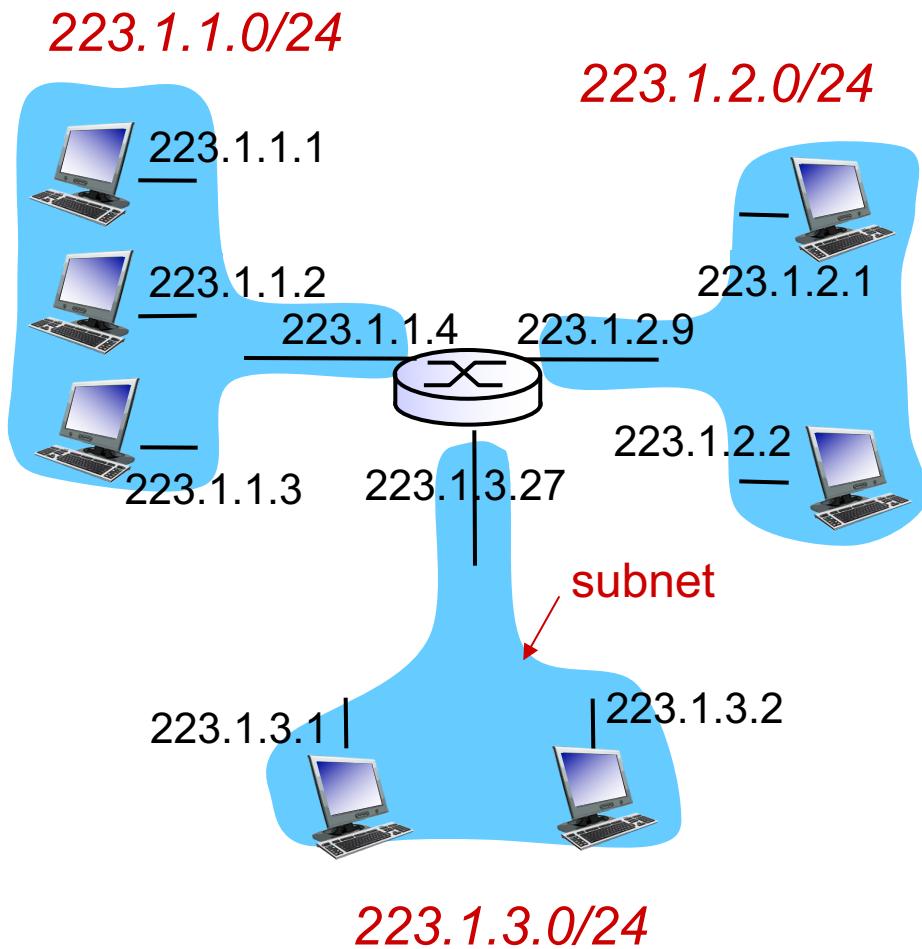


network consisting of 3 subnets

Subnets

recipe

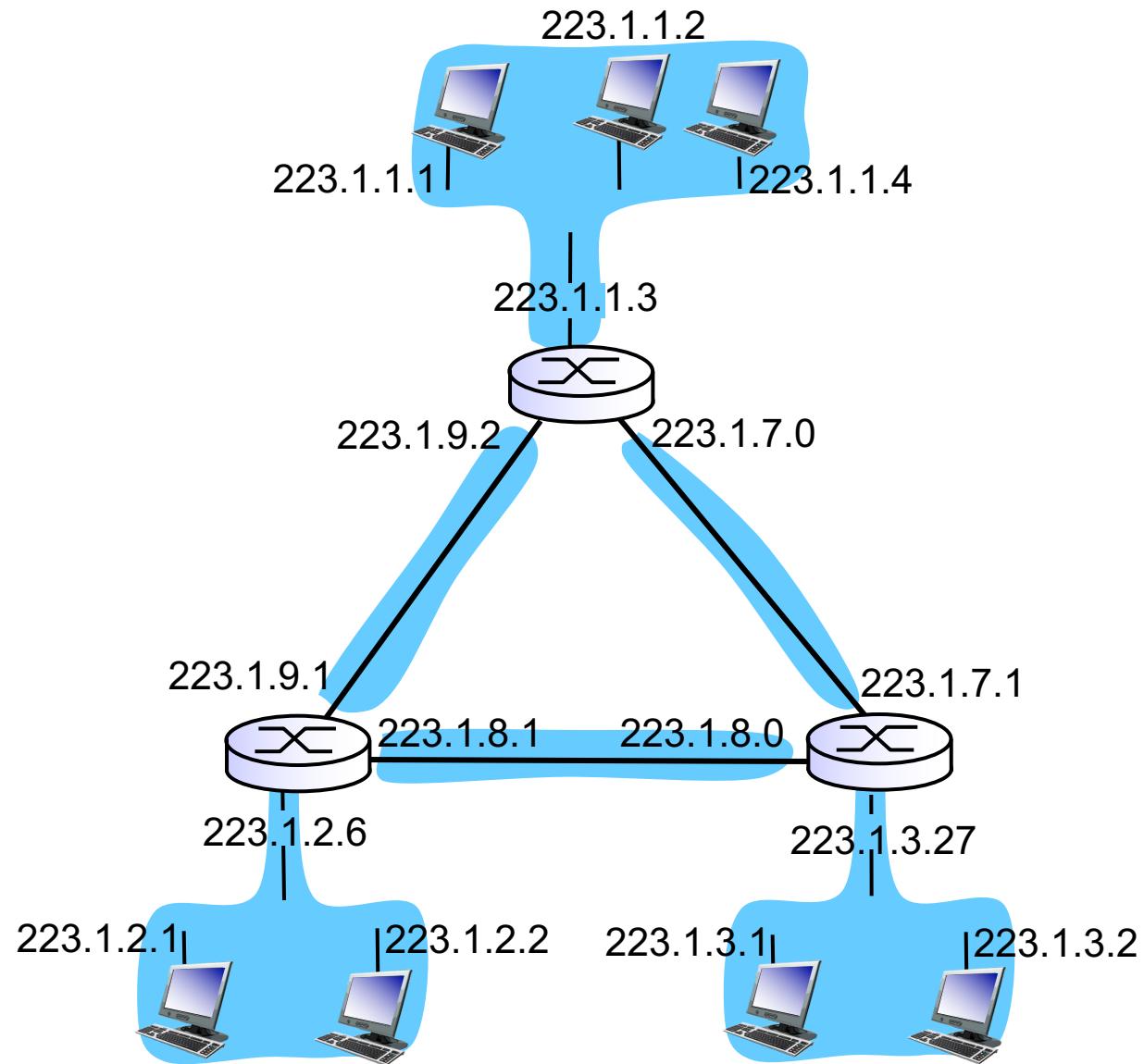
- to determine the subnets, detach each interface from its host or router, creating islands of isolated networks
- each isolated network is called a **subnet**



subnet mask: /24

Subnets

how many?



IP addressing: CIDR

CIDR: Classless InterDomain Routing

- subnet portion of address of arbitrary length
- address format: $a.b.c.d/x$, where x is # bits in subnet portion of address



IP addresses: how to get one?

Q: How does a *host* get IP address?

- hard-coded by system admin in a file
 - Windows: control-panel->network->configuration->tcp/ip->properties
 - UNIX: /etc/rc.config
- **DHCP: Dynamic Host Configuration Protocol:** dynamically get address from as server
 - “plug-and-play”

DHCP: Dynamic Host Configuration Protocol

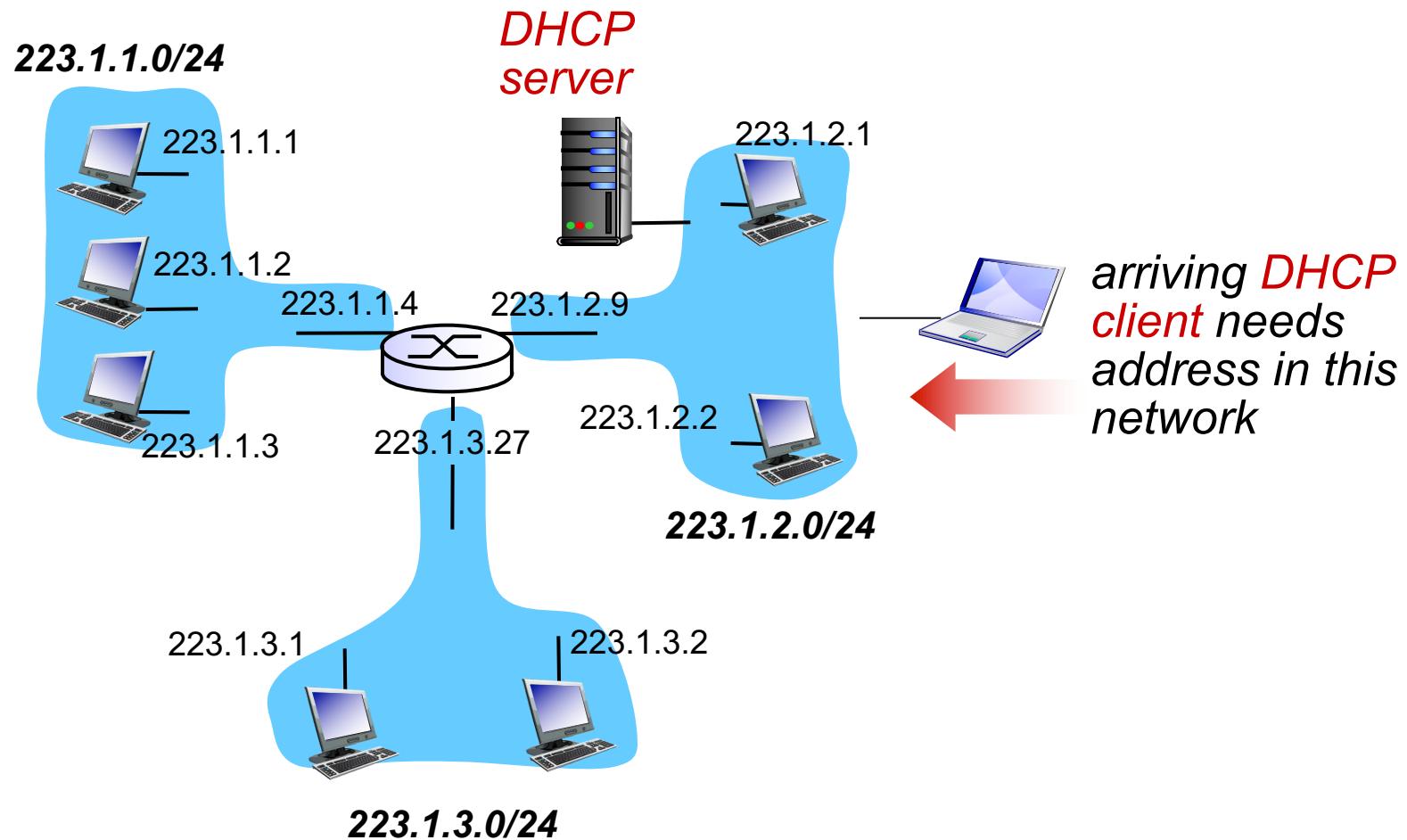
goal: allow host to *dynamically* obtain its IP address from network server when it joins network

- can renew its lease on address in use
- allows reuse of addresses (only hold address while connected/“on”)
- support for mobile users who want to join network (more shortly)

DHCP overview:

- host broadcasts “**DHCP discover**” msg [optional]
- DHCP server responds with “**DHCP offer**” msg [optional]
- host requests IP address: “**DHCP request**” msg
- DHCP server sends address: “**DHCP ack**” msg

DHCP client-server scenario



DHCP client-server scenario

DHCP server: 223.1.2.5



DHCP discover

Broadcast: is there a
DHCP server out there?

arriving
client



DHCP offer

Broadcast: I'm a DHCP
server! Here's an IP
address you can use

DHCP request

Broadcast: OK. I'll take
that IP address!

DHCP ACK

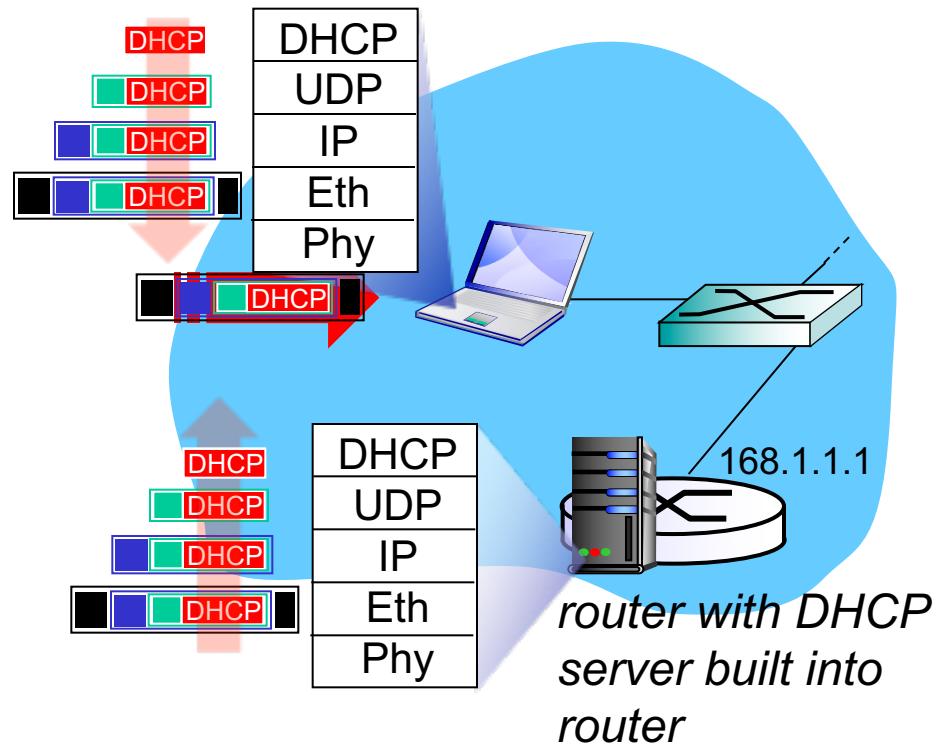
Broadcast: OK. You've
got that IP address!

DHCP: more than IP addresses

DHCP can return more than just allocated IP address on subnet:

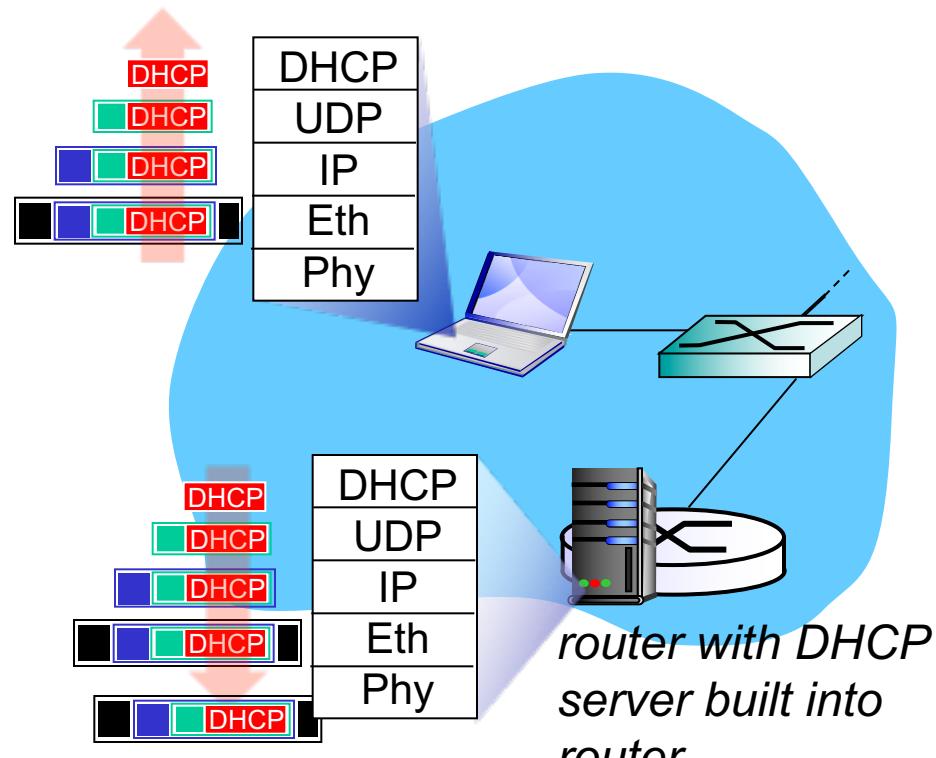
- address of first-hop router for client
- name and IP address of DNS sever
- network mask (indicating network versus host portion of address)

DHCP: example



- connecting laptop needs its IP address, addr of first-hop router, addr of DNS server: use DHCP
- DHCP request encapsulated in UDP, encapsulated in IP, encapsulated in 802.1 Ethernet
- Ethernet frame broadcast (dest: FFFFFFFFFFFF) on LAN, received at router running DHCP server
- Ethernet demuxed to IP demuxed, UDP demuxed to DHCP

DHCP: example



- DCP server formulates DHCP ACK containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server
- encapsulation of DHCP server, frame forwarded to client, demuxing up to DHCP at client
- client now knows its IP address, name and IP address of DSN server, IP address of its first-hop router

DHCP: Wireshark output (home LAN)

Message type: **Boot Request (1)**

Hardware type: Ethernet

Hardware address length: 6

Hops: 0

Transaction ID: 0x6b3a11b7

Seconds elapsed: 0

Bootp flags: 0x0000 (Unicast)

Client IP address: 0.0.0.0 (0.0.0.0)

Your (client) IP address: 0.0.0.0 (0.0.0.0)

Next server IP address: 0.0.0.0 (0.0.0.0)

Relay agent IP address: 0.0.0.0 (0.0.0.0)

Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)

Server host name not given

Boot file name not given

Magic cookie: (OK)

Option: (t=53,l=1) **DHCP Message Type = DHCP Request**

Option: (61) Client identifier

Length: 7; Value: 010016D323688A;

Hardware type: Ethernet

Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)

Option: (t=50,l=4) Requested IP Address = 192.168.1.101

Option: (t=12,l=5) Host Name = "nomad"

Option: (55) Parameter Request List

Length: 11; Value: 010F03062C2E2F1F21F92B

1 = Subnet Mask; 15 = Domain Name

3 = Router; 6 = Domain Name Server

44 = NetBIOS over TCP/IP Name Server

request

reply

Message type: **Boot Reply (2)**

Hardware type: Ethernet

Hardware address length: 6

Hops: 0

Transaction ID: 0x6b3a11b7

Seconds elapsed: 0

Bootp flags: 0x0000 (Unicast)

Client IP address: 192.168.1.101 (192.168.1.101)

Your (client) IP address: 0.0.0.0 (0.0.0.0)

Next server IP address: 192.168.1.1 (192.168.1.1)

Relay agent IP address: 0.0.0.0 (0.0.0.0)

Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)

Server host name not given

Boot file name not given

Magic cookie: (OK)

Option: (t=53,l=1) DHCP Message Type = DHCP ACK

Option: (t=54,l=4) Server Identifier = 192.168.1.1

Option: (t=1,l=4) Subnet Mask = 255.255.255.0

Option: (t=3,l=4) Router = 192.168.1.1

Option: (6) Domain Name Server

Length: 12; Value: 445747E2445749F244574092;

IP Address: 68.87.71.226;

IP Address: 68.87.73.242;

IP Address: 68.87.64.146

Option: (t=15,l=20) Domain Name = "hsd1.ma.comcast.net."

IP addresses: how to get one?

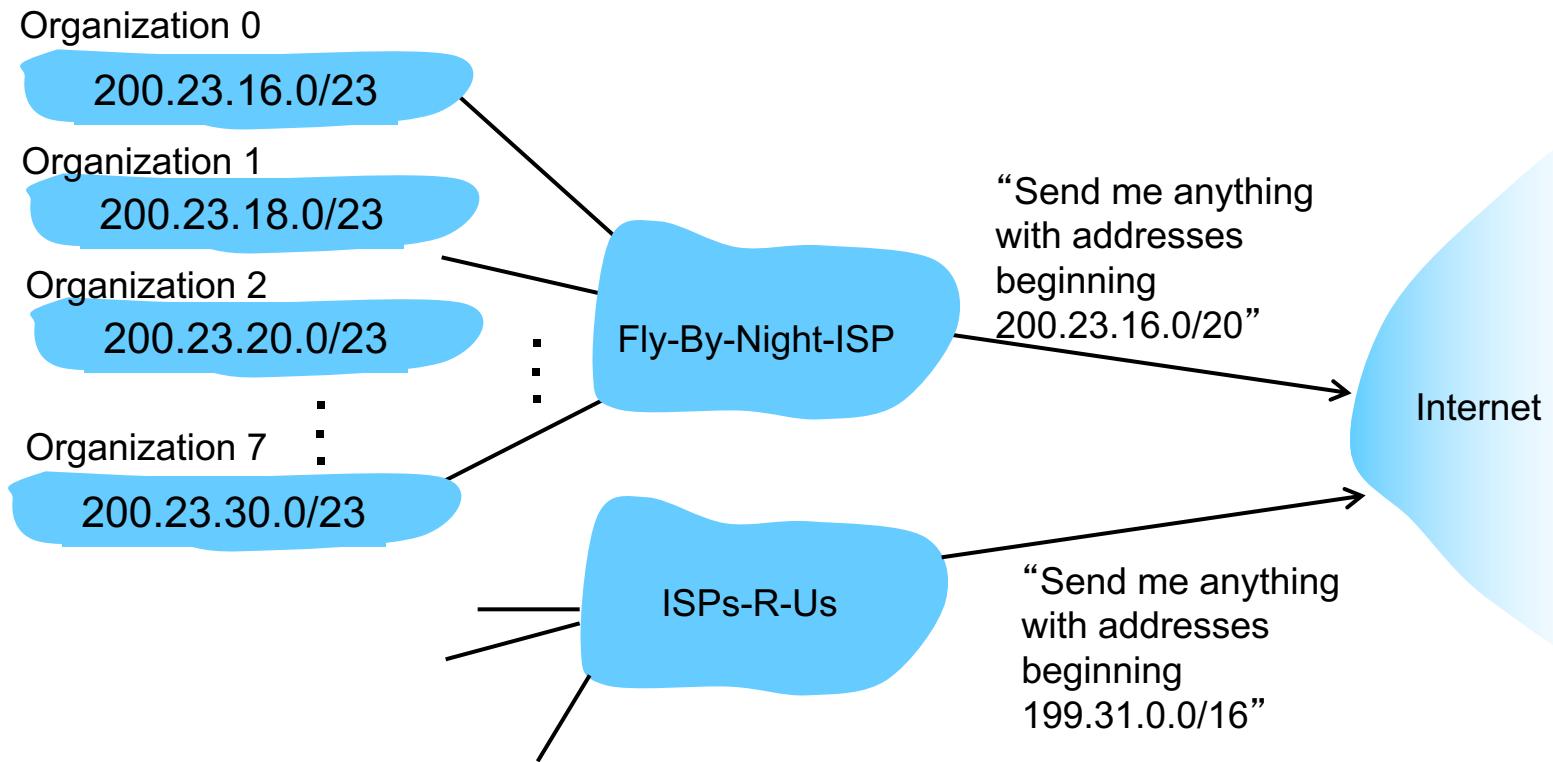
Q: how does *network* get subnet part of IP addr?

A: gets allocated portion of its provider ISP's address space

ISP's block	<u>11001000</u> <u>00010111</u> <u>00010000</u> <u>00000000</u>	200.23.16.0/20
Organization 0	<u>11001000</u> <u>00010111</u> <u>00010000</u> <u>00000000</u>	200.23.16.0/23
Organization 1	<u>11001000</u> <u>00010111</u> <u>00010010</u> <u>00000000</u>	200.23.18.0/23
Organization 2	<u>11001000</u> <u>00010111</u> <u>00010100</u> <u>00000000</u>	200.23.20.0/23
...
Organization 7	<u>11001000</u> <u>00010111</u> <u>00011110</u> <u>00000000</u>	200.23.30.0/23

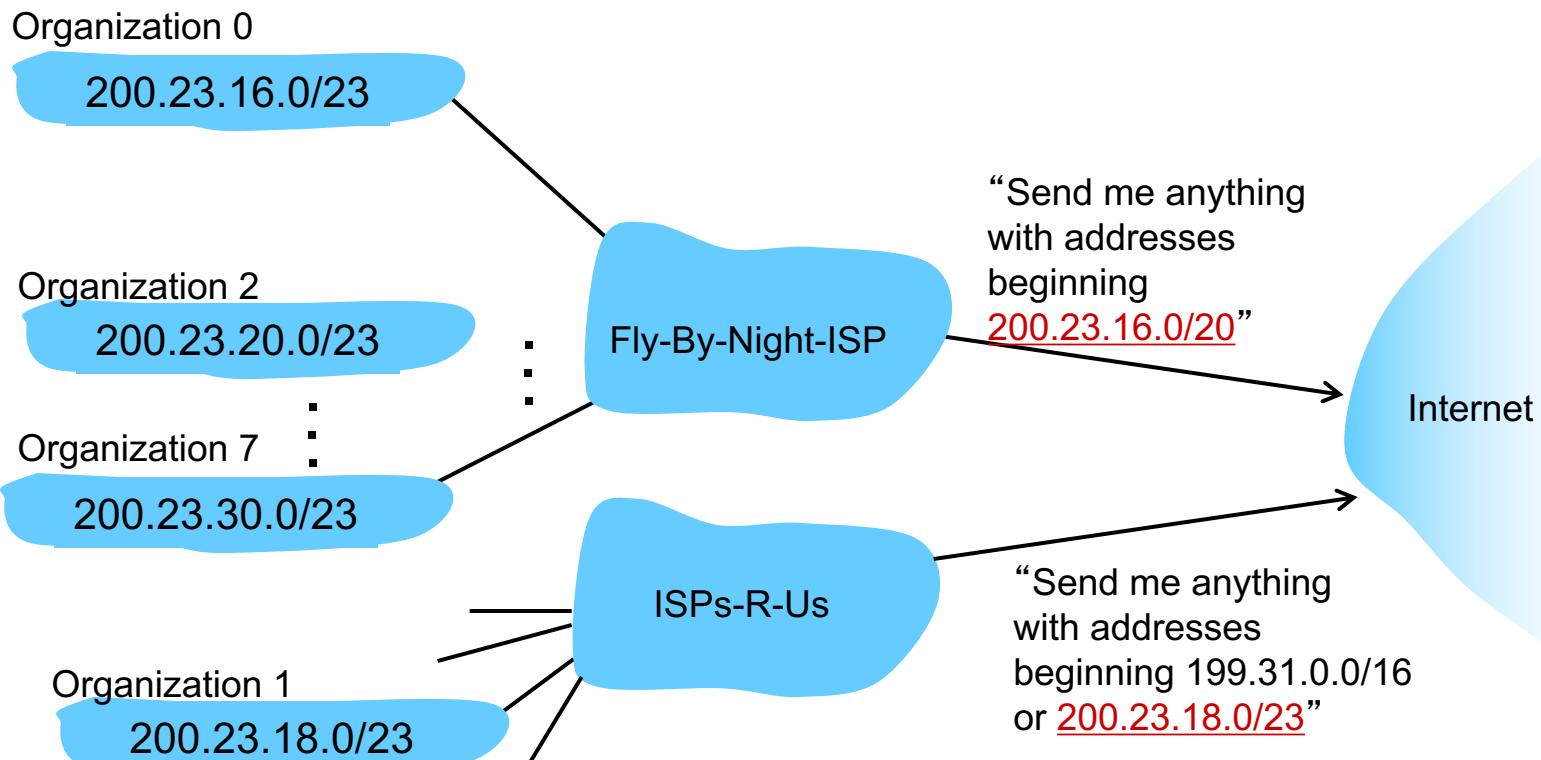
Hierarchical addressing: route aggregation

hierarchical addressing allows efficient advertisement of routing information:



Hierarchical addressing: more specific routes

ISPs-R-Us has a more specific route to Organization 1



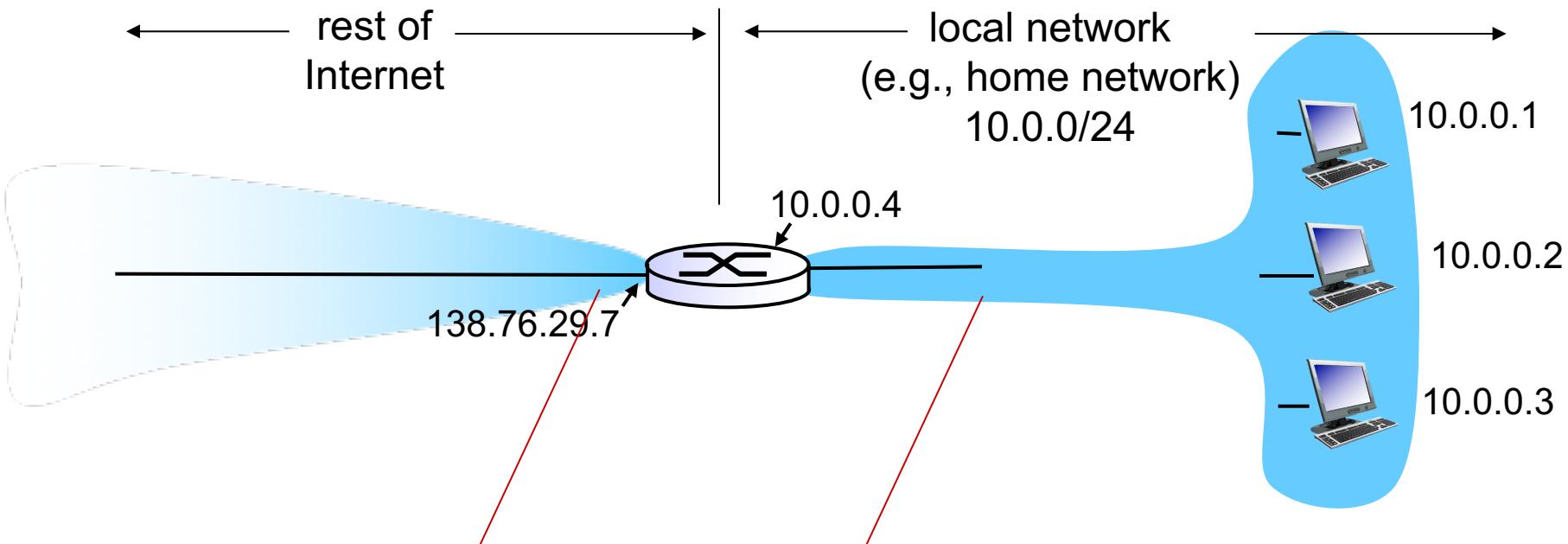
IP addressing: the last word...

Q: how does an ISP get block of addresses?

A: ICANN: Internet Corporation for Assigned Names and Numbers <http://www.icann.org/>

- allocates addresses
- manages DNS
- assigns domain names, resolves disputes

NAT: network address translation



all datagrams *leaving* local network have *same* single source NAT IP address:
138.76.29.7, different source port numbers

datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual)

NAT: network address translation

motivation: local network uses just one IP address as far as outside world is concerned:

- range of addresses not needed from ISP: just one IP address for all devices
- can change addresses of devices in local network without notifying outside world
- can change ISP without changing addresses of devices in local network
- devices inside local net not explicitly addressable, visible by outside world (a security plus)

NAT: network address translation

implementation: NAT router must:

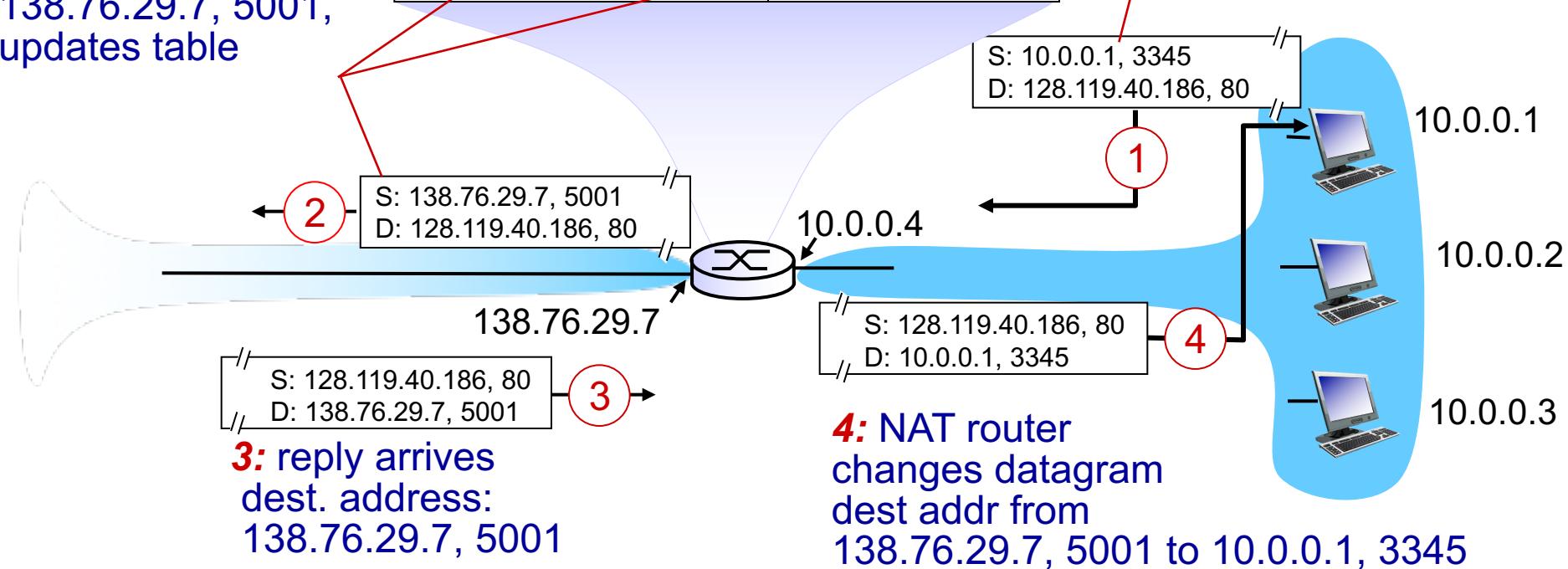
- *outgoing datagrams:* replace (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)
 . . . remote clients/servers will respond using (NAT IP address, new port #) as destination addr
- *remember (in NAT translation table)* every (source IP address, port #) to (NAT IP address, new port #) translation pair
- *incoming datagrams:* replace (NAT IP address, new port #) in dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

NAT: network address translation

2: NAT router changes datagram source addr from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table

NAT translation table	
WAN side addr	LAN side addr
138.76.29.7, 5001	10.0.0.1, 3345
.....

1: host 10.0.0.1 sends datagram to 128.119.40.186, 80



* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

NAT: network address translation

- 16-bit port-number field:
 - 60,000 simultaneous connections with a single LAN-side address!
- NAT is controversial:
 - routers should only process up to layer 3
 - address shortage should be solved by IPv6
 - violates end-to-end argument
 - NAT possibility must be taken into account by app designers, e.g., P2P applications
 - NAT traversal: what if client wants to connect to server behind NAT?

Chapter 4: outline

4.1 Overview of Network layer

- data plane
- control plane

4.2 What's inside a router

4.3 IP: Internet Protocol

- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- **IPv6**

4.4 Generalized Forward and SDN

- match
- action
- OpenFlow examples of match-plus-action in action

IPv6: motivation

- *initial motivation:* 32-bit address space soon to be completely allocated.
- additional motivation:
 - header format helps speed processing/forwarding
 - header changes to facilitate QoS

IPv6 datagram format:

- fixed-length 40 byte header
- no fragmentation allowed

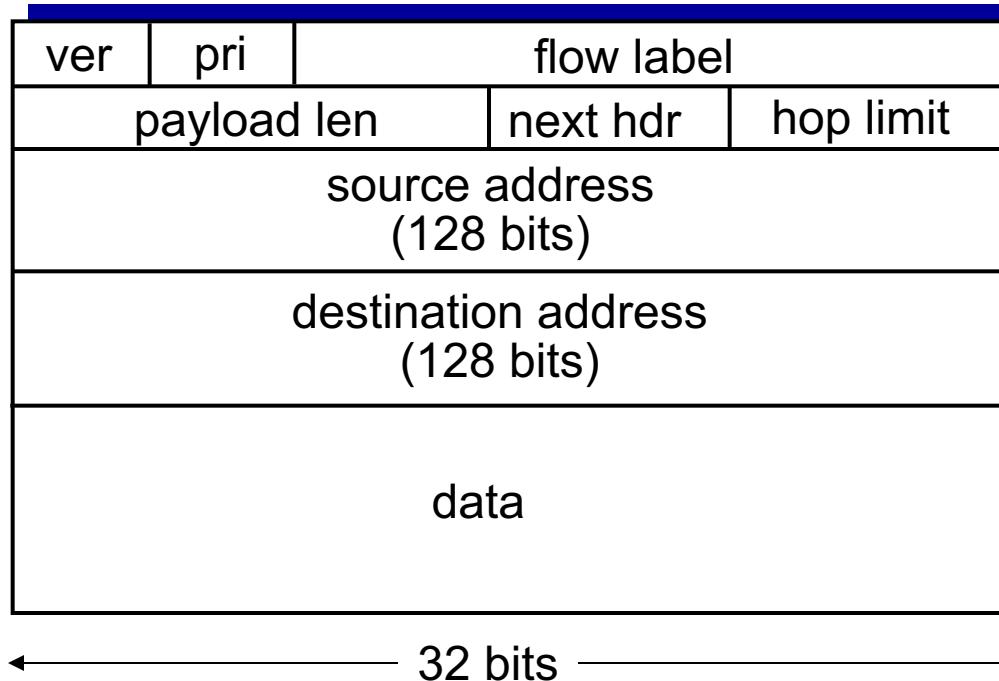
IPv6 datagram format

priority: identify priority among datagrams in flow

flow Label: identify datagrams in same “flow.”

(concept of “flow” not well defined).

next header: identify upper layer protocol for data

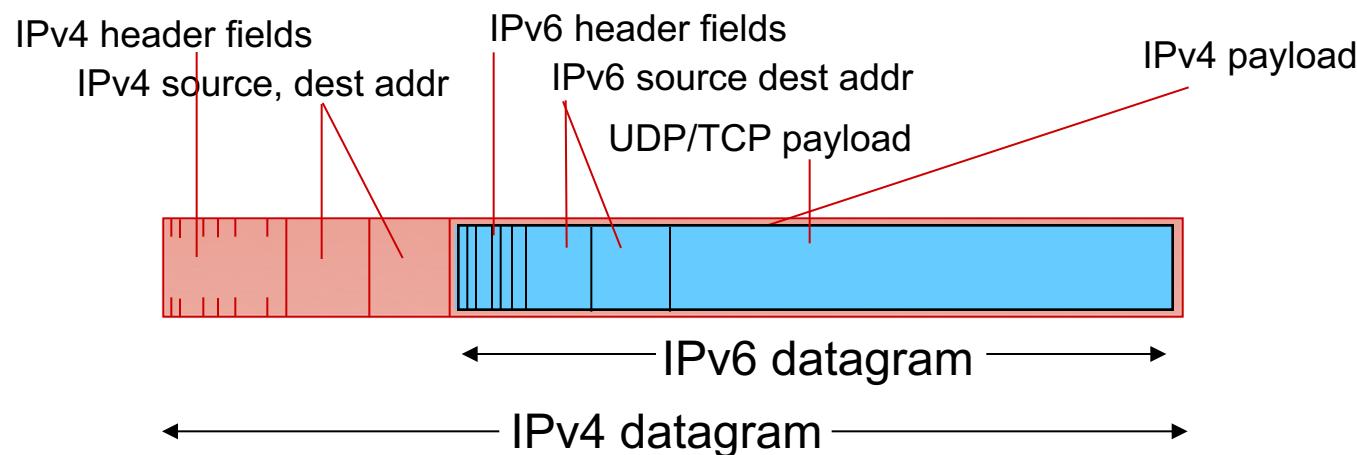


Other changes from IPv4

- *checksum*: removed entirely to reduce processing time at each hop
- *options*: allowed, but outside of header, indicated by “Next Header” field
- *ICMPv6*: new version of ICMP
 - additional message types, e.g. “Packet Too Big”
 - multicast group management functions

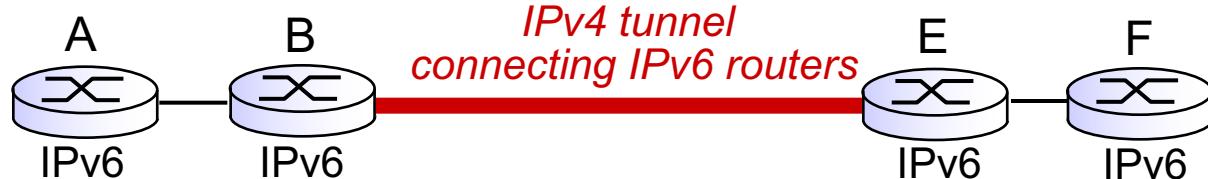
Transition from IPv4 to IPv6

- not all routers can be upgraded simultaneously
 - no “flag days”
 - how will network operate with mixed IPv4 and IPv6 routers?
- **tunneling:** IPv6 datagram carried as *payload* in IPv4 datagram among IPv4 routers

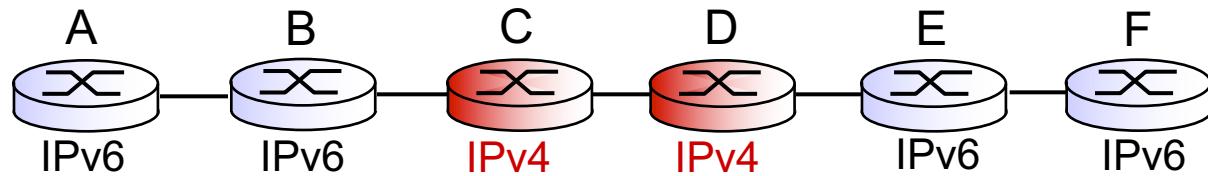


Tunneling

logical view:

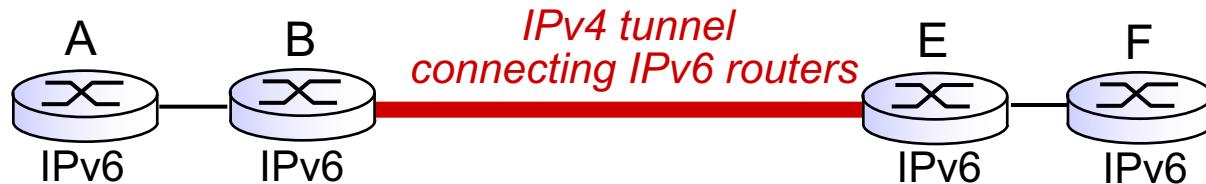


physical view:

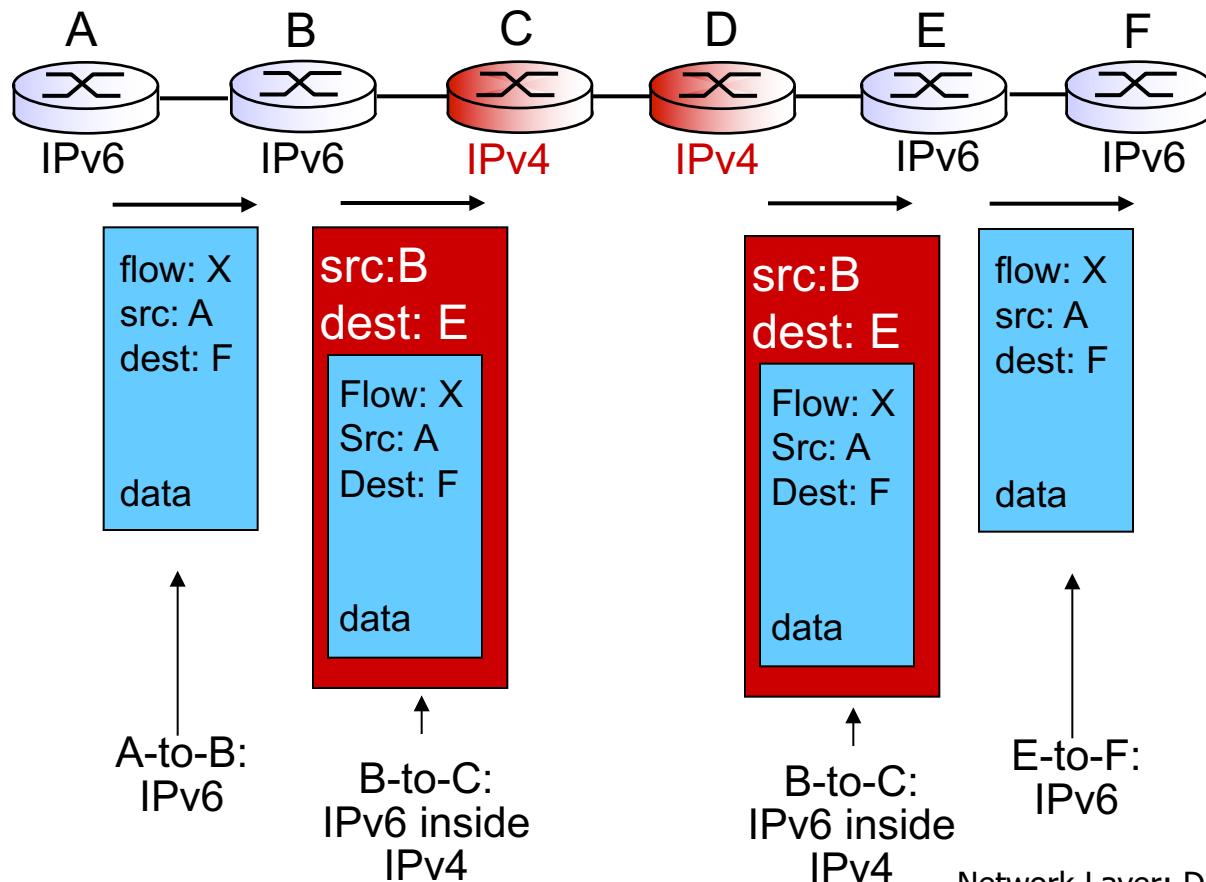


Tunneling

logical view:



physical view:



IPv6: adoption

- Google: 8% of clients access services via IPv6
- NIST: 1/3 of all US government domains are IPv6 capable
- *Long (long!) time for deployment, use*
 - 20 years and counting!
 - think of application-level changes in last 20 years: WWW, Facebook, streaming media, Skype, ...
 - *Why?*

Chapter 4: outline

4.1 Overview of Network layer

- data plane
- control plane

4.2 What's inside a router

4.3 IP: Internet Protocol

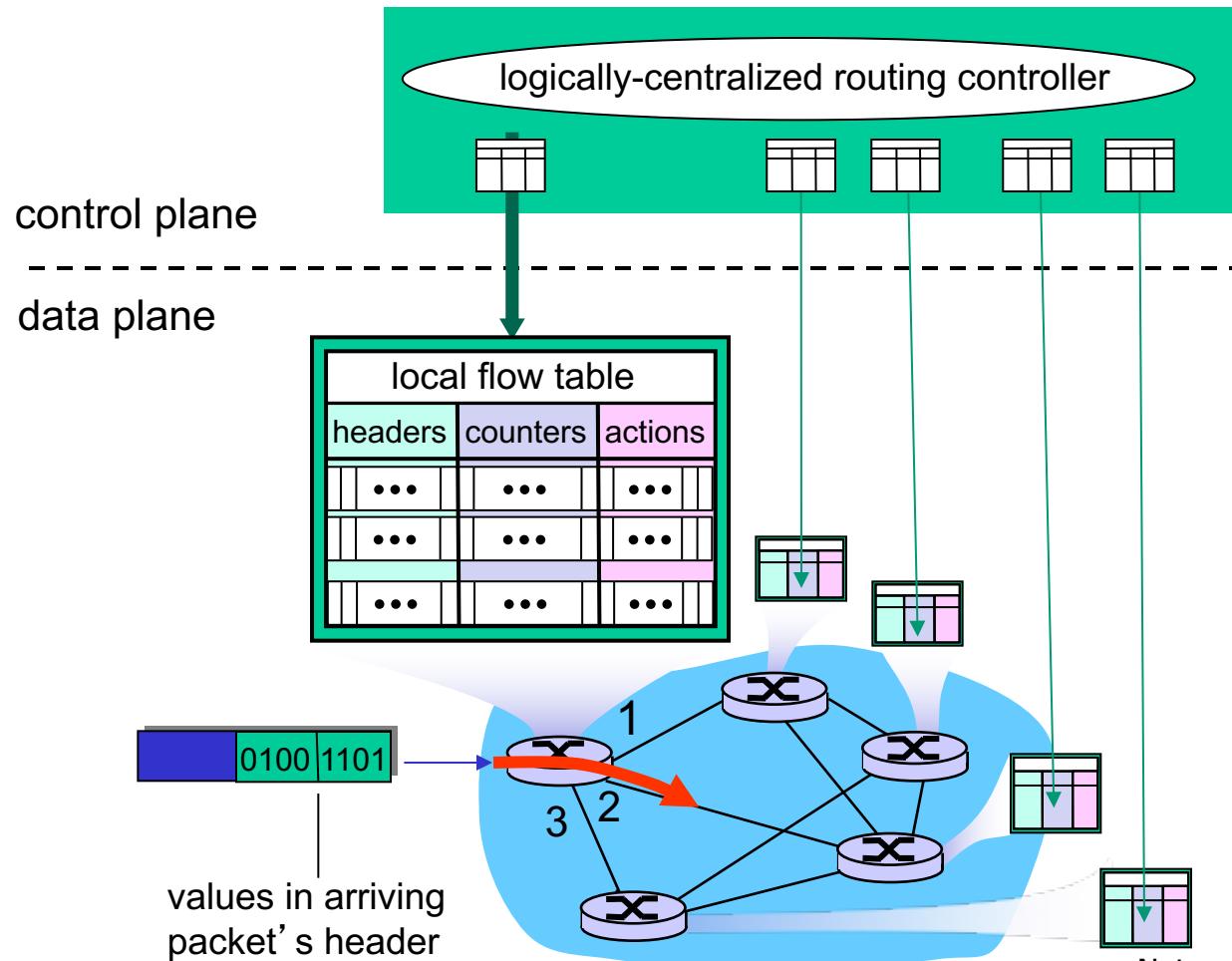
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

4.4 Generalized Forward and SDN

- match
- action
- OpenFlow examples of match-plus-action in action

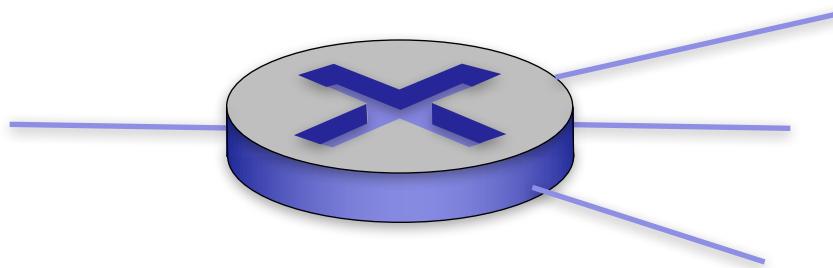
Generalized Forwarding and SDN

Each router contains a *flow table* that is computed and distributed by a *logically centralized routing controller*



OpenFlow data plane abstraction

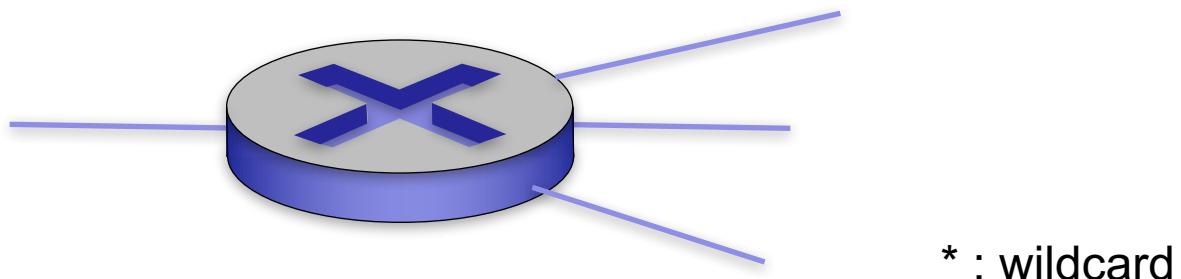
- *flow*: defined by header fields
- generalized forwarding: simple packet-handling rules
 - *Pattern*: match values in packet header fields
 - *Actions*: *for matched packet*: drop, forward, modify, matched packet or send matched packet to controller
 - *Priority*: disambiguate overlapping patterns
 - *Counters*: #bytes and #packets



Flow table in a router (computed and distributed by controller) define router's match+action rules

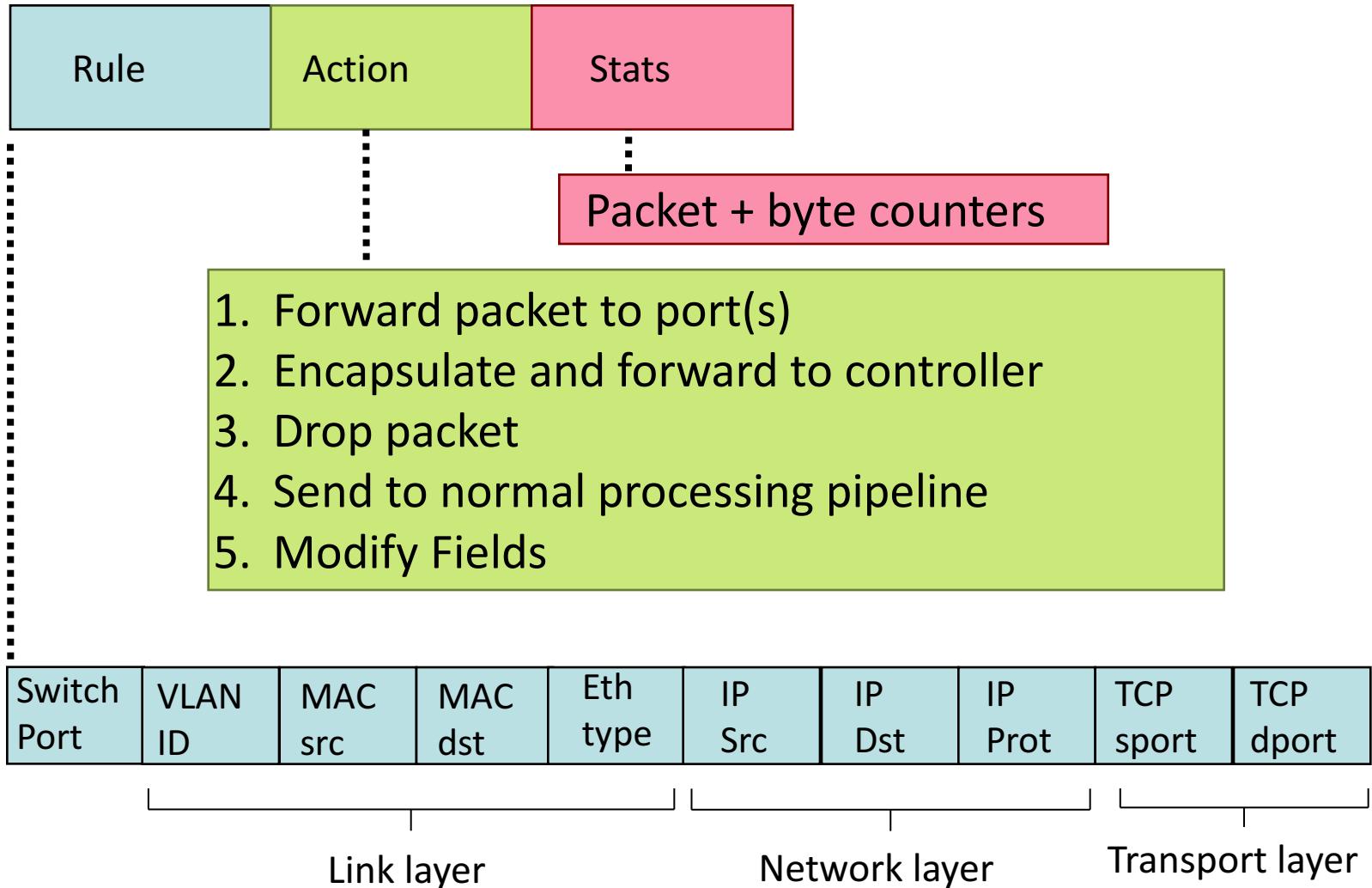
OpenFlow data plane abstraction

- *flow*: defined by header fields
- generalized forwarding: simple packet-handling rules
 - *Pattern*: match values in packet header fields
 - *Actions*: *for matched packet*: drop, forward, modify, matched packet or send matched packet to controller
 - *Priority*: disambiguate overlapping patterns
 - *Counters*: #bytes and #packets



1. $\text{src}=1.2.*.*$, $\text{dest}=3.4.5.* \rightarrow \text{drop}$
2. $\text{src} = *.*.*.*$, $\text{dest}=3.4.*.* \rightarrow \text{forward}(2)$
3. $\text{src}=10.1.2.3$, $\text{dest} = *.*.*.* \rightarrow \text{send to controller}$

OpenFlow: Flow Table Entries



Examples

Destination-based forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	51.6.0.8	*	*	*	port6

IP datagrams destined to IP address 51.6.0.8 should be forwarded to router output port 6

Firewall:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	*	*	*	22	drop

do not forward (block) all datagrams destined to TCP port 22

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	128.119.1.1	*	*	*	*	drop

do not forward (block) all datagrams sent by host 128.119.1.1

Examples

Destination-based layer 2 (switch) forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	22:A7:23: 11:E1:02	*	*	*	*	*	*	*	*	port3

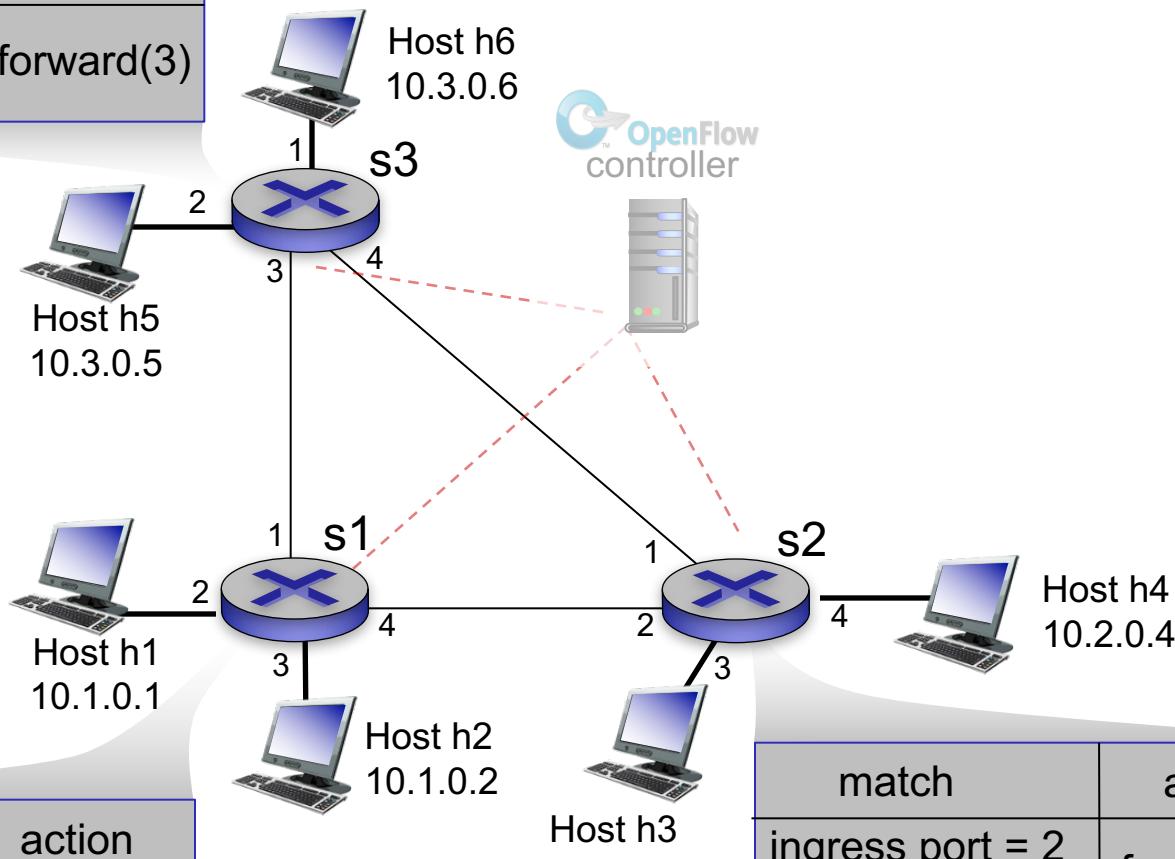
*layer 2 frames from MAC address 22:A7:23:11:E1:02
should be forwarded to output port 6*

OpenFlow abstraction

- *match+action*: unifies different kinds of devices
- Router
 - *match*: longest destination IP prefix
 - *action*: forward out a link
- Switch
 - *match*: destination MAC address
 - *action*: forward or flood
- Firewall
 - *match*: IP addresses and TCP/UDP port numbers
 - *action*: permit or deny
- NAT
 - *match*: IP address and port
 - *action*: rewrite address and port

OpenFlow example

match	action
IP Src = 10.3.*.*	
IP Dst = 10.2.*.*	forward(3)



Example: datagrams from hosts h5 and h6 should be sent to h3 or h4, via s1 and from there to s2

match	action
ingress port = 1 IP Src = 10.3.*.* IP Dst = 10.2.*.*	forward(4)

match	action
ingress port = 2 IP Dst = 10.2.0.3	forward(3)
ingress port = 2 IP Dst = 10.2.0.4	forward(4)

Chapter 4: done!

4.1 Overview of Network layer: data plane and control plane

4.2 What's inside a router

4.3 IP: Internet Protocol

- datagram format
- fragmentation
- IPv4 addressing
- NAT
- IPv6

4.4 Generalized Forward and SDN

- match plus action
- OpenFlow example

Question: how do forwarding tables (destination-based forwarding) or flow tables (generalized forwarding) computed?

Answer: by the control plane (next chapter)

Chapter 5

Network Layer: The Control Plane

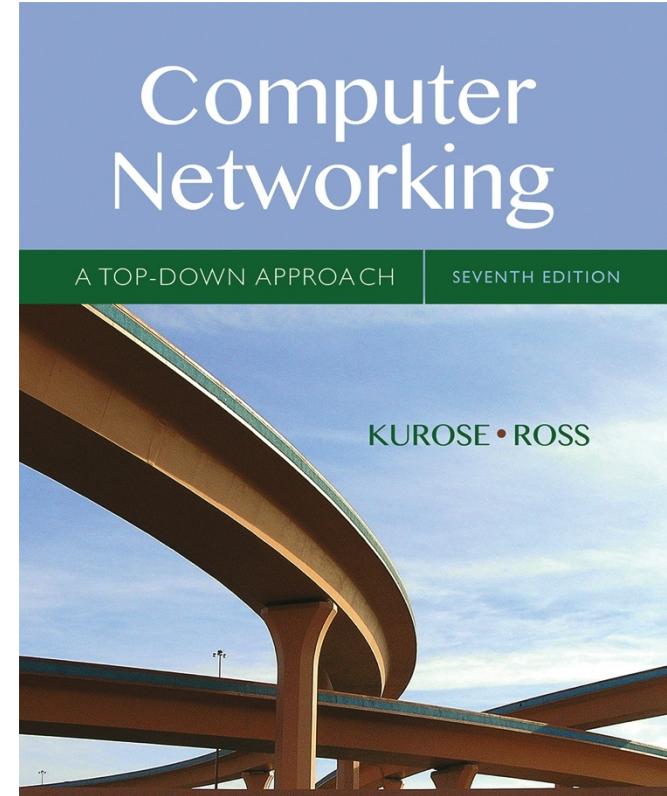
A note on the use of these Powerpoint slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR

© All material copyright 1996-2016
J.F Kurose and K.W. Ross, All Rights Reserved



*Computer
Networking: A Top
Down Approach*

7th edition

Jim Kurose, Keith Ross
Pearson/Addison Wesley
April 2016

Chapter 5: network layer control plane

chapter goals: understand principles behind network control plane

- traditional routing algorithms
- SDN controllers
- Internet Control Message Protocol
- network management

and their instantiation, implementation in the Internet:

- OSPF, BGP, OpenFlow, ODL and ONOS controllers, ICMP, SNMP

Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs:
BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

Network-layer functions

Recall: two network-layer functions:

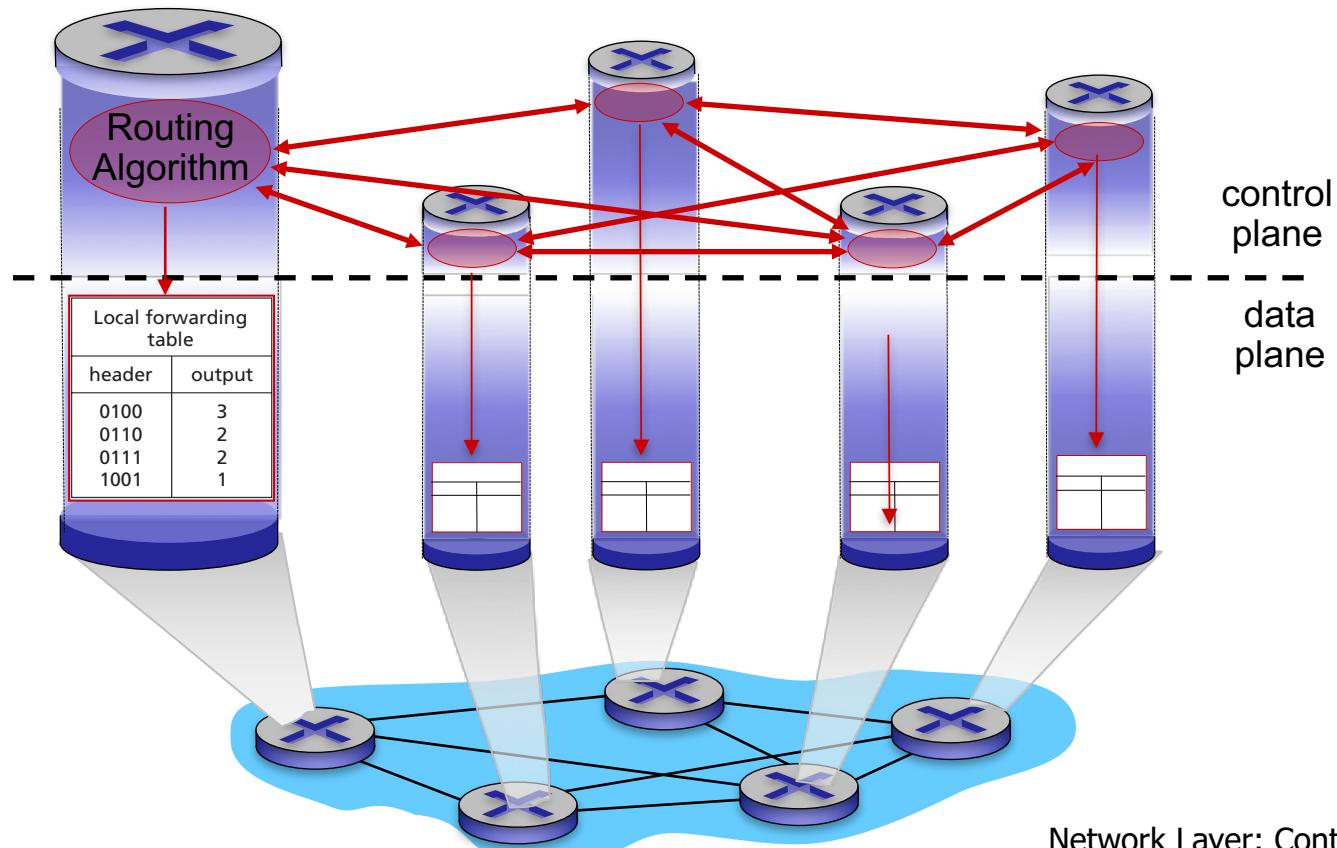
- **forwarding:** move packets from router's input to appropriate router output *data plane*
- **routing:** determine route taken by packets from source to destination *control plane*

Two approaches to structuring network control plane:

- per-router control (traditional)
- logically centralized control (software defined networking)

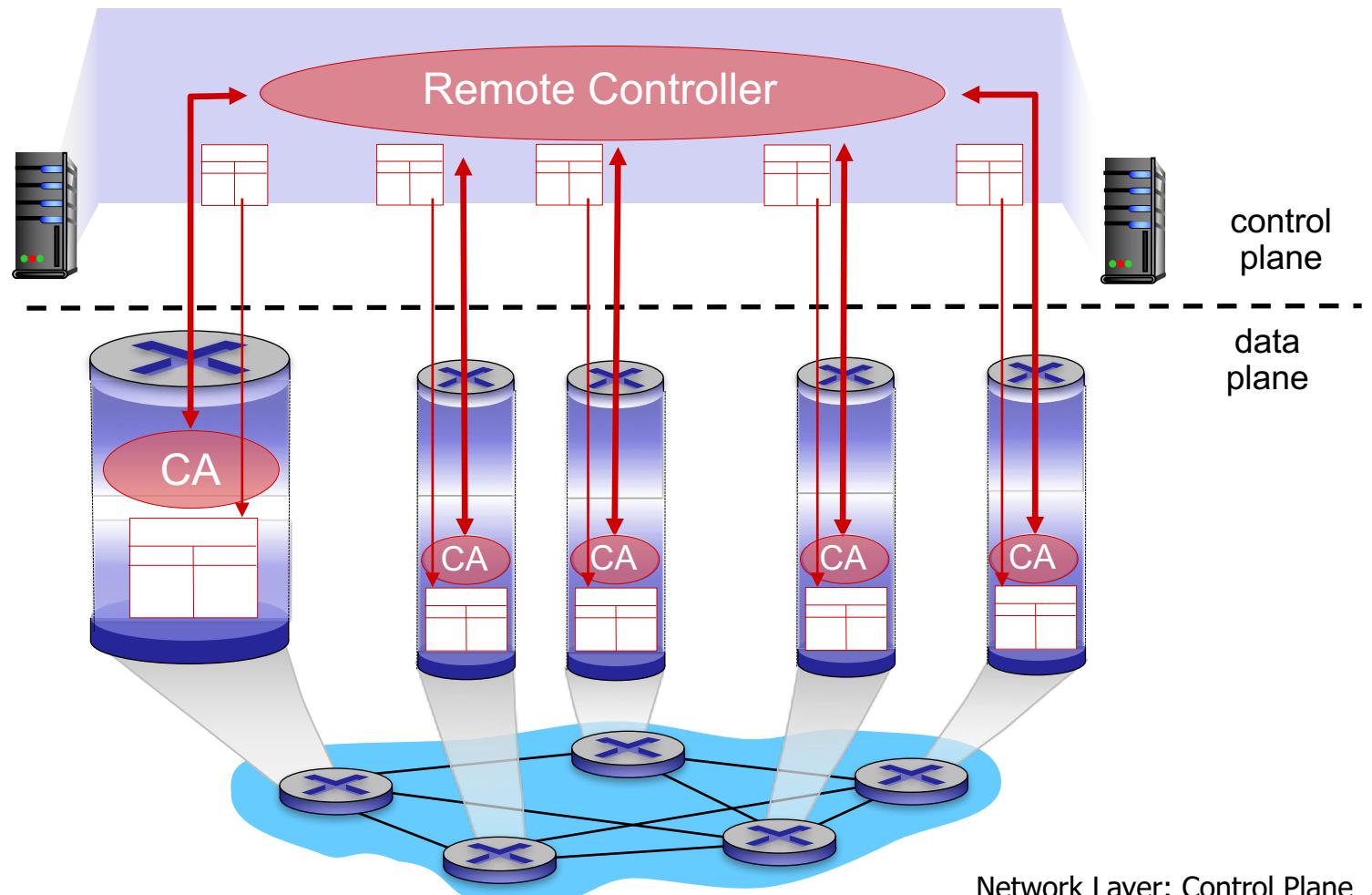
Per-router control plane

Individual routing algorithm components *in each and every router* interact with each other in control plane to compute forwarding tables



Logically centralized control plane

A distinct (typically remote) controller interacts with local control agents (CAs) in routers to compute forwarding tables



Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs:
BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

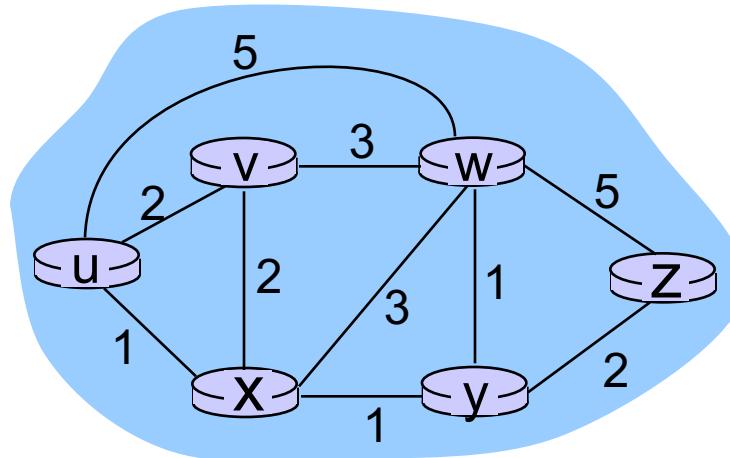
5.7 Network management and SNMP

Routing protocols

Routing protocol goal: determine “good” paths (equivalently, routes), from sending hosts to receiving host, through network of routers

- path: sequence of routers packets will traverse in going from given initial source host to given final destination host
- “good”: least “cost”, “fastest”, “least congested”
- routing: a “top-10” networking challenge!

Graph abstraction of the network



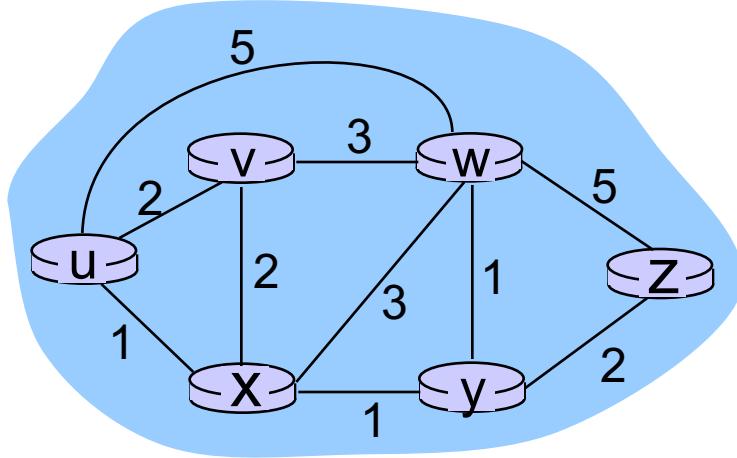
graph: $G = (N, E)$

$N = \text{set of routers} = \{ u, v, w, x, y, z \}$

$E = \text{set of links} = \{ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) \}$

aside: graph abstraction is useful in other network contexts, e.g., P2P, where N is set of peers and E is set of TCP connections

Graph abstraction: costs



$c(x,x')$ = cost of link (x,x')
e.g., $c(w,z) = 5$

cost could always be 1, or
inversely related to bandwidth,
or inversely related to
congestion

cost of path $(x_1, x_2, x_3, \dots, x_p) = c(x_1, x_2) + c(x_2, x_3) + \dots + c(x_{p-1}, x_p)$

key question: what is the least-cost path between u and z ?
routing algorithm: algorithm that finds that least cost path

Routing algorithm classification

Q: global or decentralized information?

global:

- all routers have complete topology, link cost info
- “link state” algorithms

decentralized:

- router knows physically-connected neighbors, link costs to neighbors
- iterative process of computation, exchange of info with neighbors
- “distance vector” algorithms

Q: static or dynamic?

static:

- routes change slowly over time

dynamic:

- routes change more quickly
 - periodic update
 - in response to link cost changes

Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state

- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs:
BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

A link-state routing algorithm

Dijkstra's algorithm

- net topology, link costs known to all nodes
 - accomplished via “link state broadcast”
 - all nodes have same info
- computes least cost paths from one node (‘source’) to all other nodes
 - gives *forwarding table* for that node
- iterative: after k iterations, know least cost path to k dest.’s

notation:

- $c(x,y)$: link cost from node x to y ; $= \infty$ if not direct neighbors
- $D(v)$: current value of cost of path from source to dest. v
- $p(v)$: predecessor node along path from source to v
- N' : set of nodes whose least cost path definitively known

Dijkstra's algorithm

1 **Initialization:**

2 $N' = \{u\}$

3 for all nodes v

4 if v adjacent to u

5 then $D(v) = c(u,v)$

6 else $D(v) = \infty$

7

8 **Loop**

9 find w not in N' such that $D(w)$ is a minimum

10 add w to N'

11 update $D(v)$ for all v adjacent to w and not in N' :

12 **$D(v) = \min(D(v), D(w) + c(w,v))$**

13 /* new cost to v is either old cost to v or known

14 shortest path cost to w plus cost from w to v */

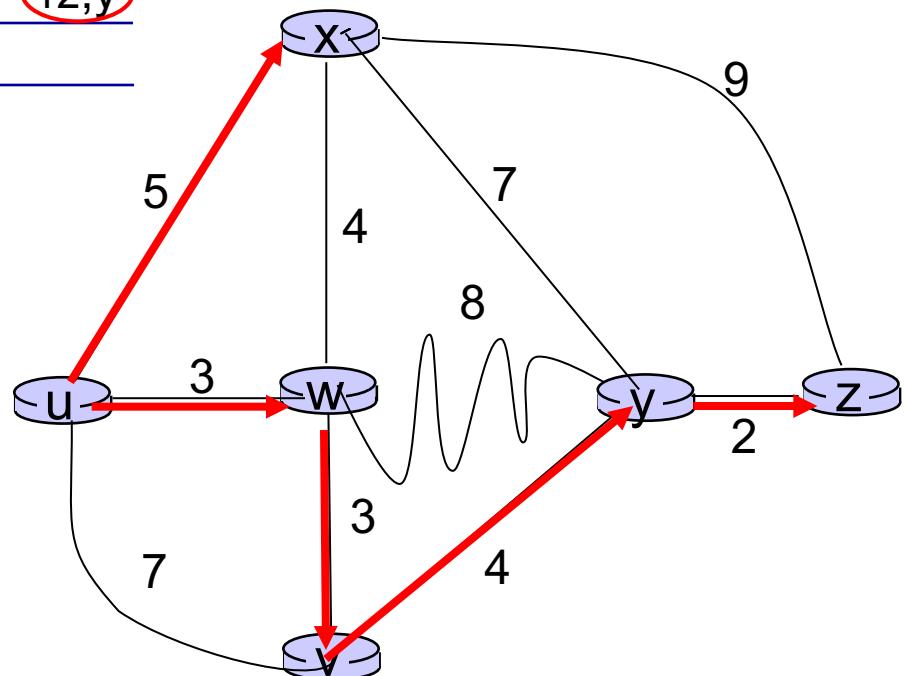
15 **until all nodes in N'**

Dijkstra's algorithm: example

Step	N'	D(v)	D(w)	D(x)	D(y)	D(z)
		p(v)	p(w)	p(x)	p(y)	p(z)
0	u	7,u	3,u	5,u	∞	∞
1	uw	6,w	5,u	11,w	∞	
2	uwx	6,w		11,w	14,x	
3	uwxv		10,v	14,x		
4	uwxvy			12,y		
5	uwxvzy					

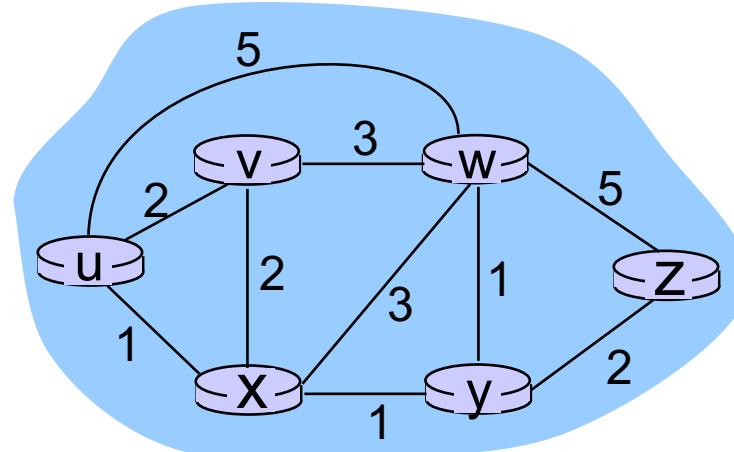
notes:

- ❖ construct shortest path tree by tracing predecessor nodes
- ❖ ties can exist (can be broken arbitrarily)



Dijkstra's algorithm: another example

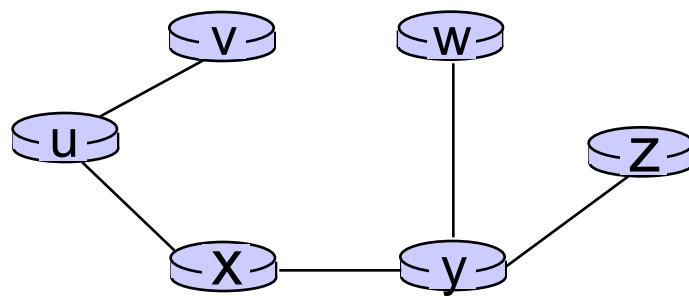
Step	N'	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

Dijkstra's algorithm: example (2)

resulting shortest-path tree from u:



resulting forwarding table in u:

destination	link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

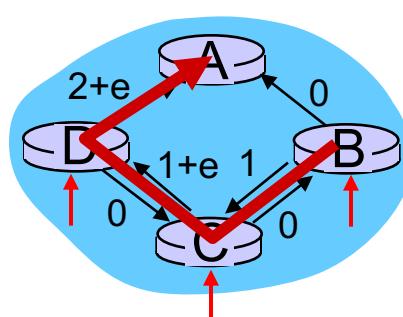
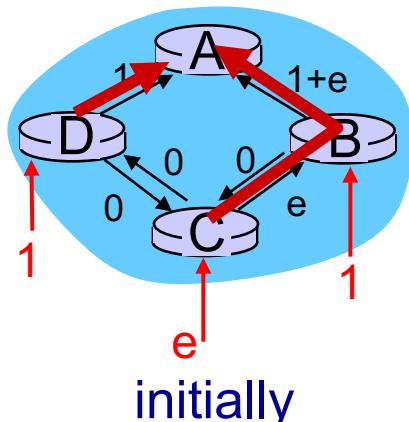
Dijkstra's algorithm, discussion

algorithm complexity: n nodes

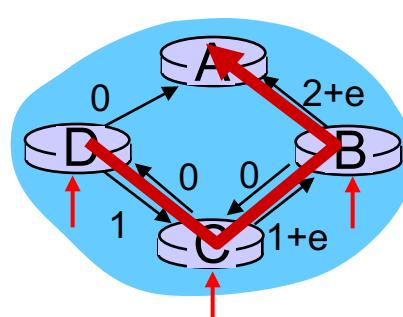
- each iteration: need to check all nodes, w, not in N
- $n(n+1)/2$ comparisons: $O(n^2)$
- more efficient implementations possible: $O(n \log n)$

oscillations possible:

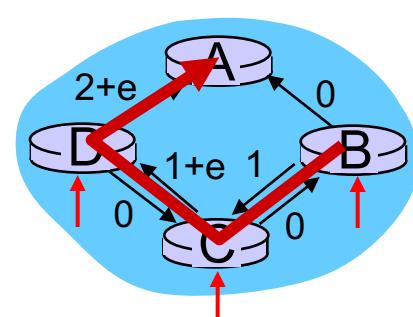
- e.g., support link cost equals amount of carried traffic:



given these costs,
find new routing....
resulting in new costs



given these costs,
find new routing....
resulting in new costs



given these costs,
find new routing....
resulting in new costs

Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- **distance vector**

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs:
BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

Distance vector algorithm

Bellman-Ford equation (dynamic programming)

let

$d_x(y) := \text{cost of least-cost path from } x \text{ to } y$

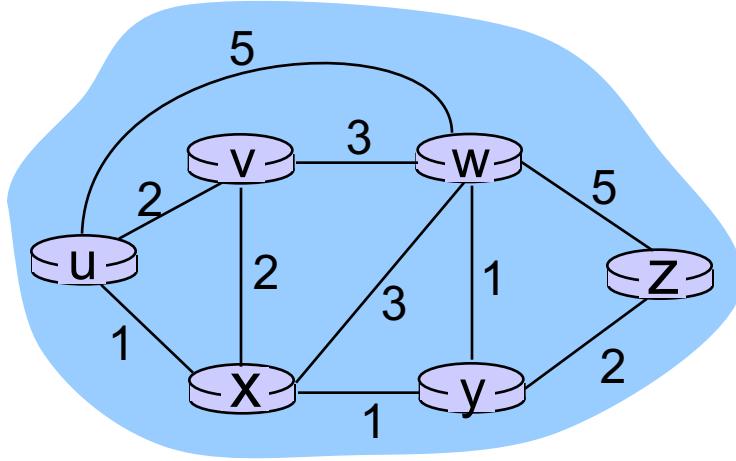
then

$$d_x(y) = \min_v \{ c(x,v) + d_v(y) \}$$

 v cost from neighbor v to destination y
cost to neighbor v

 \min taken over all neighbors v of x

Bellman-Ford example



clearly, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

B-F equation says:

$$\begin{aligned}d_u(z) &= \min \{ c(u,v) + d_v(z), \\&\quad c(u,x) + d_x(z), \\&\quad c(u,w) + d_w(z) \} \\&= \min \{ 2 + 5, \\&\quad 1 + 3, \\&\quad 5 + 3 \} = 4\end{aligned}$$

node achieving minimum is next
hop in shortest path, used in forwarding table

Distance vector algorithm

- $D_x(y)$ = estimate of least cost from x to y
 - x maintains distance vector $\mathbf{D}_x = [D_x(y): y \in N]$
- node x :
 - knows cost to each neighbor v : $c(x,v)$
 - maintains its neighbors' distance vectors. For each neighbor v , x maintains
 $\mathbf{D}_v = [D_v(y): y \in N]$

Distance vector algorithm

key idea:

- from time-to-time, each node sends its own distance vector estimate to neighbors
- when x receives new DV estimate from neighbor, it updates its own DV using B-F equation:

$$D_x(y) \leftarrow \min_v \{c(x,v) + D_v(y)\} \text{ for each node } y \in N$$

- ❖ under minor, natural conditions, the estimate $D_x(y)$ converge to the actual least cost $d_x(y)$

Distance vector algorithm

iterative, asynchronous:

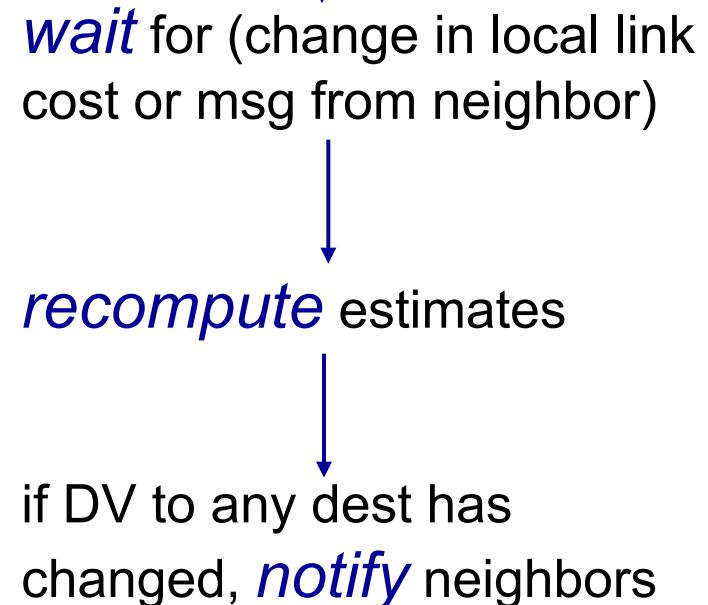
each local iteration
caused by:

- local link cost change
- DV update message from neighbor

distributed:

- each node notifies neighbors *only* when its DV changes
 - neighbors then notify their neighbors if necessary

each node:



$$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$$

$$= \min\{2+0, 7+1\} = 2$$

$$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\}$$

$$= \min\{2+1, 7+0\} = 3$$

**node x
table**

	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

**node y
table**

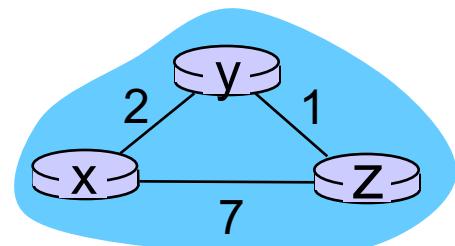
	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

**node z
table**

	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

time



$$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$$

$$= \min\{2+0, 7+1\} = 2$$

$$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\}$$

$$= \min\{2+1, 7+0\} = 3$$

**node x
table**

	cost to		
	x	y	z
from	x	0 2 7	
y	∞ ∞ ∞		
z	∞ ∞ ∞		

**node y
table**

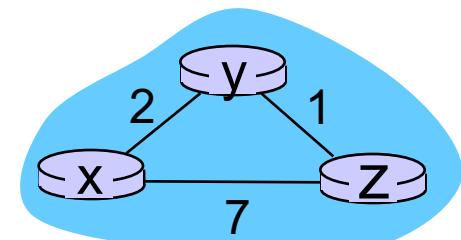
	cost to		
	x	y	z
from	x	∞ ∞ ∞	
y	2 0 1		
z	∞ ∞ ∞		

**node z
table**

	cost to		
	x	y	z
from	x	∞ ∞ ∞	
y	∞ ∞ ∞		
z	7 1 0		

	cost to		
	x	y	z
from	x	0 2 3	
y	2 0 1		
z	7 1 0		

	cost to		
	x	y	z
from	x	0 2 3	
y	2 0 1		
z	3 1 0		

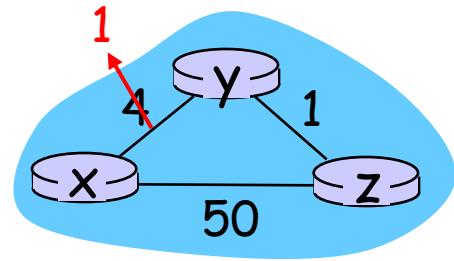


→ time

Distance vector: link cost changes

link cost changes:

- ❖ node detects local link cost change
- ❖ updates routing info, recalculates distance vector
- ❖ if DV changes, notify neighbors



**“good
news
travels
fast”**

t_0 : y detects link-cost change, updates its DV, informs its neighbors.

t_1 : z receives update from y, updates its table, computes new least cost to x , sends its neighbors its DV.

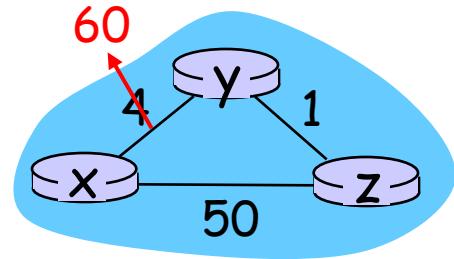
t_2 : y receives z' s update, updates its distance table. y' s least costs do *not* change, so y does *not* send a message to z.

* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

Distance vector: link cost changes

link cost changes:

- ❖ node detects local link cost change
- ❖ *bad news travels slow* - “count to infinity” problem!
- ❖ 44 iterations before algorithm stabilizes: see text



poisoned reverse:

- ❖ If Z routes through Y to get to X :
 - Z tells Y its (Z's) distance to X is infinite (so Y won't route to X via Z)
- ❖ will this completely solve count to infinity problem?

Comparison of LS and DV algorithms

message complexity

- **LS:** with n nodes, E links, $O(nE)$ msgs sent
- **DV:** exchange between neighbors only
 - convergence time varies

speed of convergence

- **LS:** $O(n^2)$ algorithm requires $O(nE)$ msgs
 - may have oscillations
- **DV:** convergence time varies
 - may be routing loops
 - count-to-infinity problem

robustness: what happens if router malfunctions?

LS:

- node can advertise incorrect *link* cost
- each node computes only its own table

DV:

- DV node can advertise incorrect *path* cost
- each node's table used by others
 - error propagate thru network

Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs:
BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

Making routing scalable

our routing study thus far - idealized

- all routers identical
 - network “flat”
- ... *not true in practice*

scale: with billions of destinations:

- can't store all destinations in routing tables!
- routing table exchange would swamp links!

administrative autonomy

- internet = network of networks
- each network admin may want to control routing in its own network

Internet approach to scalable routing

aggregate routers into regions known as “autonomous systems” (AS) (a.k.a. “domains”)

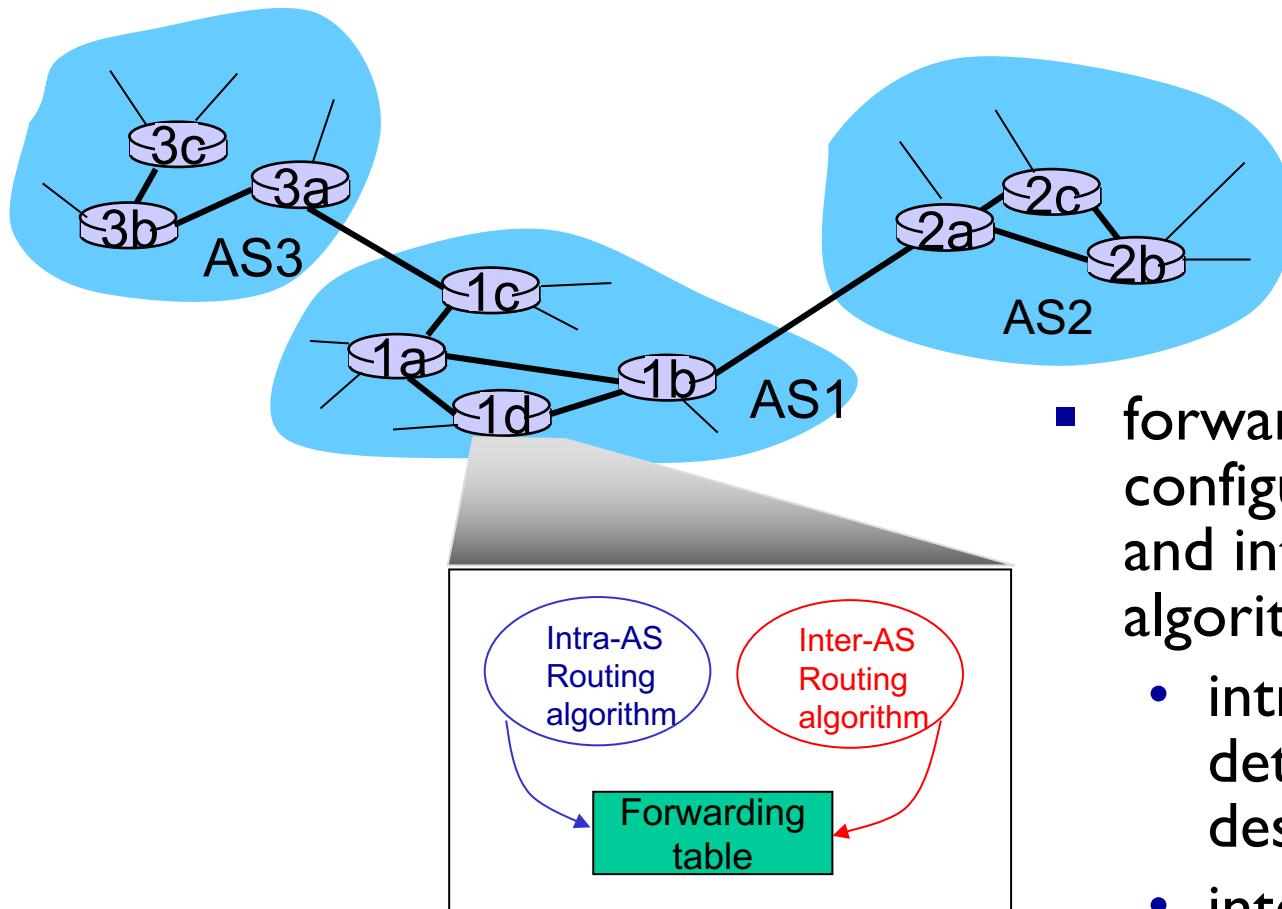
intra-AS routing

- routing among hosts, routers in same AS (“network”)
- all routers in AS must run *same* intra-domain protocol
- routers in *different* AS can run *different* intra-domain routing protocol
- gateway router: at “edge” of its own AS, has link(s) to router(s) in other AS’es

inter-AS routing

- routing among AS’es
- gateways perform inter-domain routing (as well as intra-domain routing)

Interconnected ASes



- **forwarding table**
configured by both intra-
and inter-AS routing
algorithm
 - **intra-AS routing**
determine entries for
destinations within AS
 - **inter-AS & intra-AS**
determine entries for
external destinations

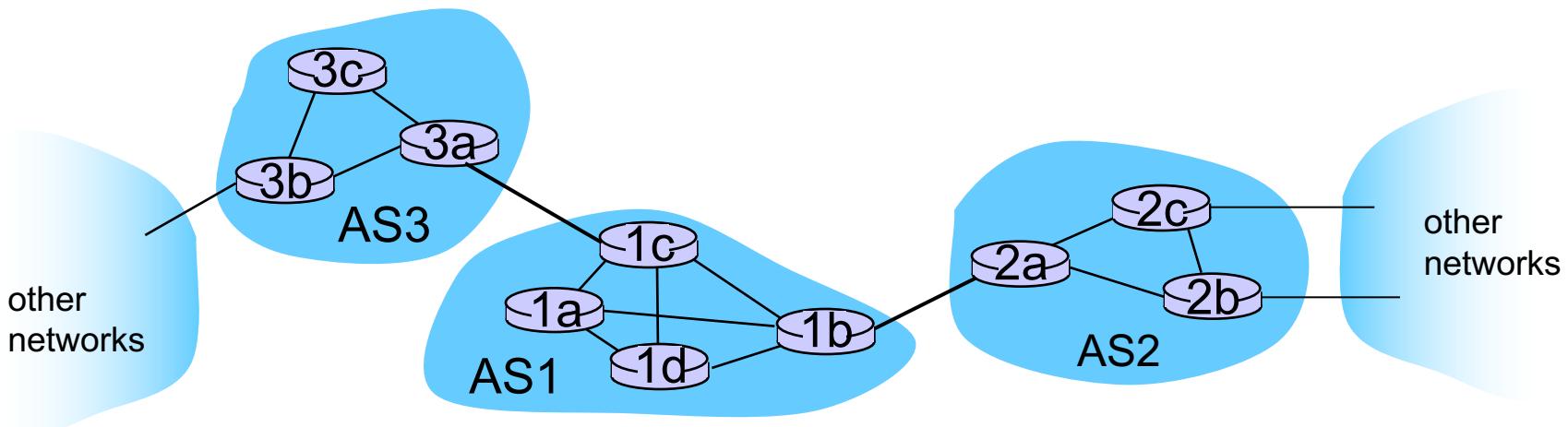
Inter-AS tasks

- suppose router in AS1 receives datagram destined outside of AS1:
 - router should forward packet to gateway router, but which one?

AS1 must:

- I. learn which dests are reachable through AS2, which through AS3
 2. propagate this reachability info to all routers in AS1

job of inter-AS routing!



Intra-AS Routing

- also known as *interior gateway protocols (IGP)*
- most common intra-AS routing protocols:
 - RIP: Routing Information Protocol
 - OSPF: Open Shortest Path First (IS-IS protocol essentially same as OSPF)
 - IGRP: Interior Gateway Routing Protocol
(Cisco proprietary for decades, until 2016)

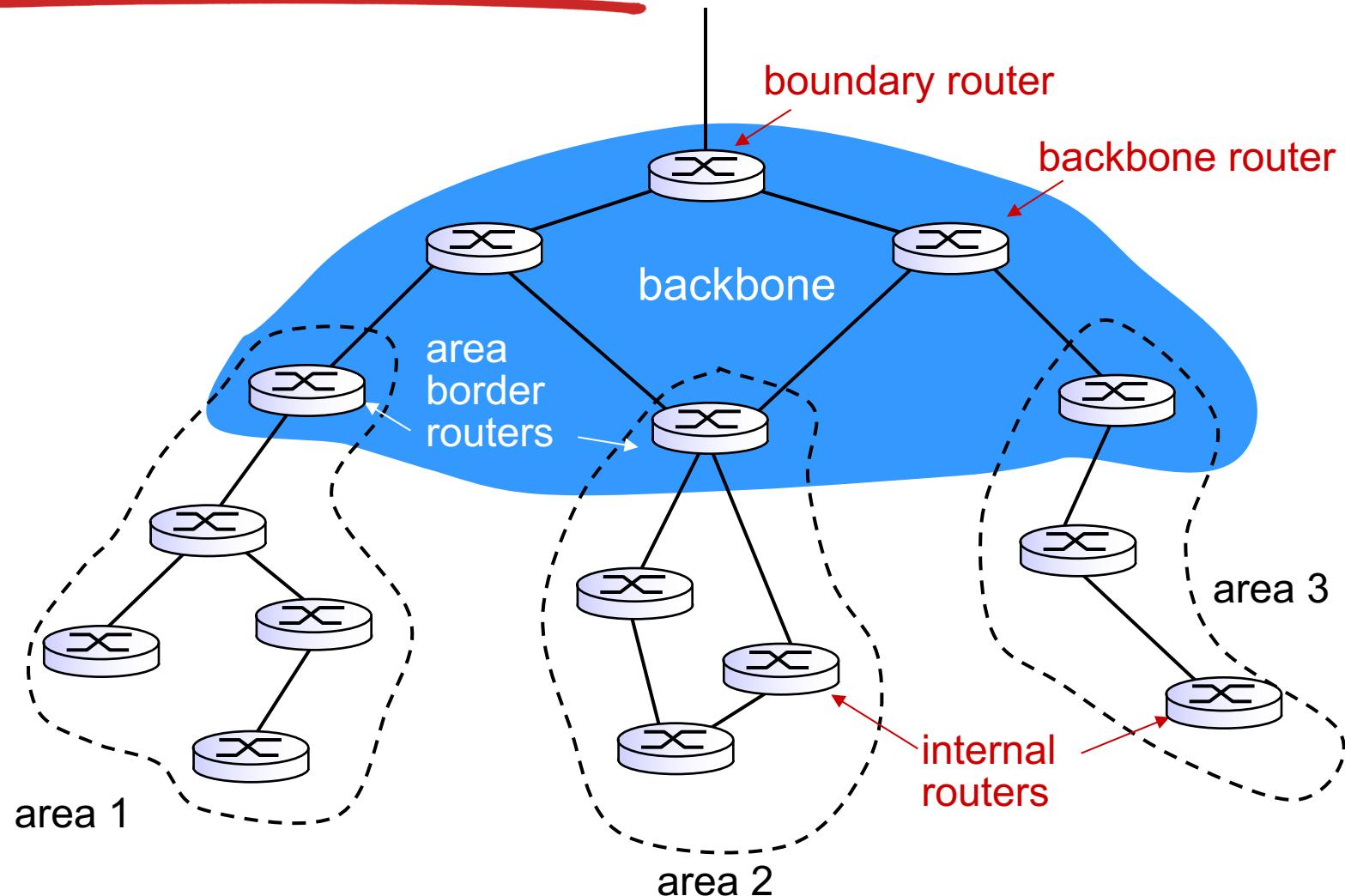
OSPF (Open Shortest Path First)

- “open”: publicly available
- uses link-state algorithm
 - link state packet dissemination
 - topology map at each node
 - route computation using Dijkstra’s algorithm
- router floods OSPF link-state advertisements to all other routers in *entire AS*
 - carried in OSPF messages directly over IP (rather than TCP or UDP)
 - link state: for each attached link
- *IS-IS routing* protocol: nearly identical to OSPF

OSPF “advanced” features

- **security:** all OSPF messages authenticated (to prevent malicious intrusion)
- **multiple same-cost paths** allowed (only one path in RIP)
- for each link, multiple cost metrics for different **TOS** (e.g., satellite link cost set low for best effort ToS; high for real-time ToS)
- integrated uni- and **multi-cast** support:
 - Multicast OSPF (MOSPF) uses same topology data base as OSPF
- **hierarchical** OSPF in large domains.

Hierarchical OSPF



Hierarchical OSPF

- *two-level hierarchy*: local area, backbone.
 - link-state advertisements only in area
 - each node has detailed area topology; only know direction (shortest path) to nets in other areas.
- *area border routers*: “summarize” distances to nets in own area, advertise to other Area Border routers.
- *backbone routers*: run OSPF routing limited to backbone.
- *boundary routers*: connect to other AS' es.

Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state

- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs:
BGP

5.5 The SDN control plane

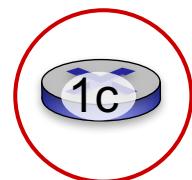
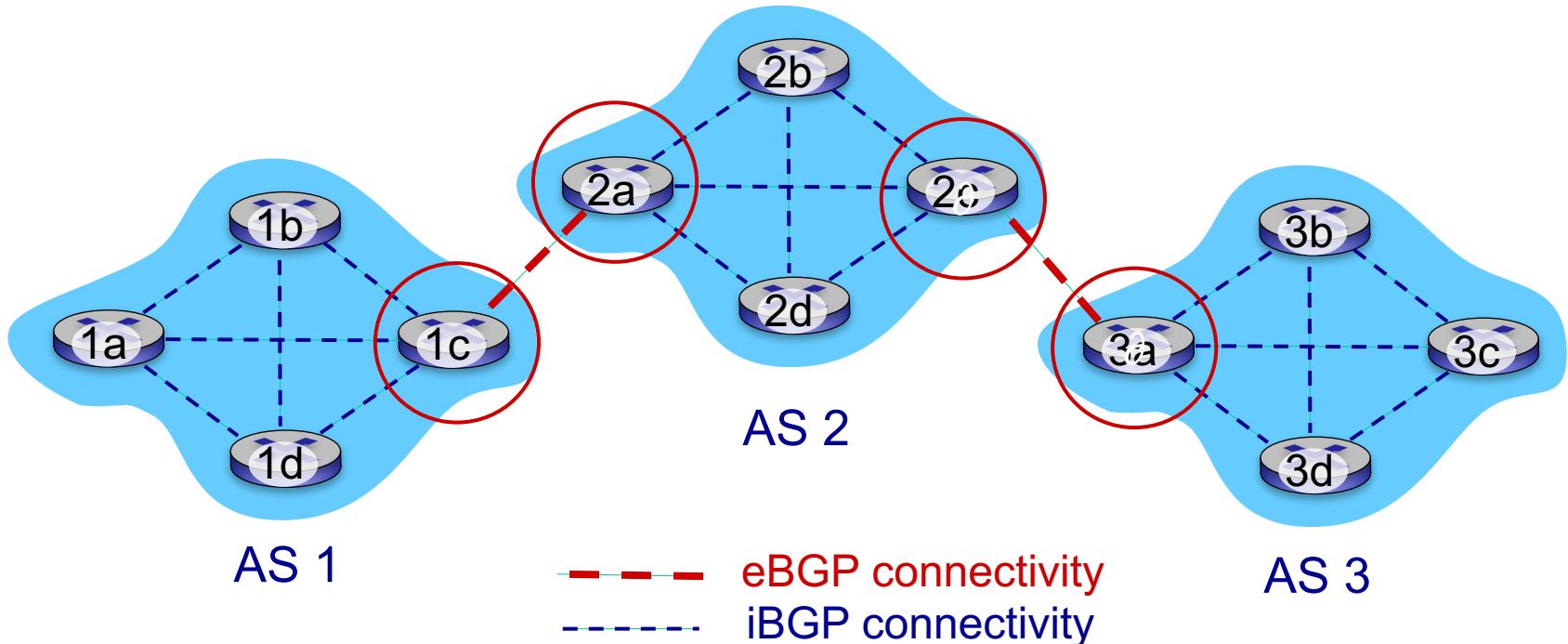
5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

Internet inter-AS routing: BGP

- **BGP (Border Gateway Protocol):** *the de facto* inter-domain routing protocol
 - “glue that holds the Internet together”
- BGP provides each AS a means to:
 - **eBGP:** obtain subnet reachability information from neighboring ASes
 - **iBGP:** propagate reachability information to all AS-internal routers.
 - determine “good” routes to other networks based on reachability information and *policy*
- allows subnet to advertise its existence to rest of Internet: “*I am here*”

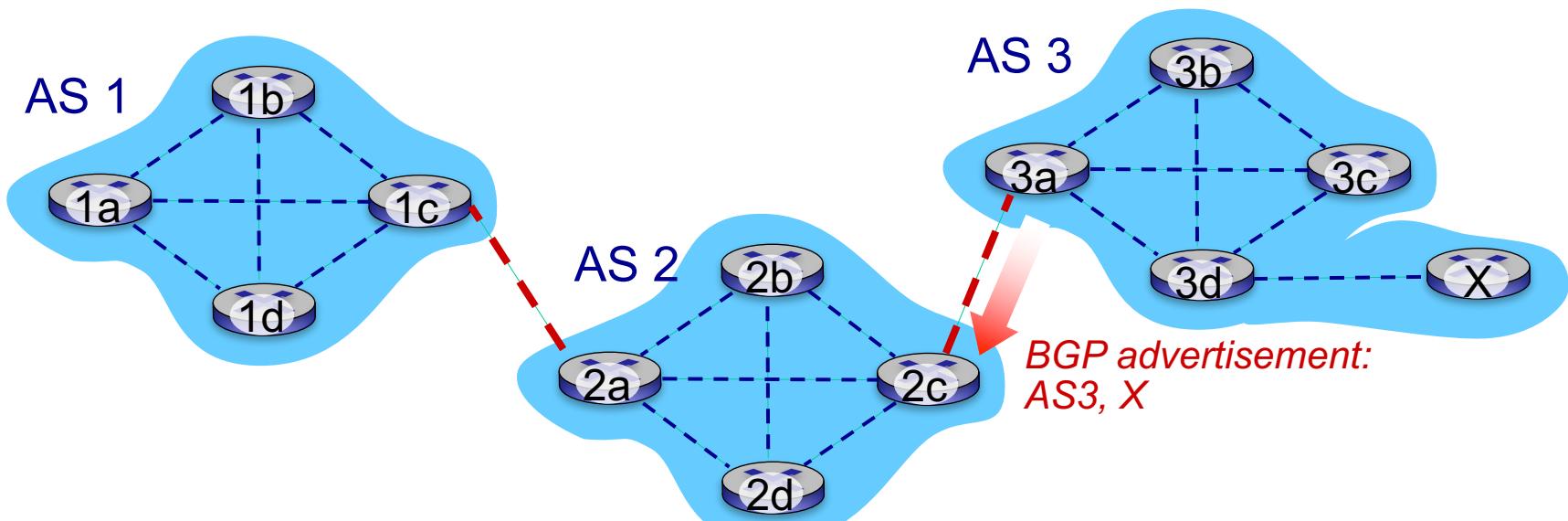
eBGP, iBGP connections



gateway routers run both eBGP and iBGP protocols

BGP basics

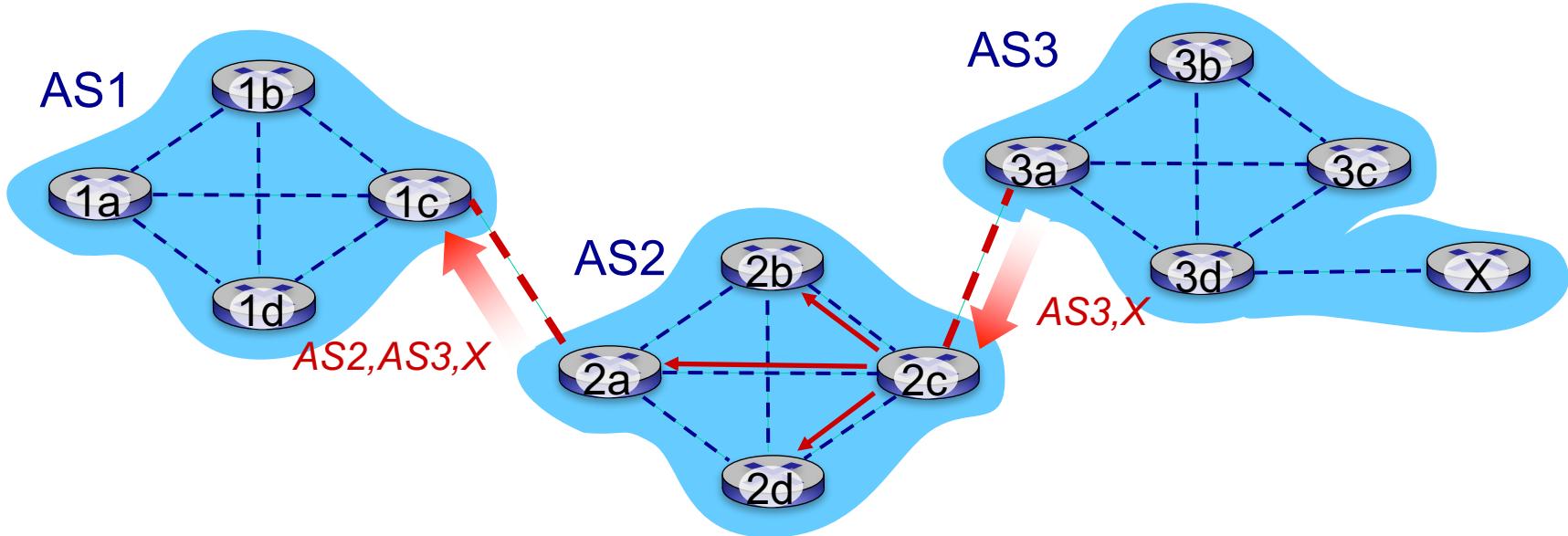
- **BGP session:** two BGP routers (“peers”) exchange BGP messages over semi-permanent TCP connection:
 - advertising *paths* to different destination network prefixes (BGP is a “path vector” protocol)
- when AS3 gateway router 3a advertises path **AS3,X** to AS2 gateway router 2c:
 - AS3 *promises* to AS2 it will forward datagrams towards X



Path attributes and BGP routes

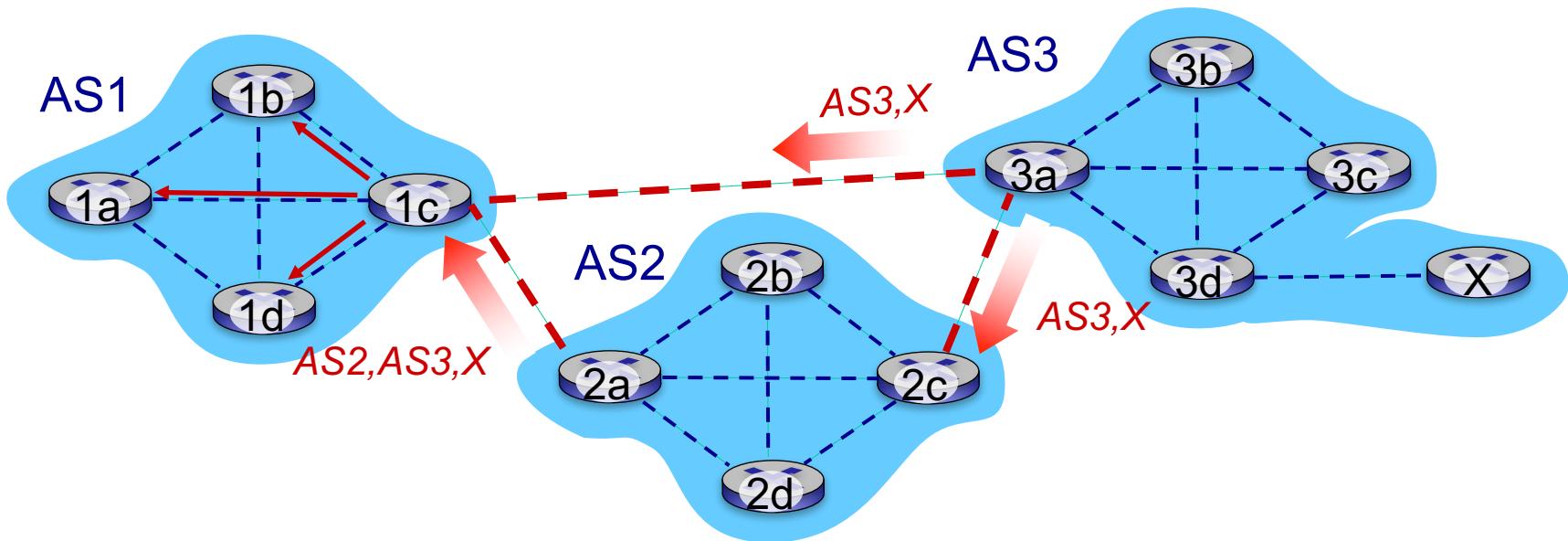
- advertised prefix includes BGP attributes
 - prefix + attributes = “route”
- two important attributes:
 - **AS-PATH**: list of ASes through which prefix advertisement has passed
 - **NEXT-HOP**: indicates specific internal-AS router to next-hop AS
- *Policy-based routing*:
 - gateway receiving route advertisement uses *import policy* to accept/decline path (e.g., never route through AS Y).
 - AS policy also determines whether to *advertise* path to other other neighboring ASes

BGP path advertisement



- AS2 router 2c receives path advertisement **AS3,X** (via eBGP) from AS3 router 3a
- Based on AS2 policy, AS2 router 2c accepts path **AS3,X**, propagates (via iBGP) to all AS2 routers
- Based on AS2 policy, AS2 router 2a advertises (via eBGP) path **AS2, AS3,X** to AS1 router 1c

BGP path advertisement



gateway router may learn about **multiple** paths to destination:

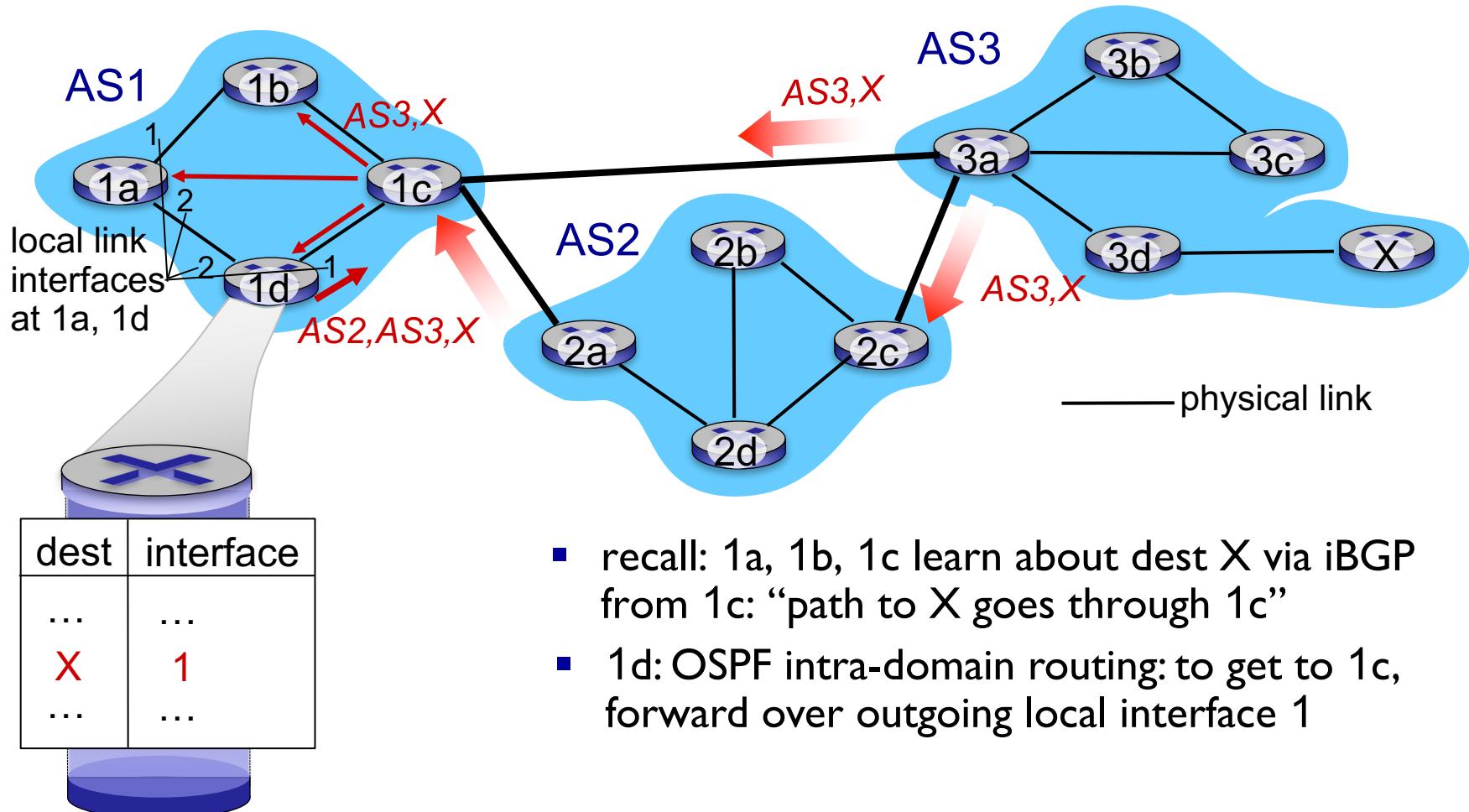
- AS1 gateway router 1c learns path **AS2,AS3,X** from 2a
- AS1 gateway router 1c learns path **AS3,X** from 3a
- Based on policy, AS1 gateway router 1c chooses path **AS3,X, and advertises path within AS1 via iBGP**

BGP messages

- BGP messages exchanged between peers over TCP connection
- BGP messages:
 - **OPEN**: opens TCP connection to remote BGP peer and authenticates sending BGP peer
 - **UPDATE**: advertises new path (or withdraws old)
 - **KEEPALIVE**: keeps connection alive in absence of UPDATES; also ACKs OPEN request
 - **NOTIFICATION**: reports errors in previous msg; also used to close connection

BGP, OSPF, forwarding table entries

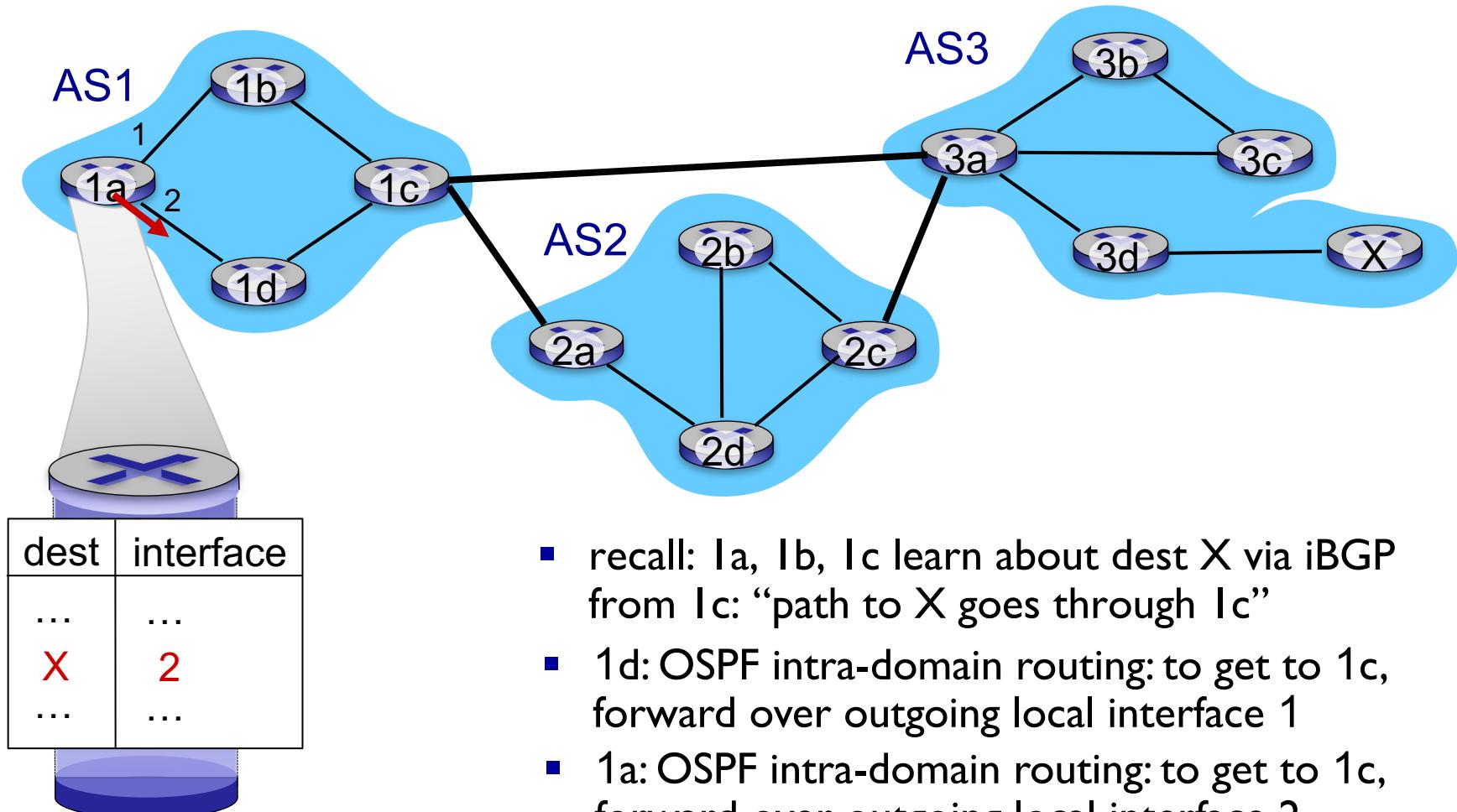
Q: how does router set forwarding table entry to distant prefix?



- recall: 1a, 1b, 1c learn about dest X via iBGP from 1c: “path to X goes through 1c”
- 1d: OSPF intra-domain routing: to get to 1c, forward over outgoing local interface 1

BGP, OSPF, forwarding table entries

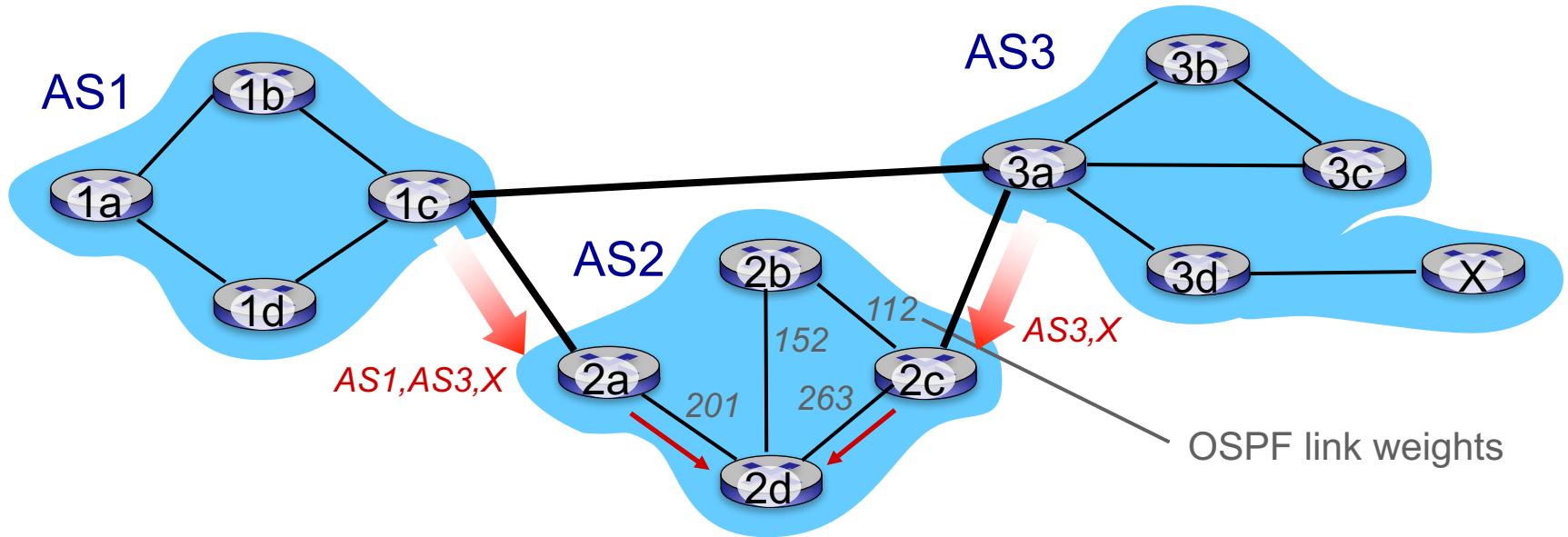
Q: how does router set forwarding table entry to distant prefix?



BGP route selection

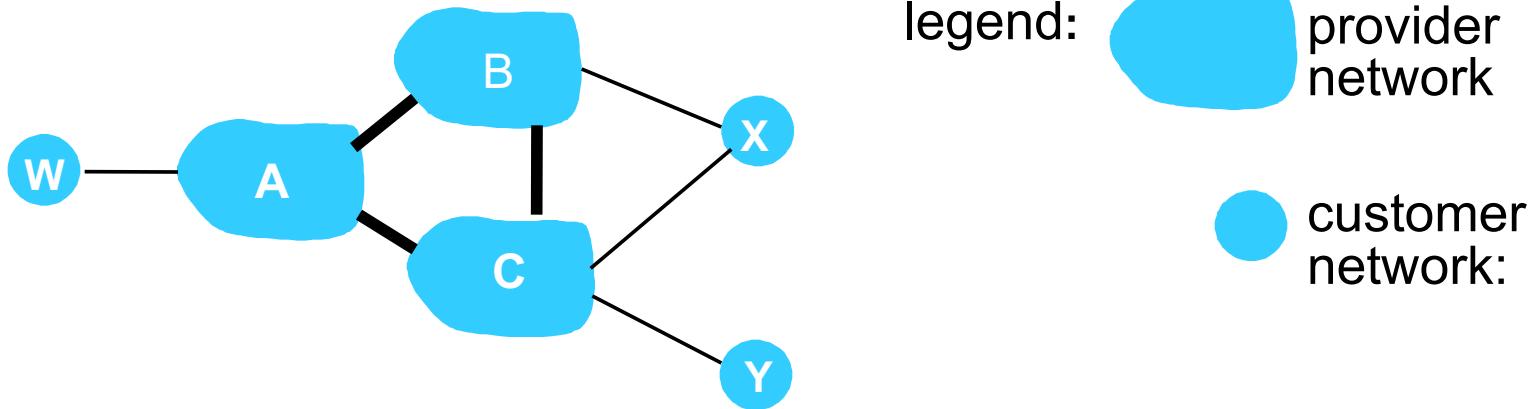
- router may learn about more than one route to destination AS, selects route based on:
 1. local preference value attribute: policy decision
 2. shortest AS-PATH
 3. closest NEXT-HOP router: hot potato routing
 4. additional criteria

Hot Potato Routing



- 2d learns (via iBGP) it can route to X via 2a or 2c
- *hot potato routing*: choose local gateway that has least intra-domain cost (e.g., 2d chooses 2a, even though more AS hops to X): don't worry about inter-domain cost!

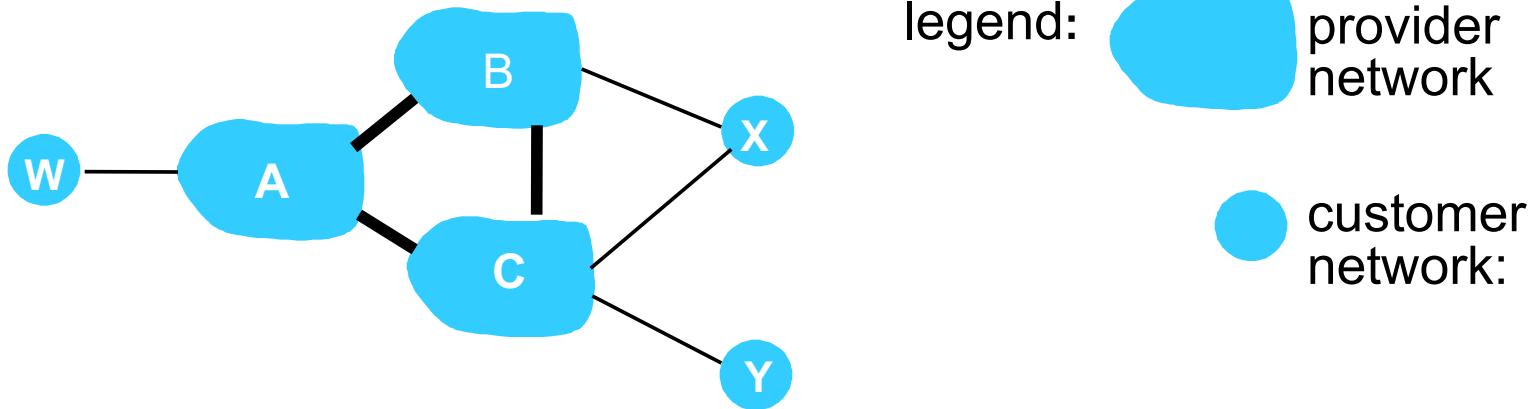
BGP: achieving policy via advertisements



Suppose an ISP only wants to route traffic to/from its customer networks (does not want to carry transit traffic between other ISPs)

- A advertises path Aw to B and to C
- B *chooses not to advertise* BAw to C:
 - B gets no “revenue” for routing CBAw, since none of C, A, w are B’s customers
 - C does not learn about CBAw path
- C will route CAw (not using B) to get to w

BGP: achieving policy via advertisements



Suppose an ISP only wants to route traffic to/from its customer networks (does not want to carry transit traffic between other ISPs)

- A,B,C are *provider networks*
- X,W,Y are customer (of provider networks)
- X is *dual-homed*: attached to two networks
- *policy to enforce*: X does not want to route from B to C via X
 - .. so X will not advertise to B a route to C

Why different Intra-, Inter-AS routing ?

policy:

- inter-AS: admin wants control over how its traffic routed, who routes through its net.
- intra-AS: single admin, so no policy decisions needed

scale:

- hierarchical routing saves table size, reduced update traffic

performance:

- intra-AS: can focus on performance
- inter-AS: policy may dominate over performance

Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs:
BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

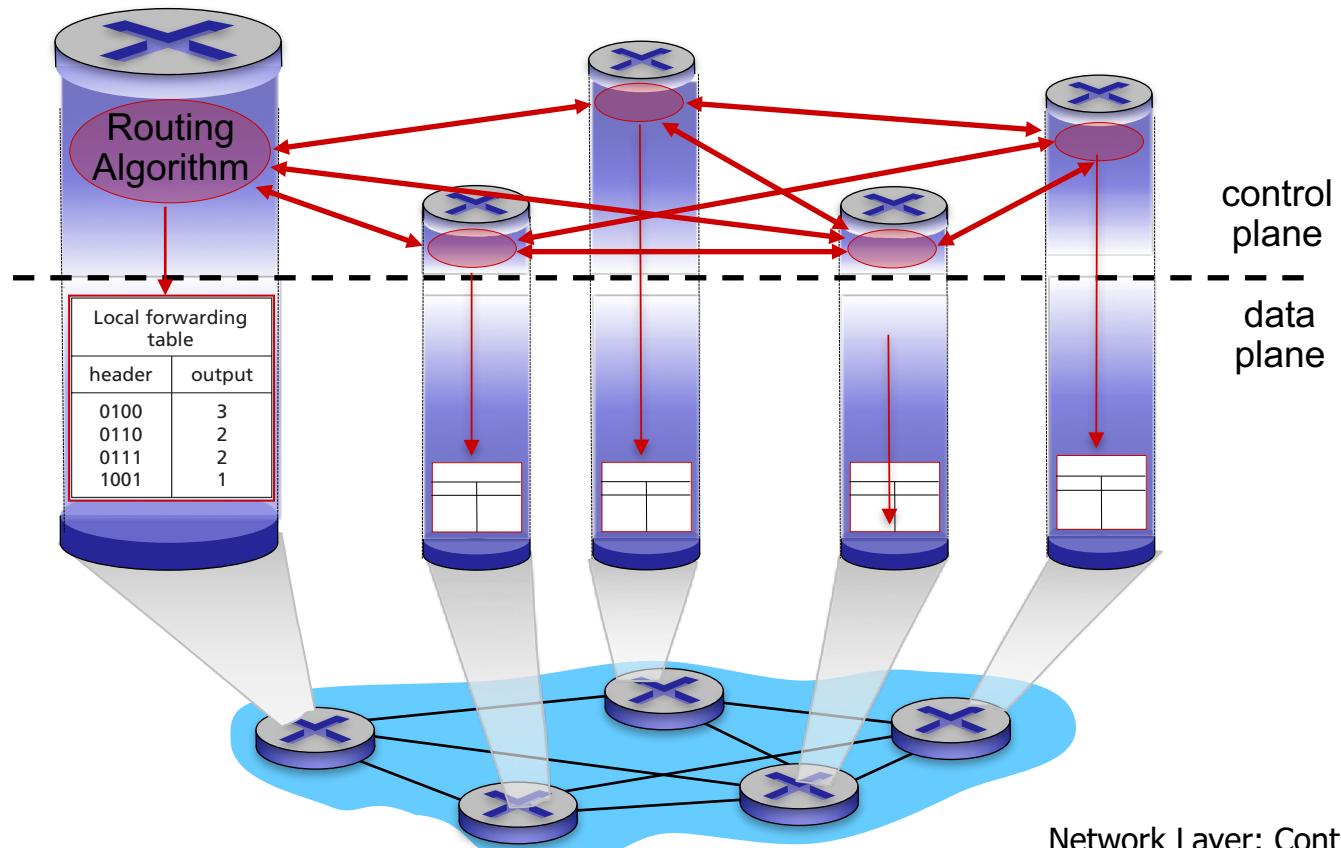
5.7 Network management and SNMP

Software defined networking (SDN)

- Internet network layer: historically has been implemented via distributed, per-router approach
 - *monolithic* router contains switching hardware, runs proprietary implementation of Internet standard protocols (IP, RIP, IS-IS, OSPF, BGP) in proprietary router OS (e.g., Cisco IOS)
 - different “middleboxes” for different network layer functions: firewalls, load balancers, NAT boxes, ..
- ~2005: renewed interest in rethinking network control plane

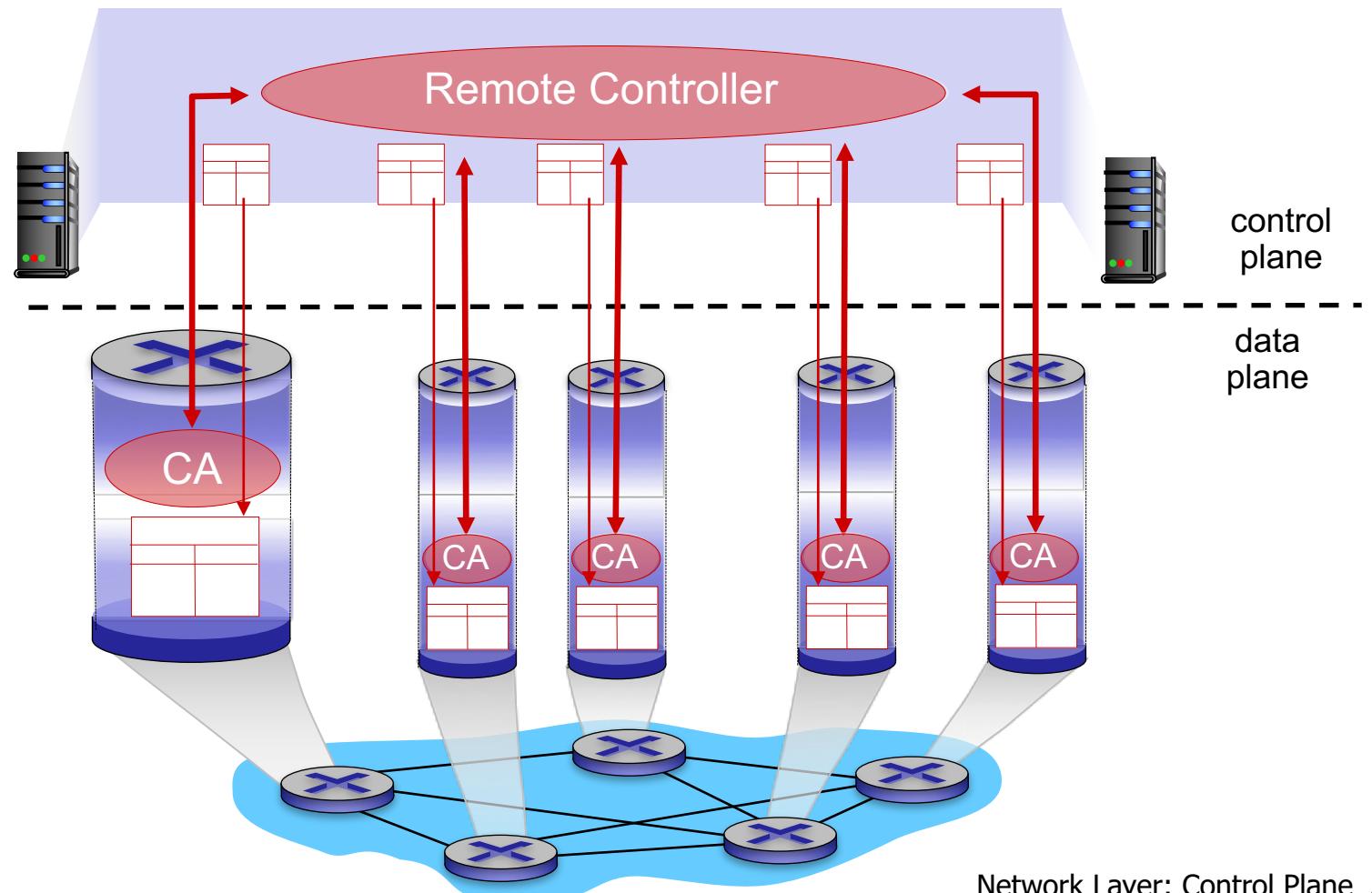
Recall: per-router control plane

Individual routing algorithm components *in each and every router* interact with each other in control plane to compute forwarding tables



Recall: logically centralized control plane

A distinct (typically remote) controller interacts with local control agents (CAs) in routers to compute forwarding tables

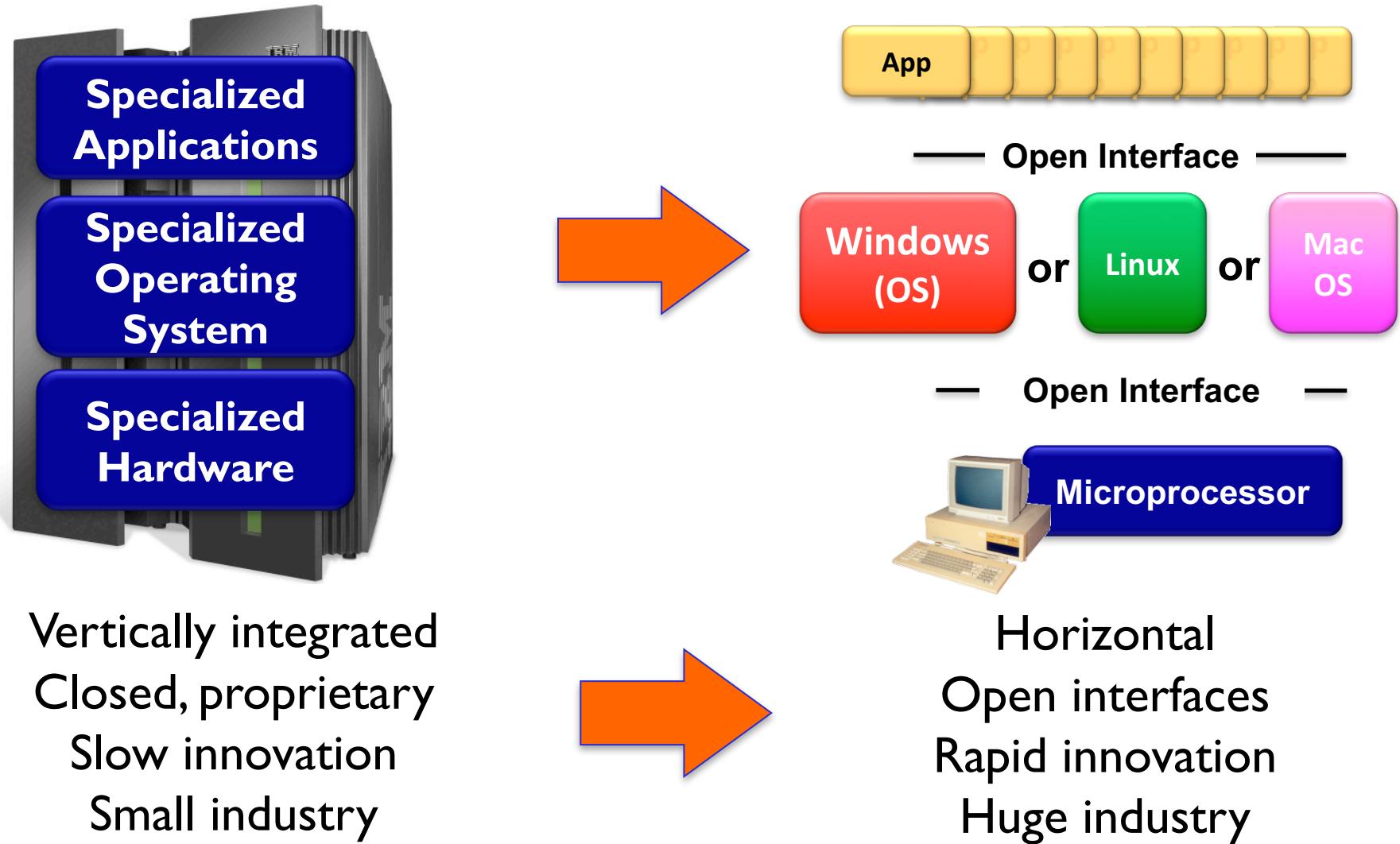


Software defined networking (SDN)

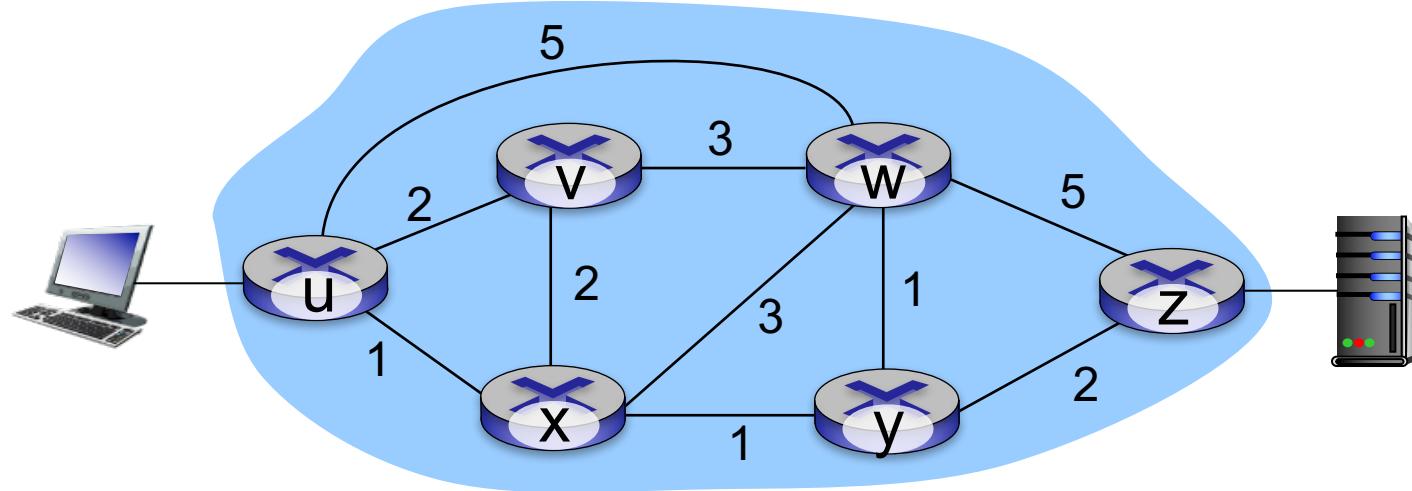
Why a *logically centralized* control plane?

- easier network management: avoid router misconfigurations, greater flexibility of traffic flows
- table-based forwarding (recall OpenFlow API) allows “programming” routers
 - centralized “programming” easier: compute tables centrally and distribute
 - distributed “programming”: more difficult: compute tables as result of distributed algorithm (protocol) implemented in each and every router
- open (non-proprietary) implementation of control plane

Analogy: mainframe to PC evolution*



Traffic engineering: difficult traditional routing

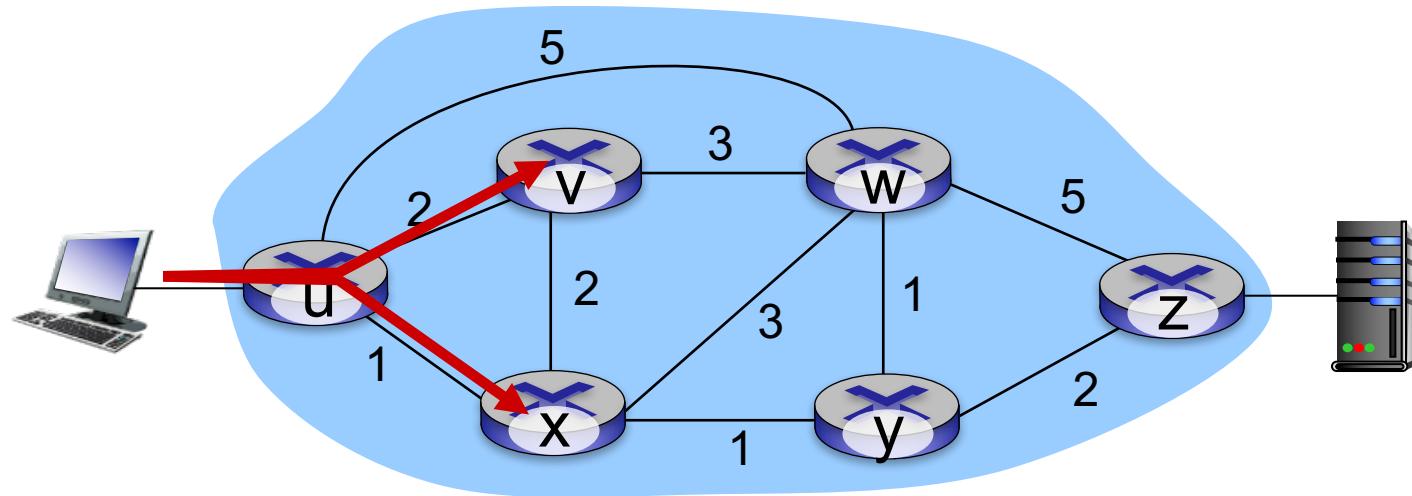


Q: what if network operator wants u-to-z traffic to flow along $uvwz$, x-to-z traffic to flow $xwyz$?

A: need to define link weights so traffic routing algorithm computes routes accordingly (or need a new routing algorithm)!

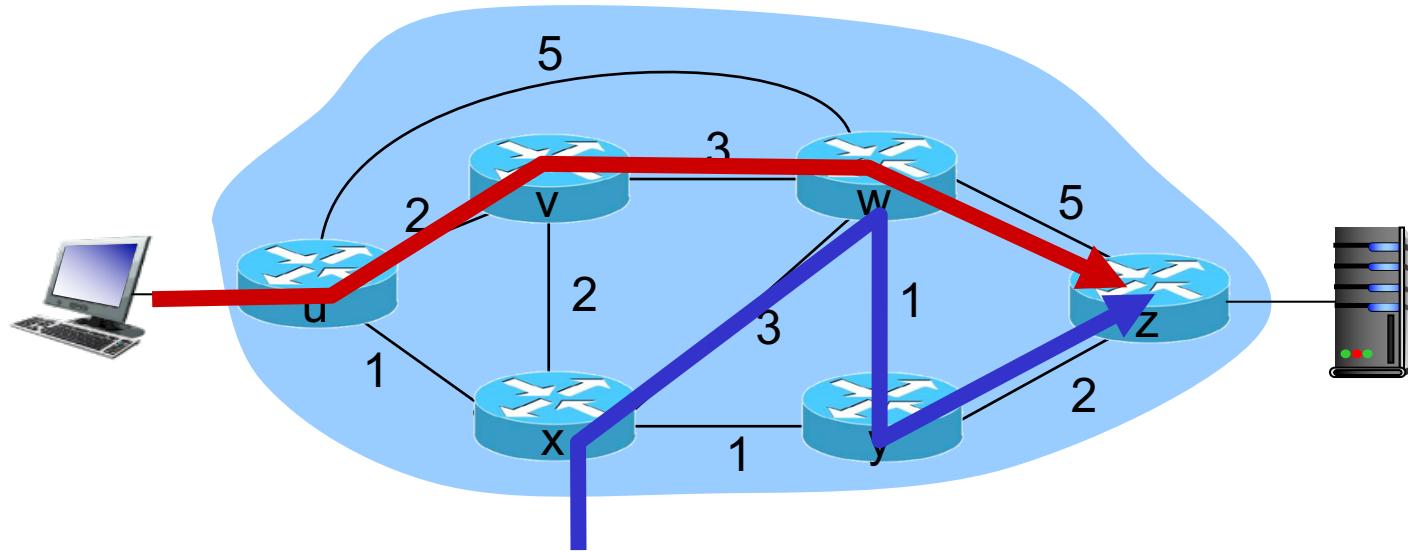
Link weights are only control “knobs”: wrong!

Traffic engineering: difficult



Q: what if network operator wants to split u-to-z traffic along uvwz *and* uxzy (load balancing)?
A: can't do it (or need a new routing algorithm)

Traffic engineering: difficult



Q: what if w wants to route blue and red traffic differently?

A: can't do it (with destination based forwarding, and LS, DV routing)

Software defined networking (SDN)

4. programmable control —
applications

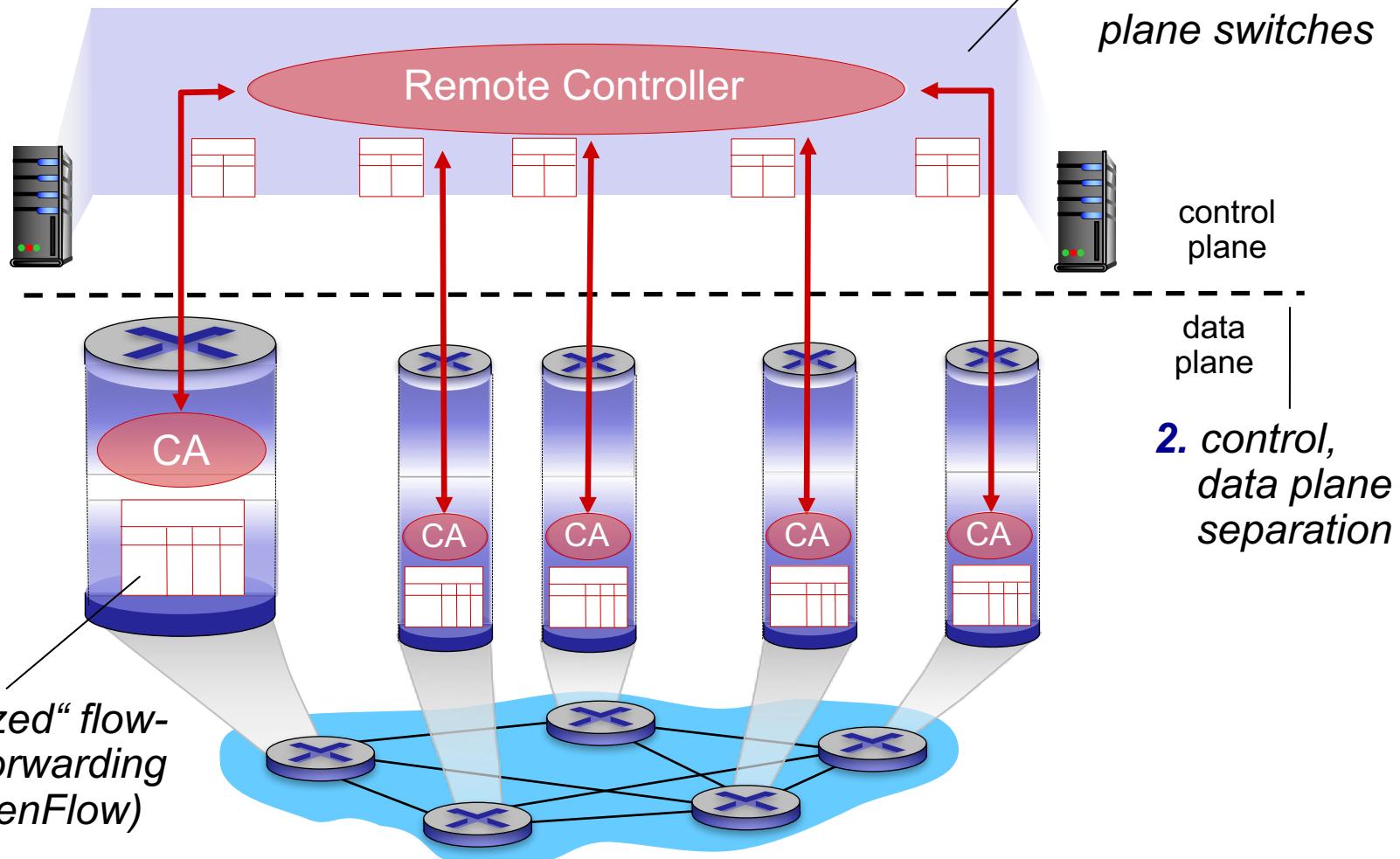
routing

access
control

...

load
balance

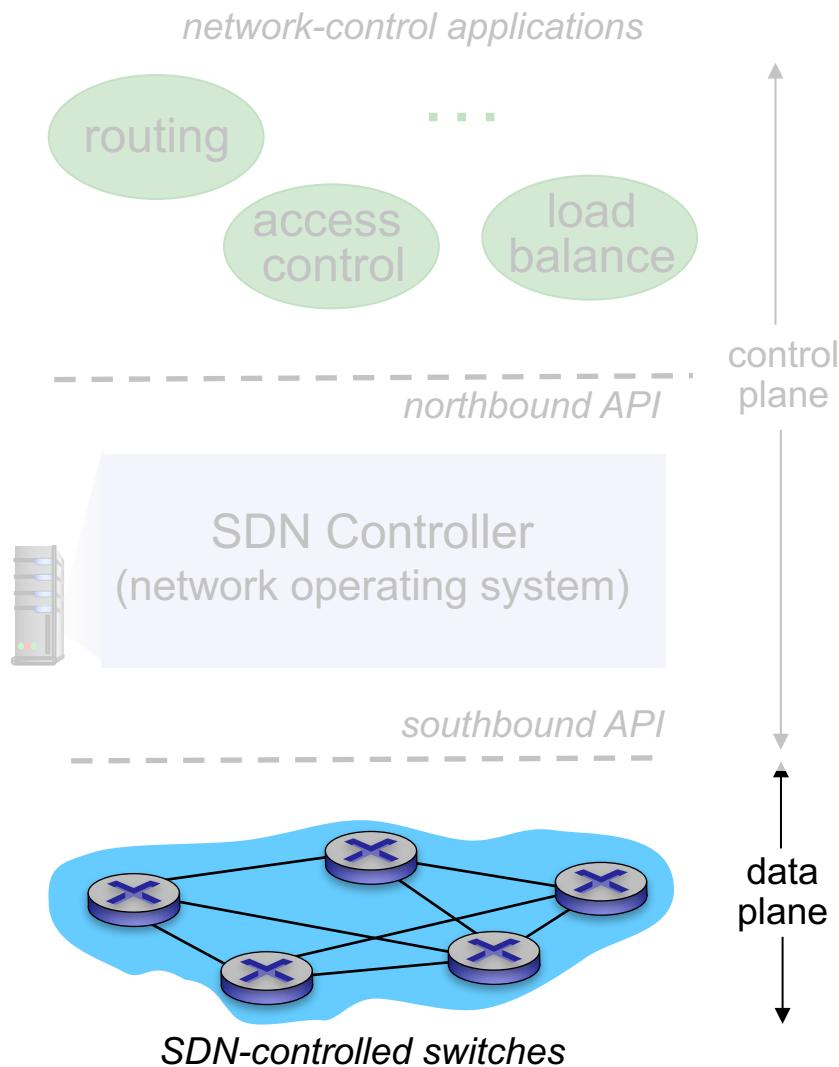
3. control plane
functions
external to data-
plane switches



SDN perspective: data plane switches

Data plane switches

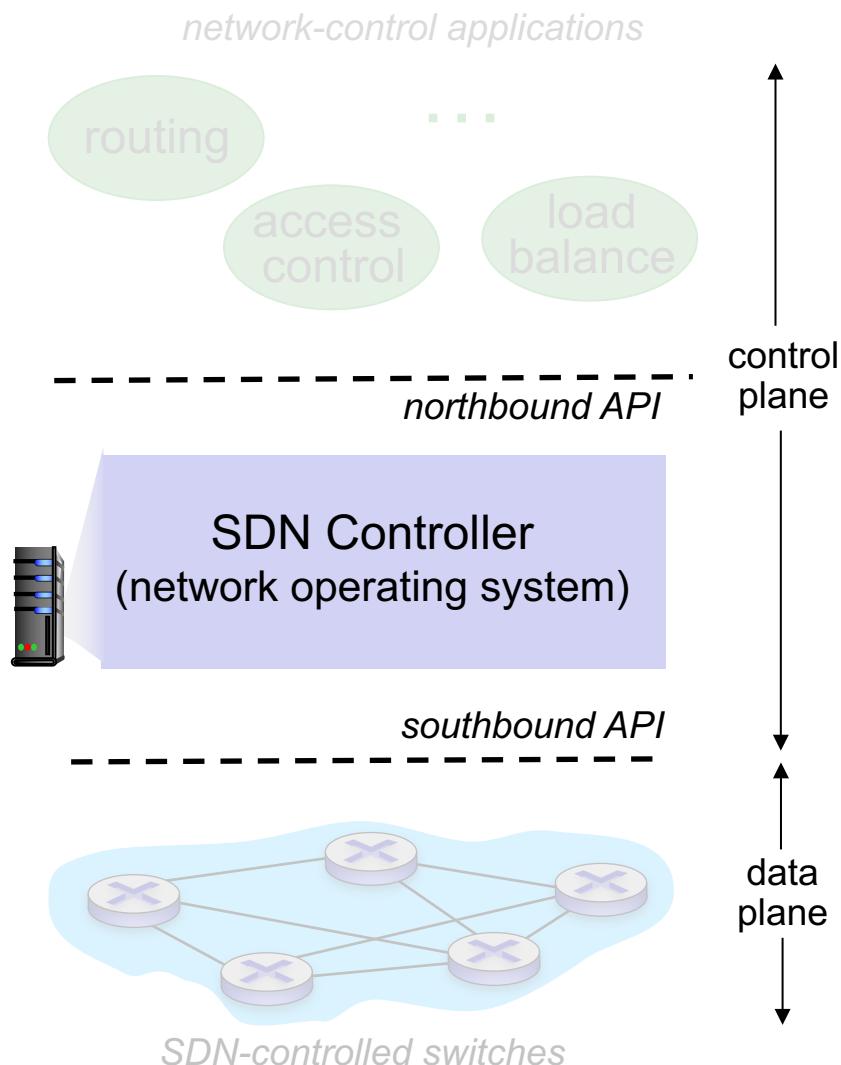
- fast, simple, commodity switches implementing generalized data-plane forwarding (Section 4.4) in hardware
- switch flow table computed, installed by controller
- API for table-based switch control (e.g., OpenFlow)
 - defines what is controllable and what is not
- protocol for communicating with controller (e.g., OpenFlow)



SDN perspective: SDN controller

SDN controller (network OS):

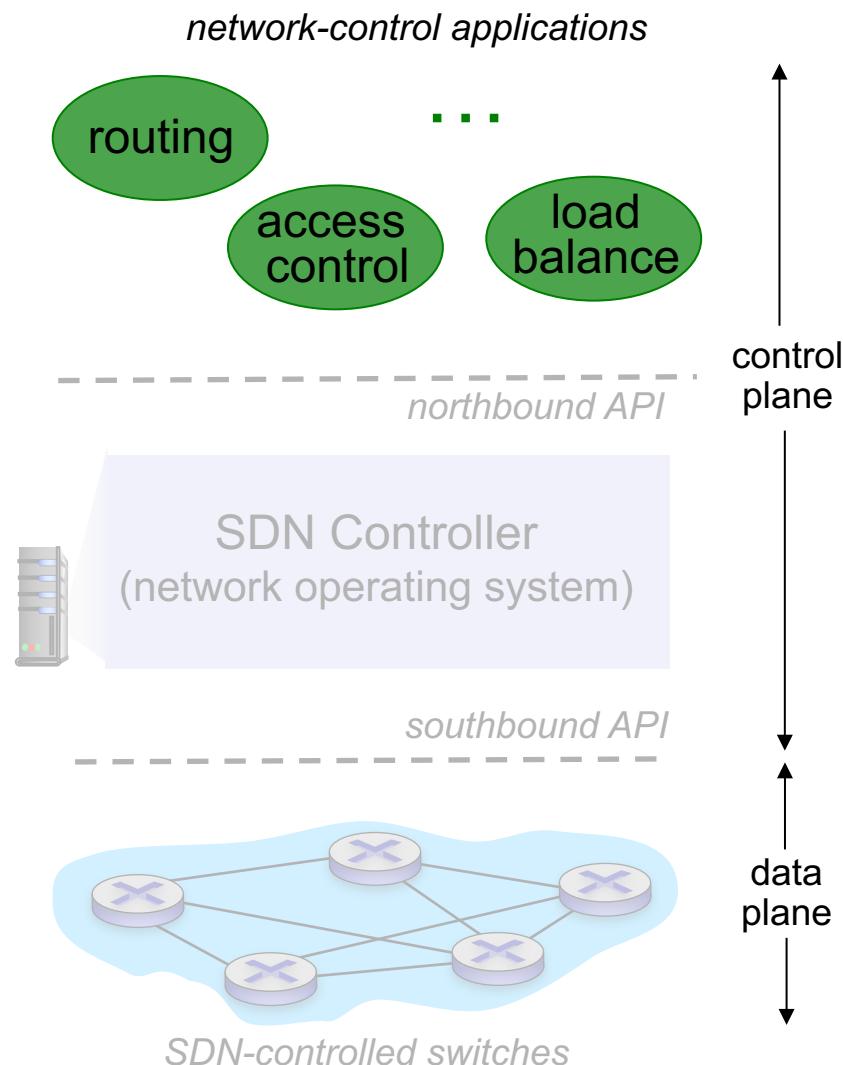
- maintain network state information
- interacts with network control applications “above” via northbound API
- interacts with network switches “below” via southbound API
- implemented as distributed system for performance, scalability, fault-tolerance, robustness



SDN perspective: control applications

network-control apps:

- “brains” of control:
implement control functions
using lower-level services, API
provided by SDN controller
- *unbundled*: can be provided by
3rd party: distinct from routing
vendor, or SDN controller

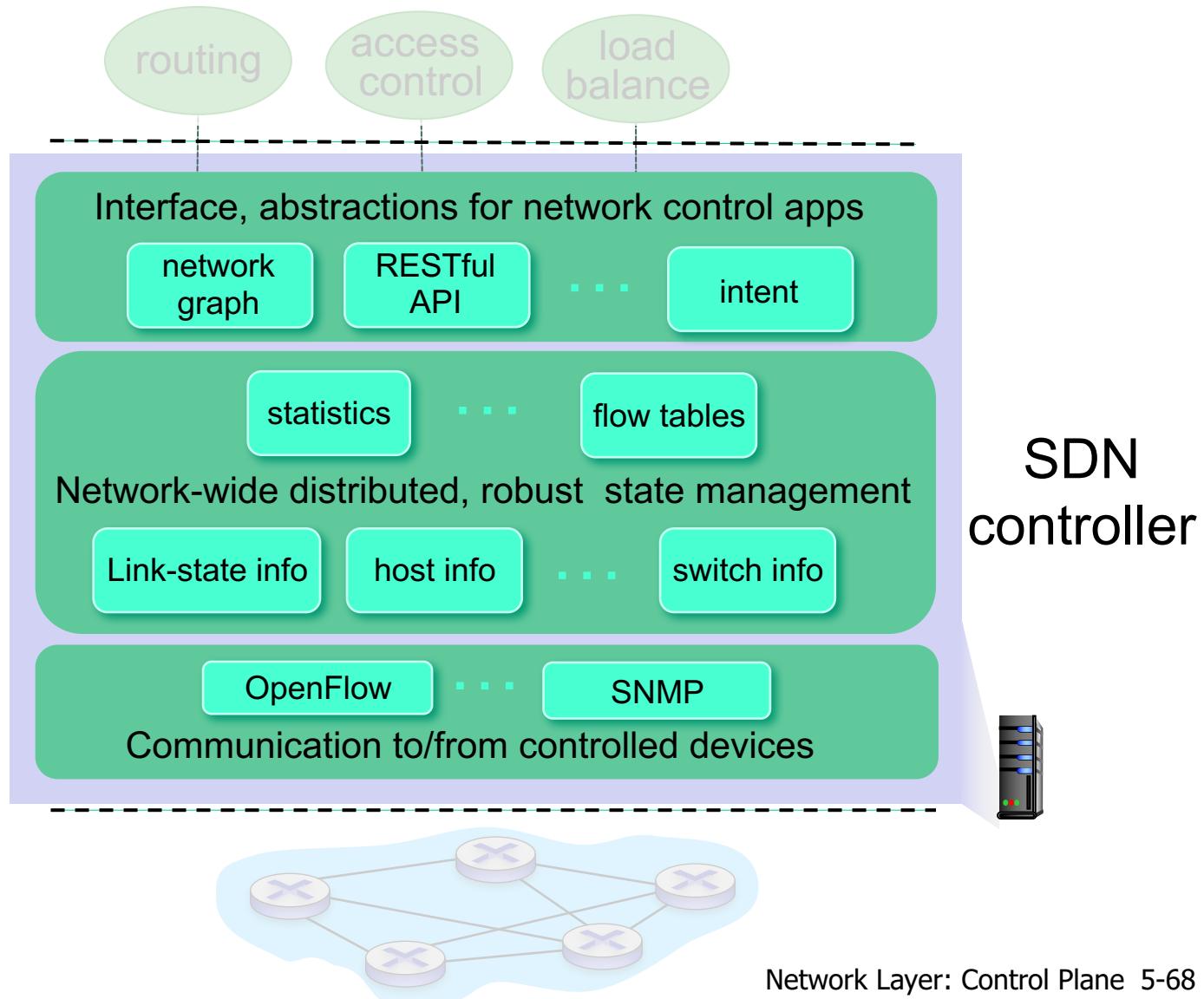


Components of SDN controller

Interface layer to network control apps: abstractions API

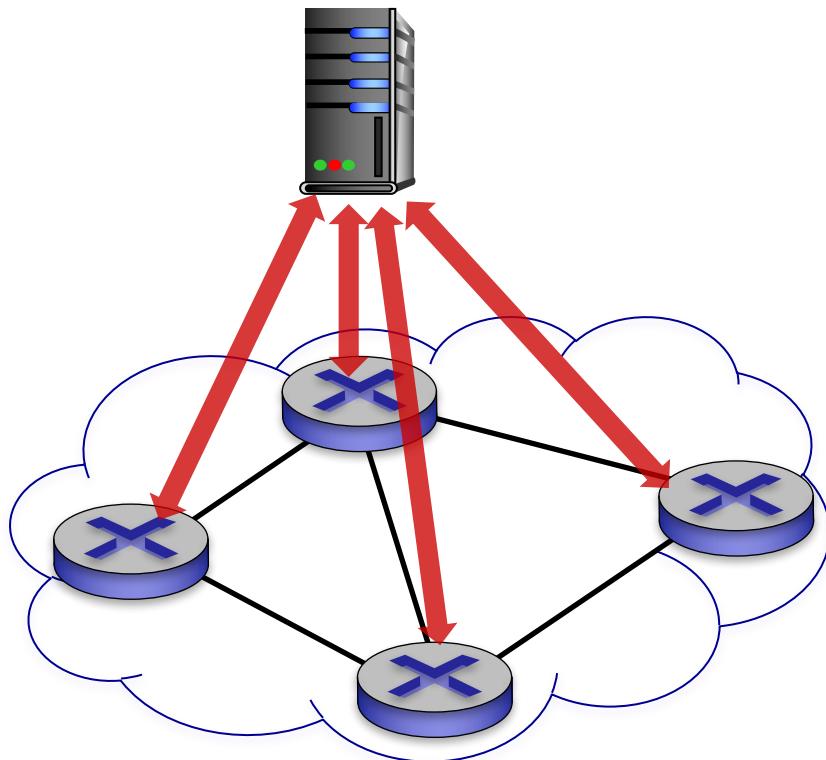
Network-wide state management layer: state of networks links, switches, services: a *distributed database*

communication layer: communicate between SDN controller and controlled switches



OpenFlow protocol

OpenFlow Controller

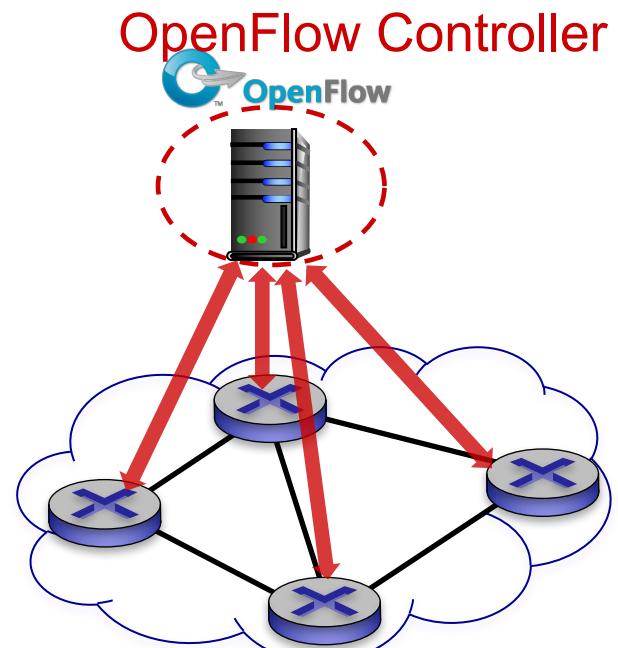


- operates between controller, switch
- TCP used to exchange messages
 - optional encryption
- three classes of OpenFlow messages:
 - controller-to-switch
 - asynchronous (switch to controller)
 - symmetric (misc)

OpenFlow: controller-to-switch messages

Key controller-to-switch messages

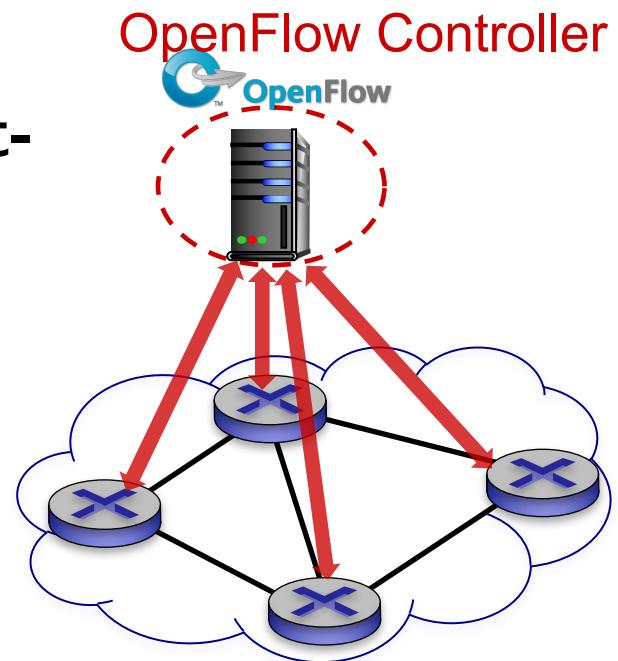
- **features**: controller queries switch features, switch replies
- **configure**: controller queries/sets switch configuration parameters
- **modify-state**: add, delete, modify flow entries in the OpenFlow tables
- **packet-out**: controller can send this packet out of specific switch port



OpenFlow: switch-to-controller messages

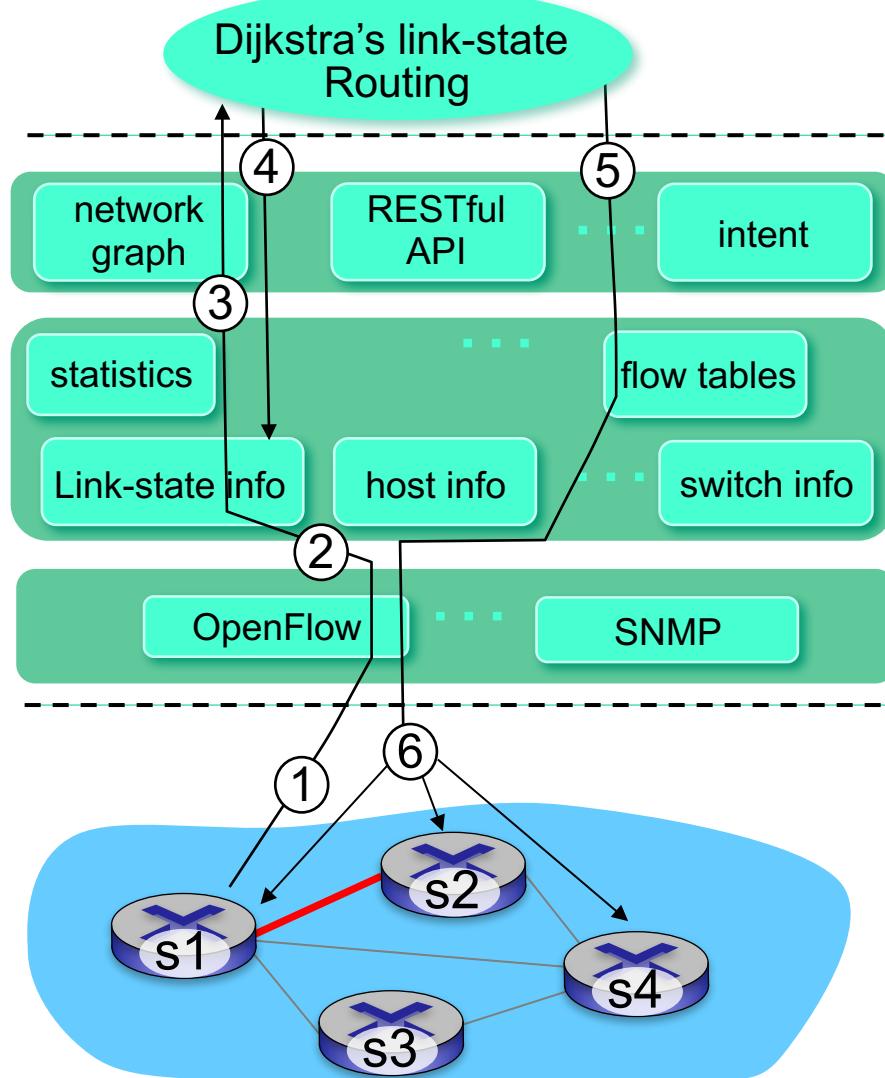
Key switch-to-controller messages

- **packet-in:** transfer packet (and its control) to controller. See packet-out message from controller
- **flow-removed:** flow table entry deleted at switch
- **port status:** inform controller of a change on a port.



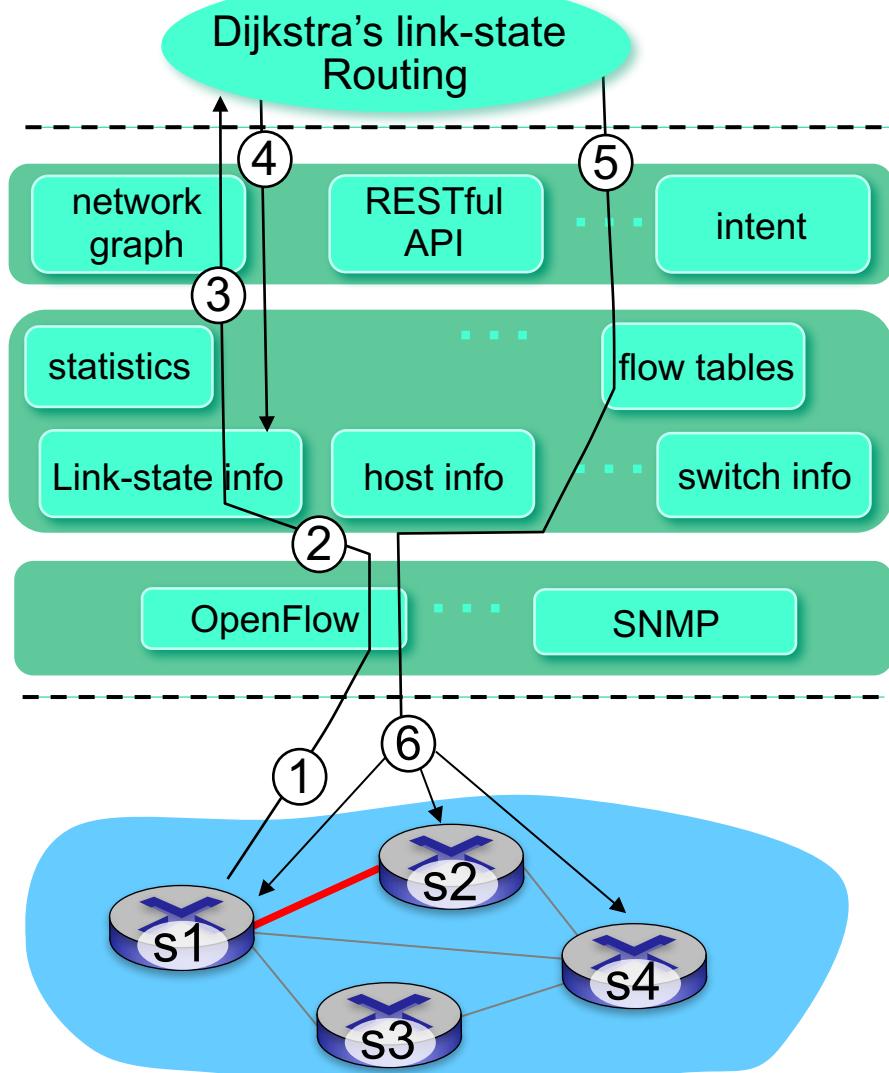
Fortunately, network operators don't “program” switches by creating/sending OpenFlow messages directly. Instead use higher-level abstraction at controller

SDN: control/data plane interaction example



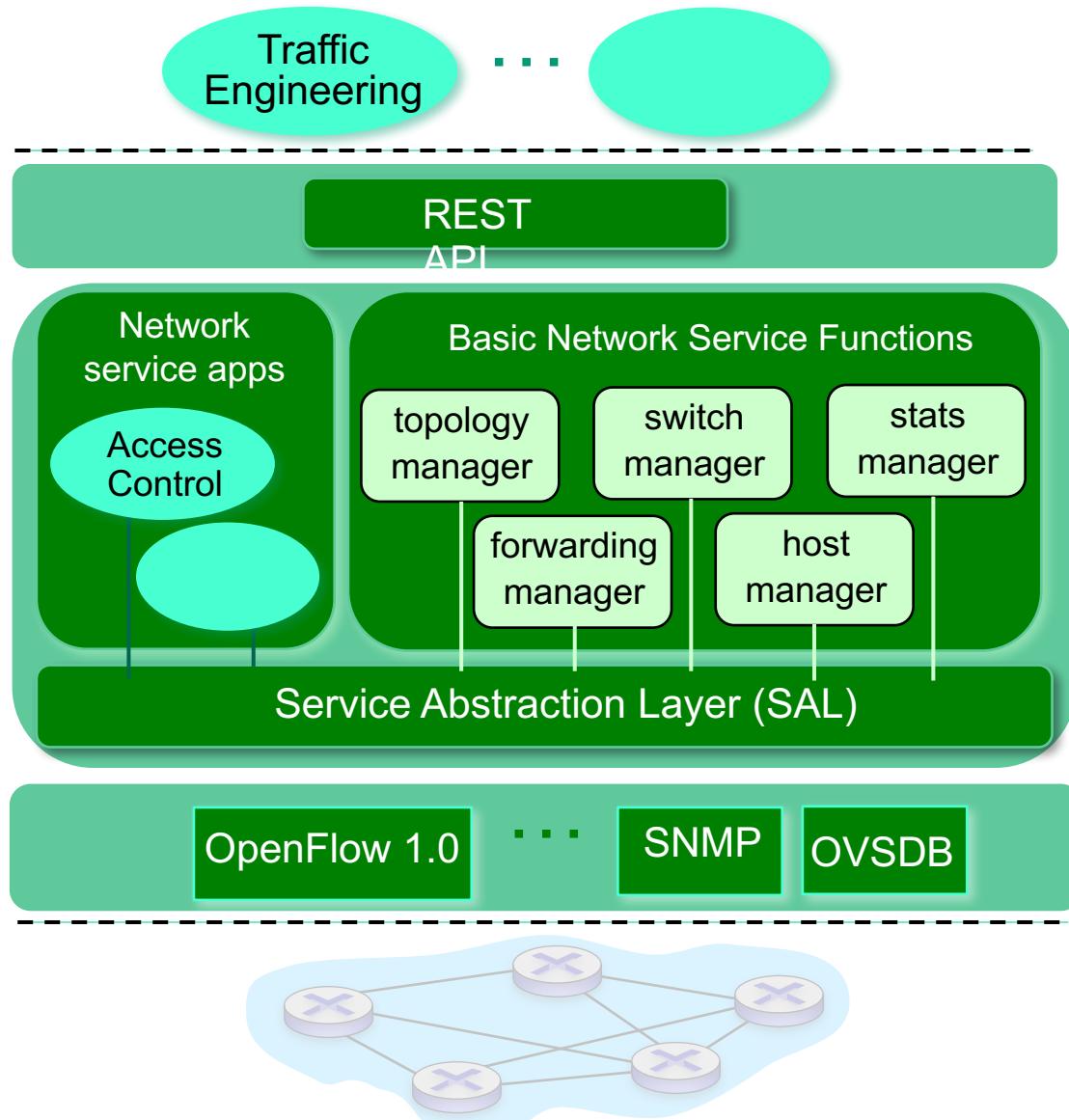
- ① SI, experiencing link failure using OpenFlow port status message to notify controller
- ② SDN controller receives OpenFlow message, updates link status info
- ③ Dijkstra's routing algorithm application has previously registered to be called whenever link status changes. It is called.
- ④ Dijkstra's routing algorithm access network graph info, link state info in controller, computes new routes

SDN: control/data plane interaction example



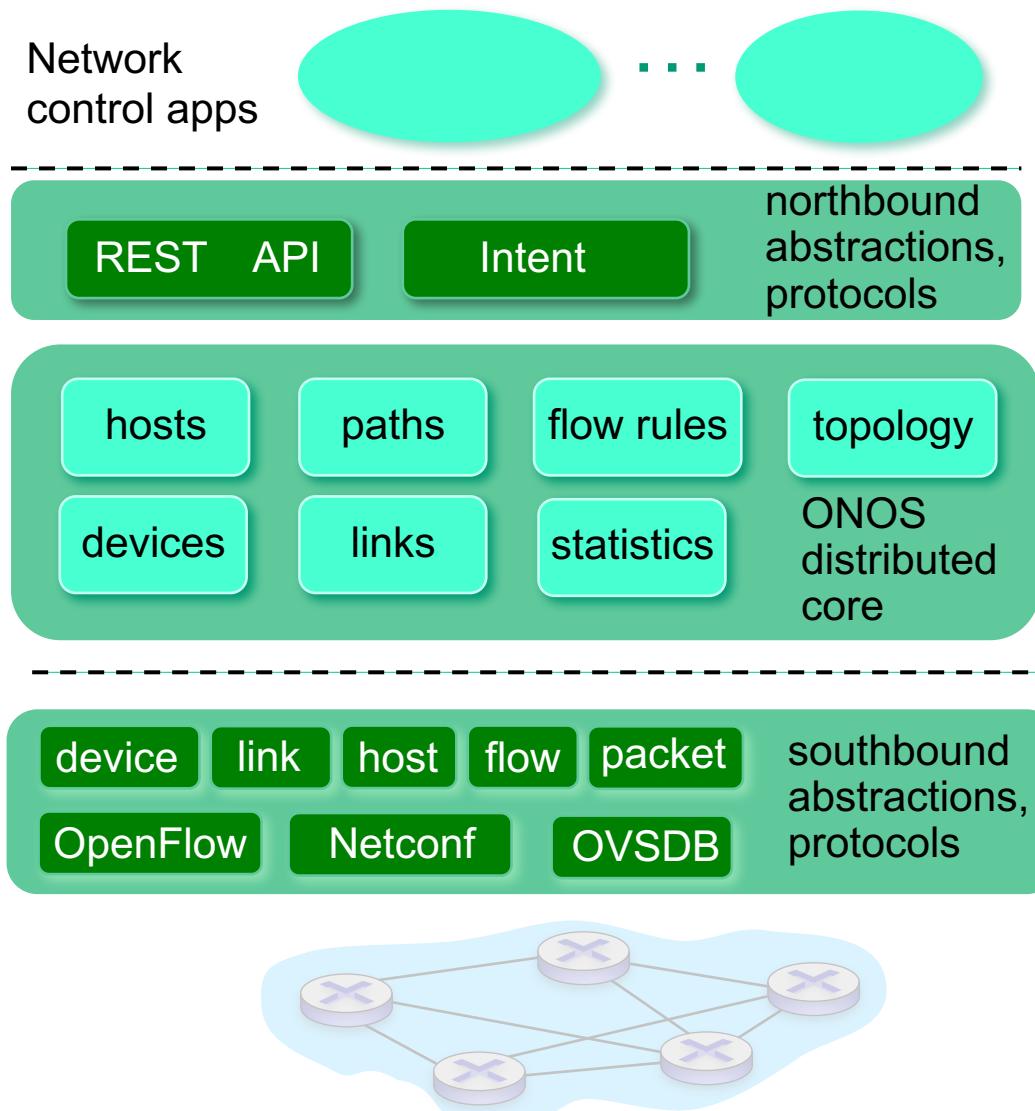
- ⑤ link state routing app interacts with flow-table-computation component in SDN controller, which computes new flow tables needed
- ⑥ Controller uses OpenFlow to install new tables in switches that need updating

OpenDaylight (ODL) controller



- ODL Lithium controller
- network apps may be contained within, or be external to SDN controller
- Service Abstraction Layer: interconnects internal, external applications and services

ONOS controller



- **control apps separate from controller**
- **intent framework:** high-level specification of service: what rather than how
- **considerable emphasis on distributed core:** service reliability, replication performance scaling

SDN: selected challenges

- hardening the control plane: dependable, reliable, performance-scalable, secure distributed system
 - robustness to failures: leverage strong theory of reliable distributed system for control plane
 - dependability, security: “baked in” from day one?
- networks, protocols meeting mission-specific requirements
 - e.g., real-time, ultra-reliable, ultra-secure
- Internet-scaling

Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state

- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs:
BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

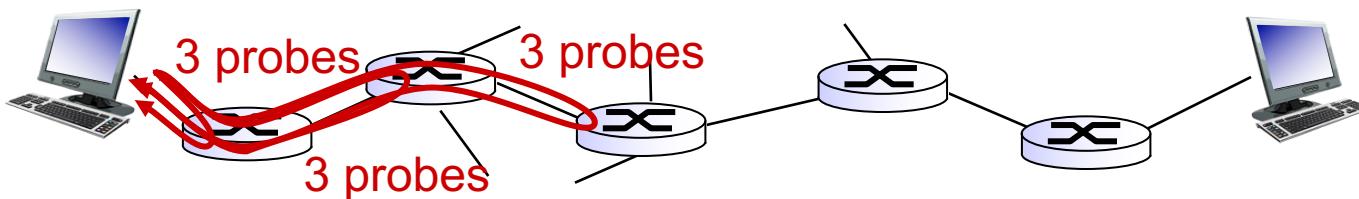
ICMP: internet control message protocol

- used by hosts & routers to communicate network-level information
 - error reporting: unreachable host, network, port, protocol
 - echo request/reply (used by ping)
- network-layer “above” IP:
 - ICMP msgs carried in IP datagrams
- **ICMP message:** type, code plus first 8 bytes of IP datagram causing error

Type	Code	description
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

Traceroute and ICMP

- source sends series of UDP segments to destination
 - first set has TTL = 1
 - second set has TTL=2, etc.
 - unlikely port number
 - when datagram in n th set arrives to n th router:
 - router discards datagram and sends source ICMP message (type 11, code 0)
 - ICMP message include name of router & IP address
 - when ICMP message arrives, source records RTTs
- stopping criteria:*
- UDP segment eventually arrives at destination host
 - destination returns ICMP “port unreachable” message (type 3, code 3)
 - source stops



Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs:
BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

What is network management?

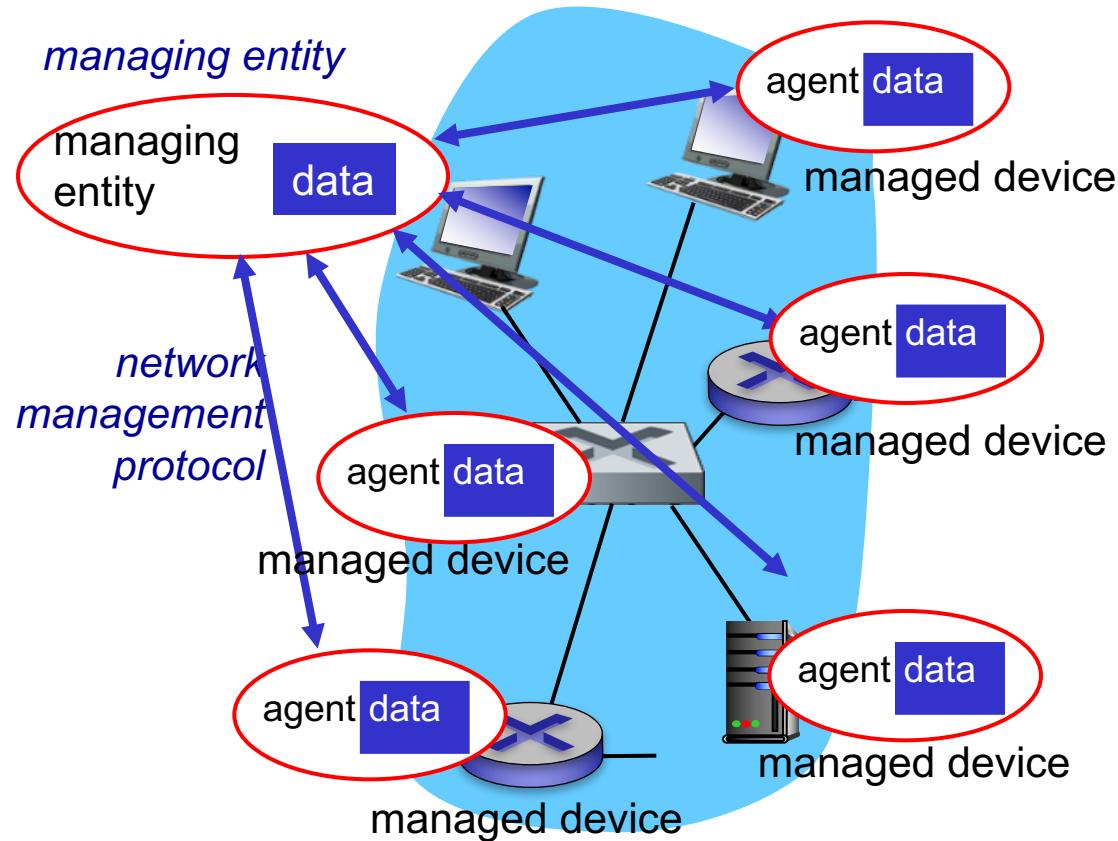
- autonomous systems (aka “network”): 1000s of interacting hardware/software components
- other complex systems requiring monitoring, control:
 - jet airplane
 - nuclear power plant
 - others?



"Network management includes the deployment, integration and coordination of the hardware, software, and human elements to monitor, test, poll, configure, analyze, evaluate, and control the network and element resources to meet the real-time, operational performance, and Quality of Service requirements at a reasonable cost."

Infrastructure for network management

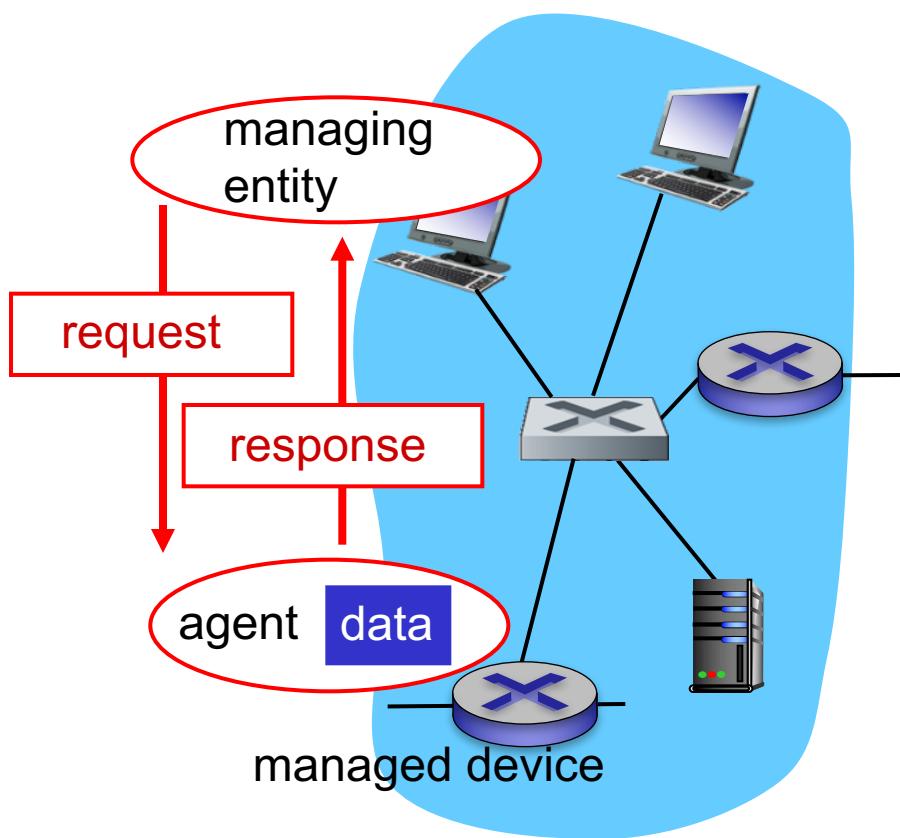
definitions:



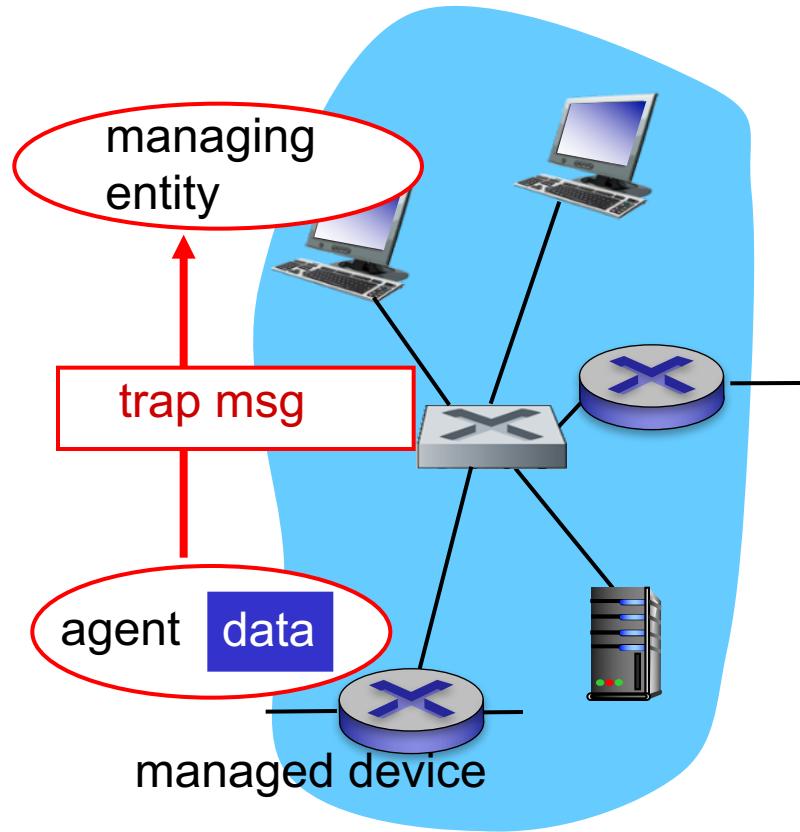
managed devices contain managed objects whose data is gathered into a Management Information Base (MIB)

SNMP protocol

Two ways to convey MIB info, commands:



request/response mode



trap mode

SNMP protocol: message types

Message type

GetRequest
GetNextRequest
GetBulkRequest

Function

manager-to-agent: “get me data”
(data instance, next data in list, block of data)

InformRequest

manager-to-manager: here’s MIB value

SetRequest

manager-to-agent: set MIB value

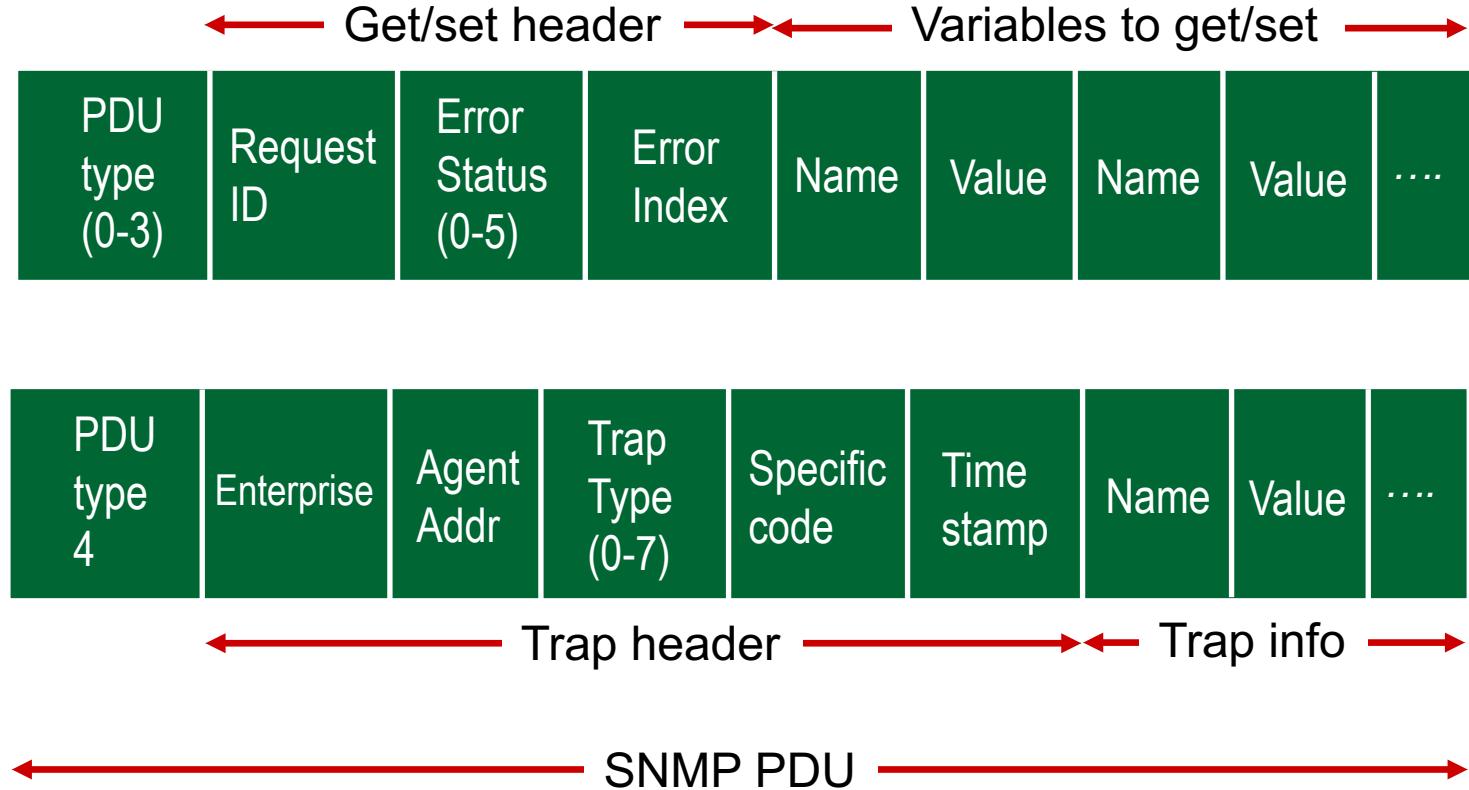
Response

Agent-to-manager: value, response to Request

Trap

Agent-to-manager: inform manager of exceptional event

SNMP protocol: message formats



More on network management: see earlier editions of text!

Chapter 5: summary

we've learned a lot!

- approaches to network control plane
 - per-router control (traditional)
 - logically centralized control (software defined networking)
- traditional routing algorithms
 - implementation in Internet: OSPF, BGP
- SDN controllers
 - implementation in practice: ODL, ONOS
- Internet Control Message Protocol
- network management

next stop: link layer!

Alma Mater Studiorum – University of Bologna
Department of Computer Science and Engineering

**Wireless Systems (1) –
Physical Systems, Signals and Wireless Systems' Design**



Luciano Bononi

(luciano.bononi@unibo.it)

Figure-credits: some figures have been taken from slides published on the Web, by the following authors (in alphabetical order):

J.J. Garcia Luna Aceves (ucsc), James F. Kurose & Keith W. Ross, Jochen Schiller (fub), Nitin Vaidya (uiuc)



Background on wireless PHY layer

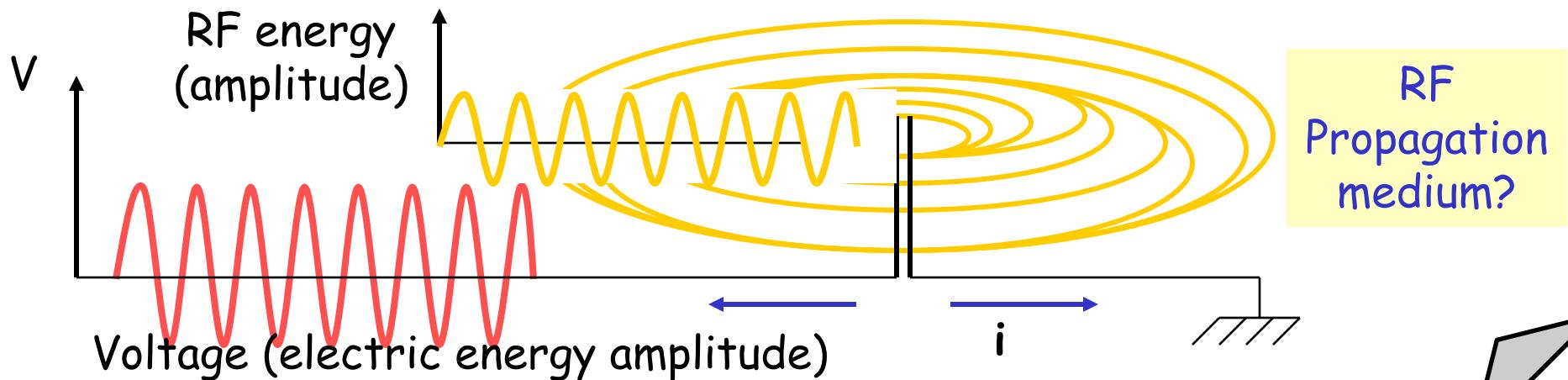
RF Properties

▪ Understanding Radio Frequency

- Generation, coverage and propagation issues
- Fundamental for wireless planning and management
- <http://faraday.physics.utoronto.ca/PVB/Harrison/Flash/EM/EMWave/EMWave.html>
- http://wwwhome.ewi.utwente.nl/~ptdeboer/ham/xnecview/dipole_anim.html

▪ Radio Frequency Signals

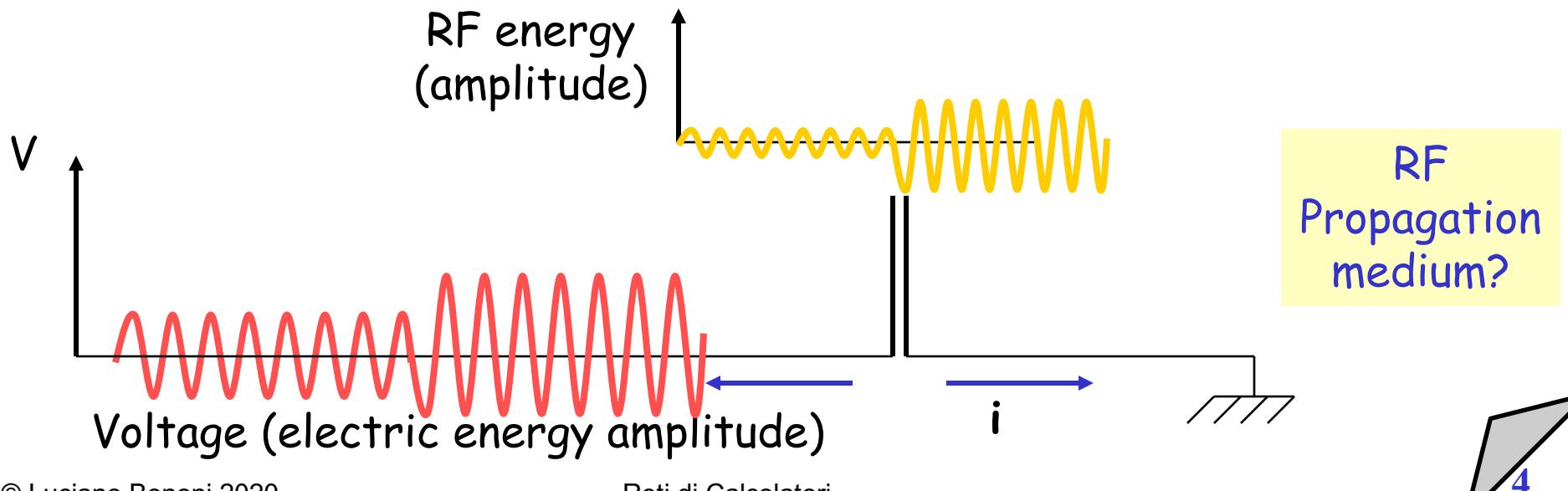
- Electromagnetic energy generated by high frequency alternate current (AC) in antennas
- Antenna: converts the wired current to RF and viceversa



RF Properties

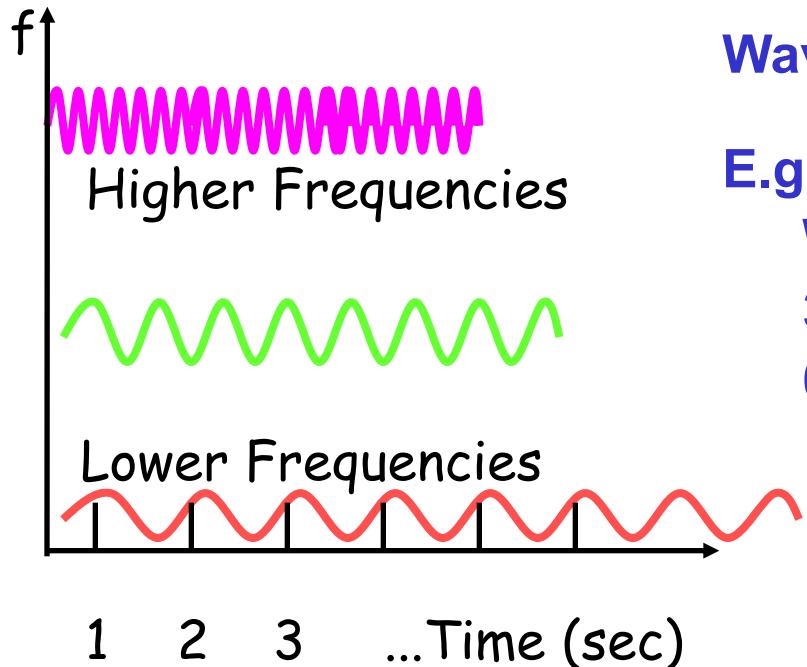
▪ Amplitude

- Higher amplitude RF signals go farther
- Transmission Power (Watts) = Energy / Time = Joule / Sec
 - More energy (voltage) moves more electrons (current)
 - Power = Voltage * Current



RF Properties

- **Frequency (and Wavelength)**
 - Wireless Spectrum (see next slides)
 - Portion of wireless spectrum regulated by regional authorities and assigned to wireless technologies



Wavelength = $c / \text{frequency}$

E.g. 2.4 GhZ (ISM band)

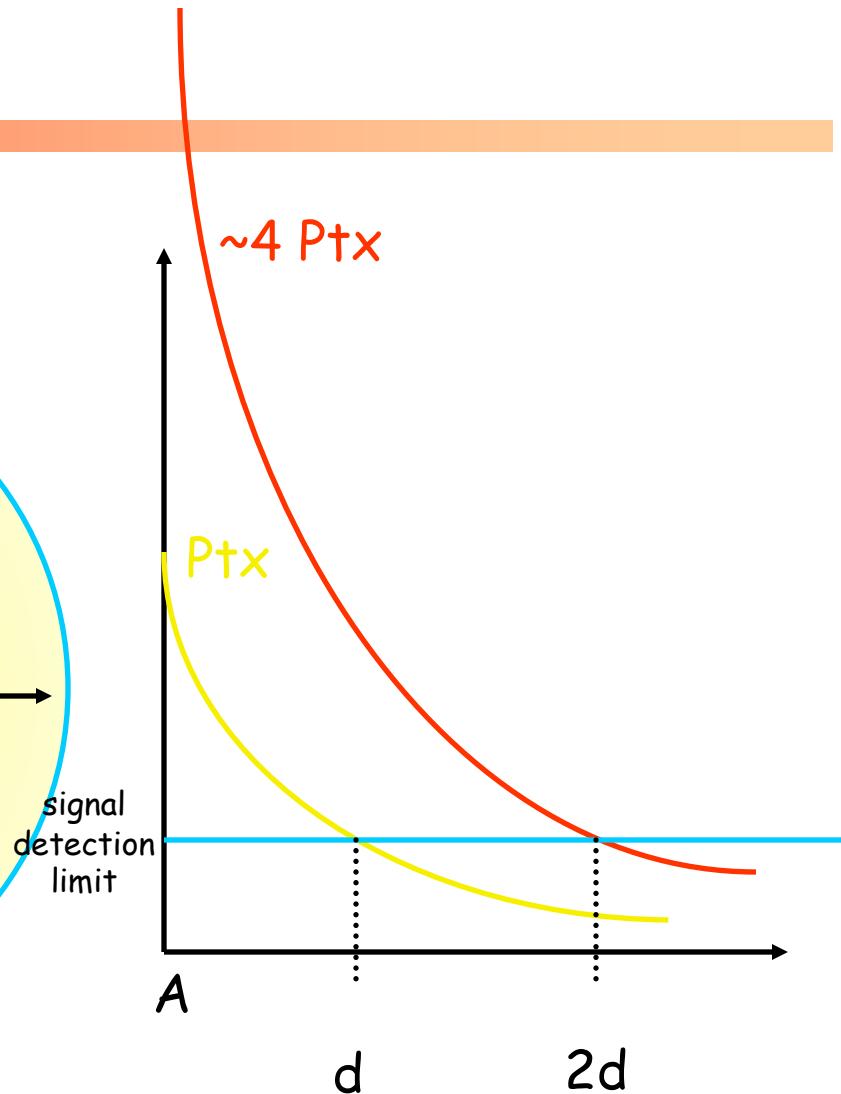
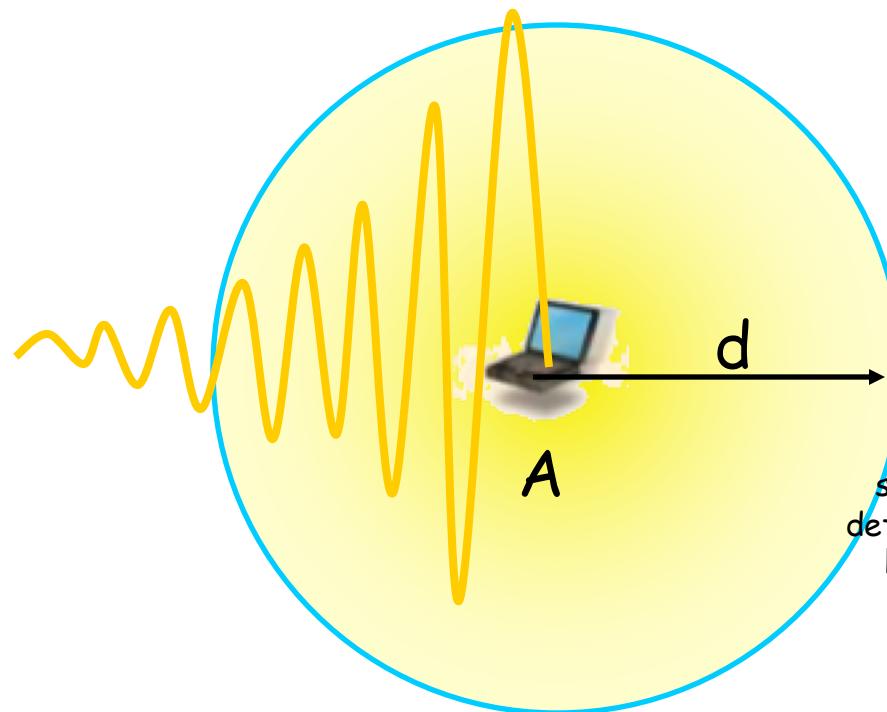
Wave Length =

$$300.000.000(\text{m/s}) / 2.400.000.000 \text{ Hz} = \\ 0.125 \text{ m} = 12.5 \text{ cm}$$

In practice:
Antennas work better
with size = $1, \frac{1}{2}, \frac{1}{4}$ of wavelength
(try to measure antenna size of
your IEEE 802.11 device)

RF propagation

- Radio transmission coverage

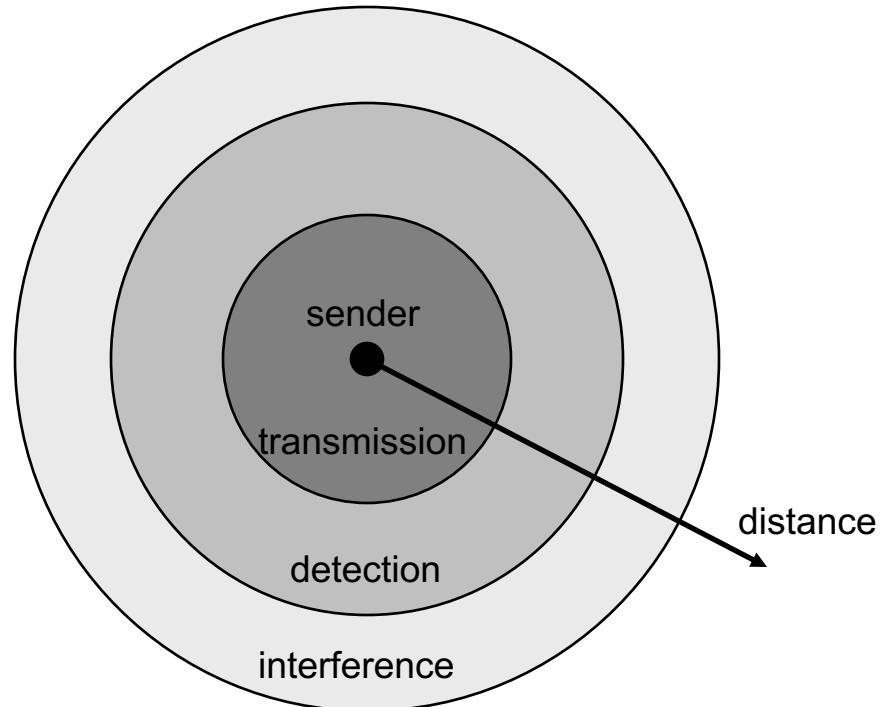


The range is a function of power transmission (Ptx)
Signal strength reduces with d^k \longrightarrow
($K \geq 2..3$, no obstacles, isotropic radiator)

In 3D, sphere:
 $V = (4 \pi r^3 / 3)$
 $S = (4 \pi r^2)$

Wireless signal propagation ranges

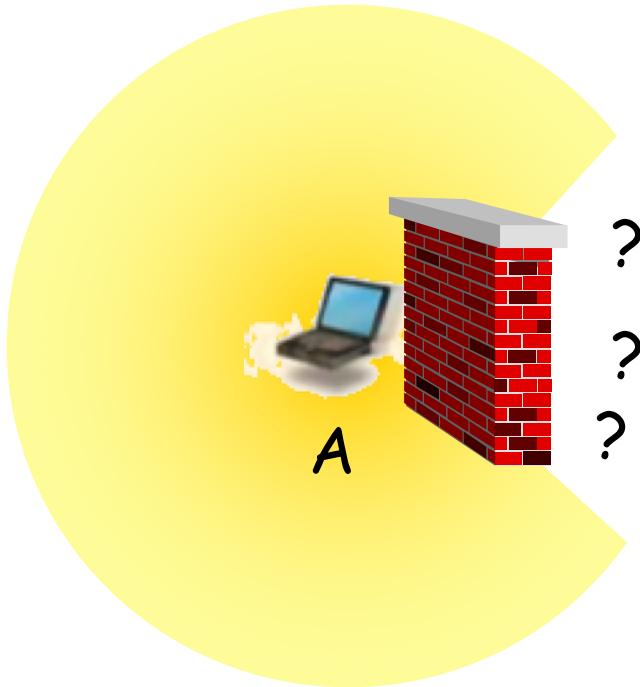
- **Transmission range**
 - communication possible
 - low error rate
- **Detection range**
 - detection of the signal possible
 - no communication possible
- **Interference range**
 - signal may not be detected
 - signal adds to the background noise



Ranges depend on receiver's sensitivity!

Wireless networks' technology

▪ Radio transmission coverage



obstacles can reflect or absorb waves
depending on materials and wave frequencies

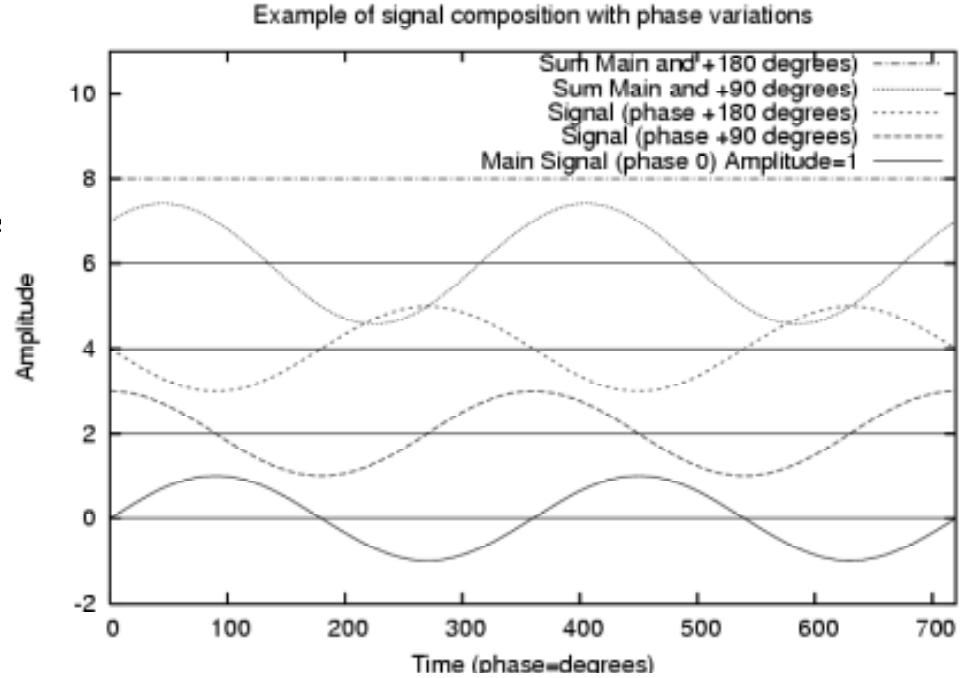
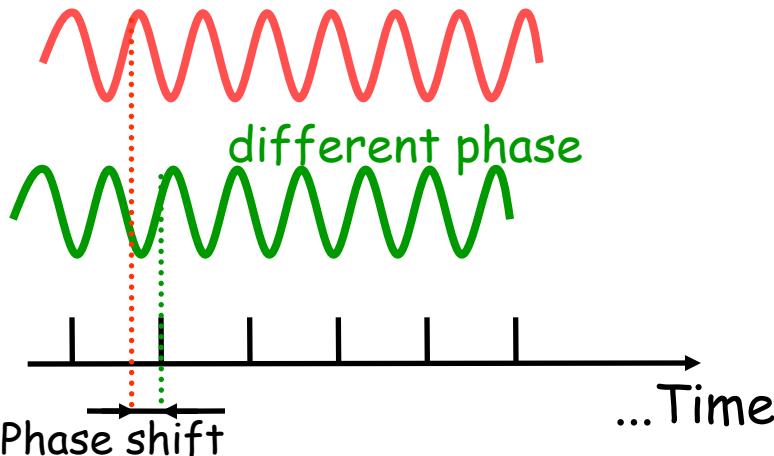
Rules of thumb:

- high frequencies are good for short distances and are affected by obstacles
- low frequencies are good for long distances and are less affected by obstacles

RF Properties

Phase: shift of the wave (in degrees or radians)

- Positive phase (left-shift), early wavefront
- Negative phase (right-shift), late wavefront

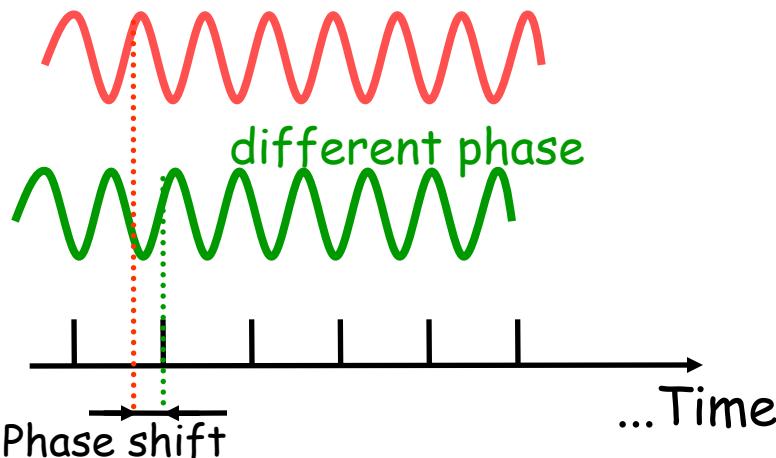


In practice:
RF echoes arriving at receivers with different phase may have positive or negative effects... Why?

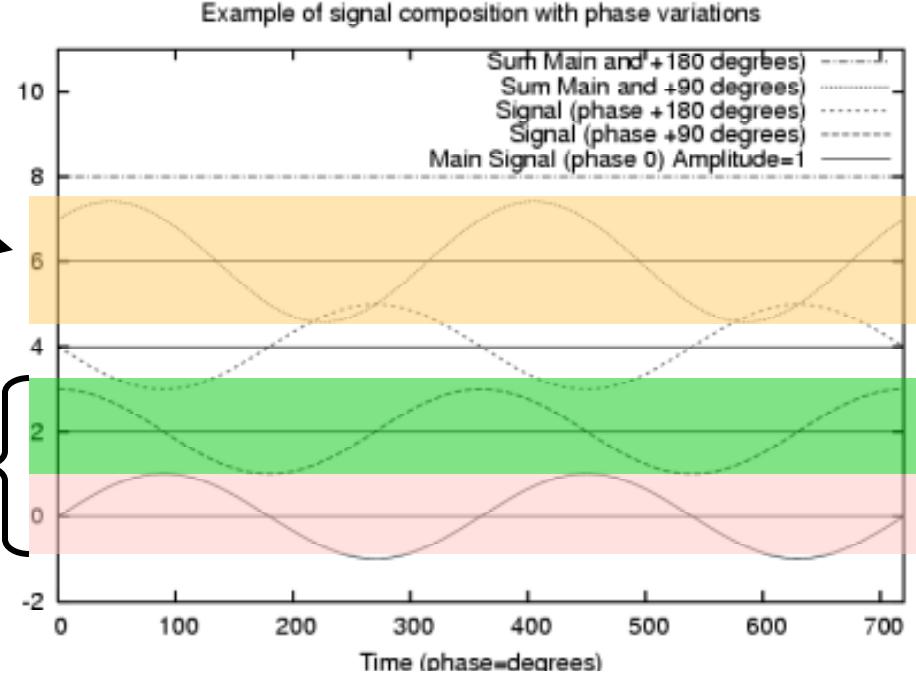
RF Properties

Phase: shift of the wave (in degrees or radians)

- Positive phase (left-shift), early wavefront
- Negative phase (right-shift), late wavefront



+



In practice:
RF echoes arriving at receivers with different phase may have positive or negative effects... Why?

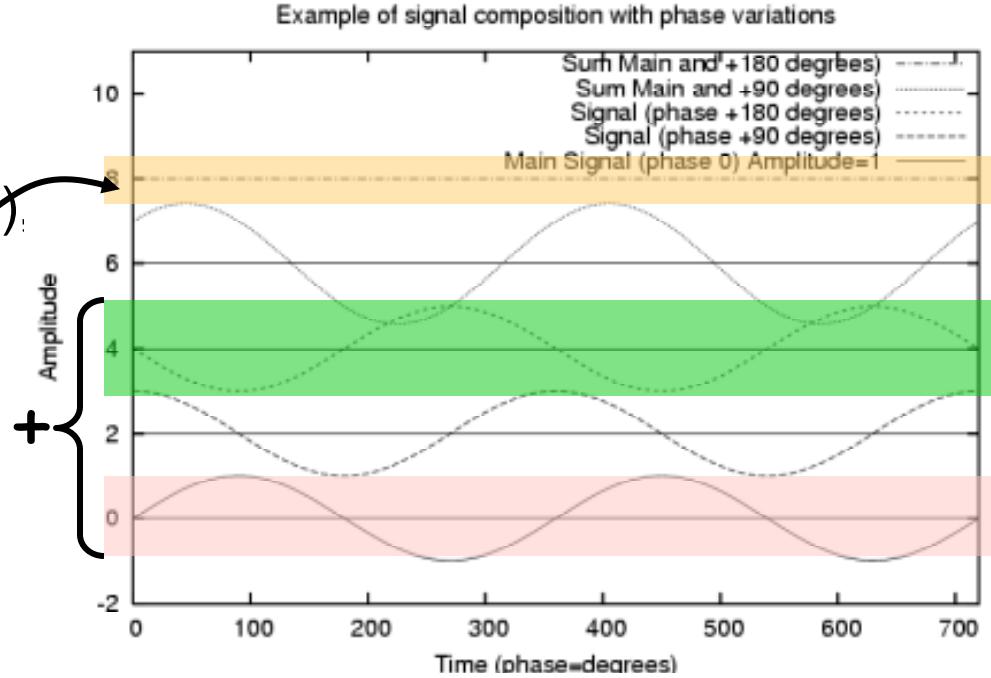
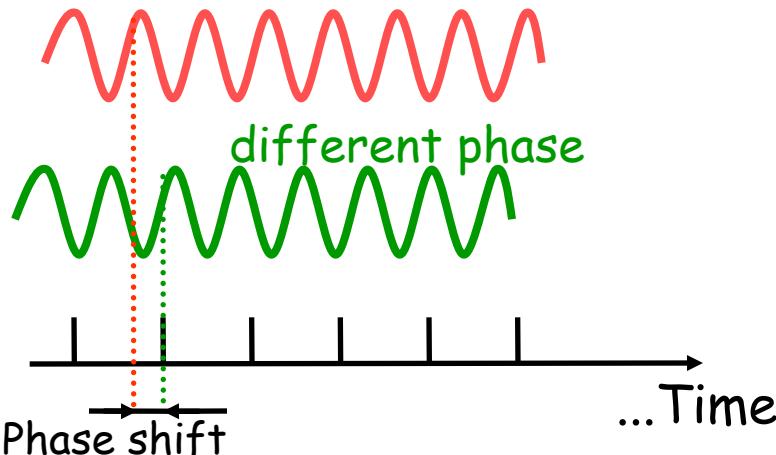
RF Properties

Phase: shift of the wave (in degrees or radians)

- Positive phase (left-shift), early wavefront
- Negative phase (right-shift), late wavefront

Test with audio (demo):

<https://www.szynalski.com/tone-generator/>

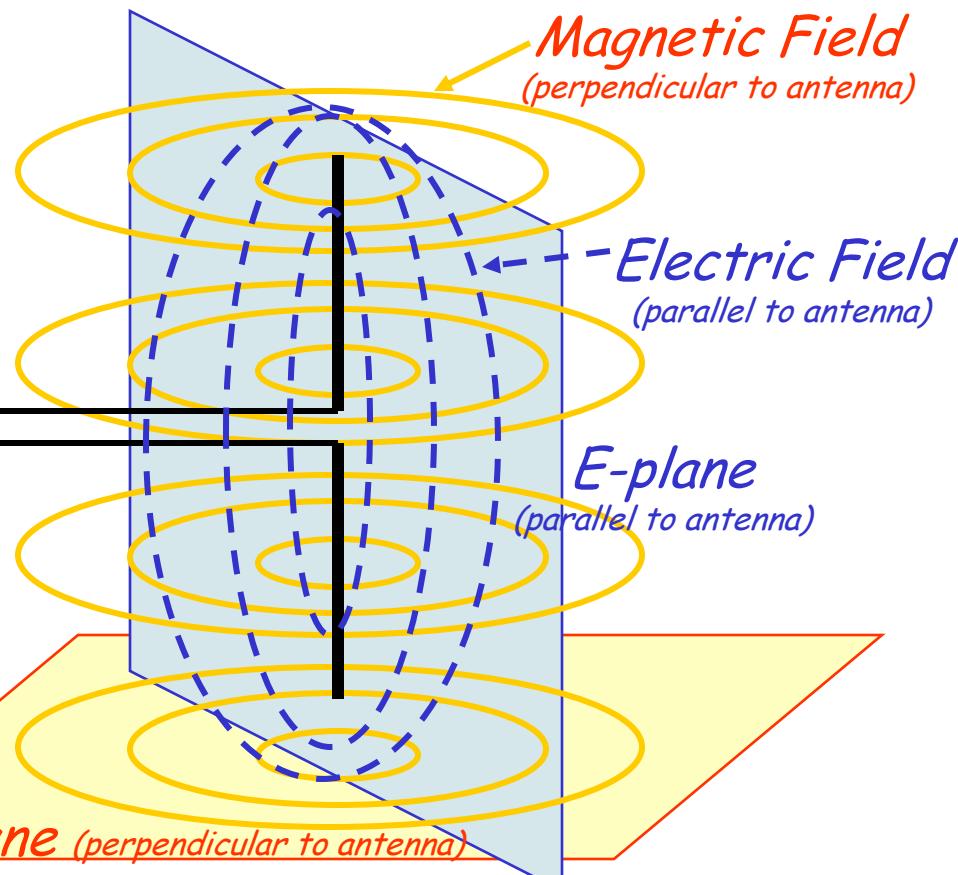


In practice:
RF echoes arriving at receivers with different phase may have positive or negative effects... Why?

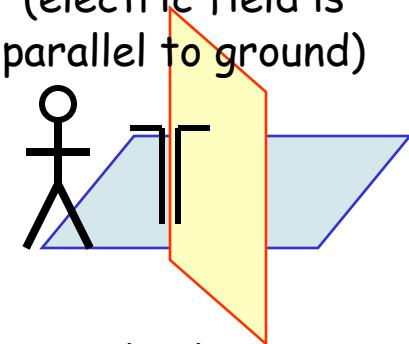
RF Properties

Polarization: (physical orientation of antenna)

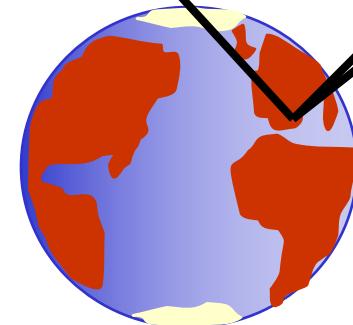
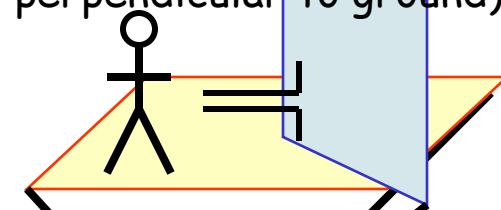
- RF waves are made by two perpendicular fields:
 - Electric field and Magnetic field



Horizontal Polarization
(electric field is parallel to ground)



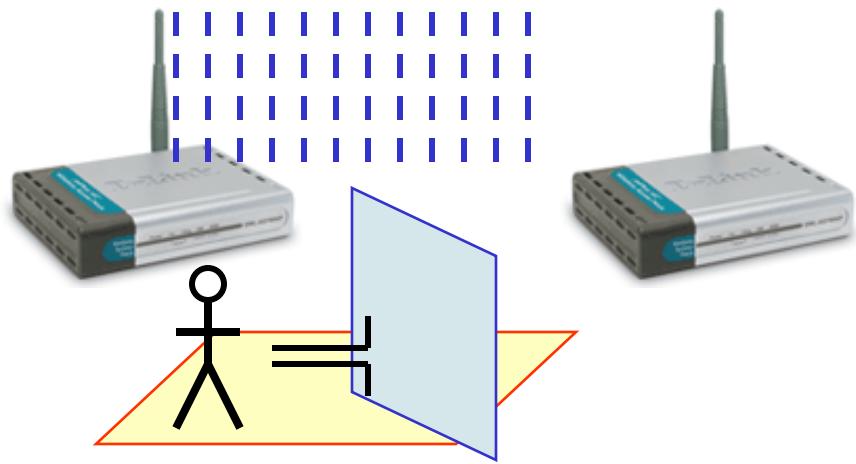
Vertical Polarization
(electric field is perpendicular to ground)



RF Properties

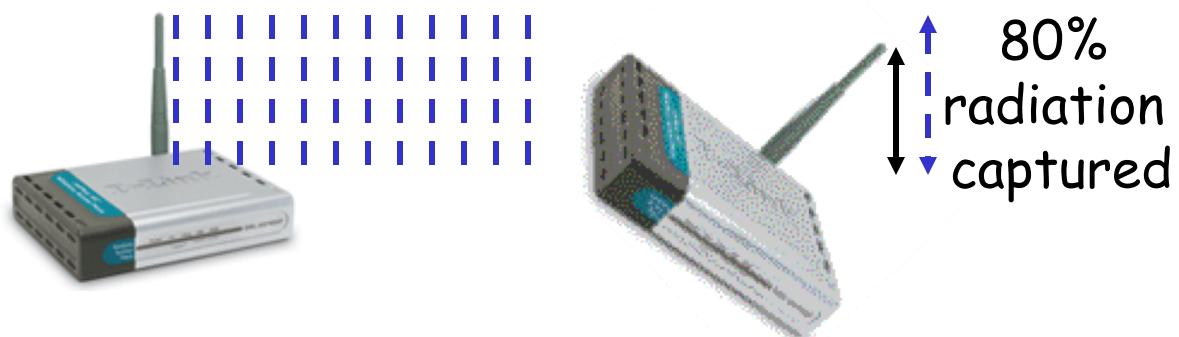
Vertical Polarization: typically used in WLANs

OK Transferred radiation OK



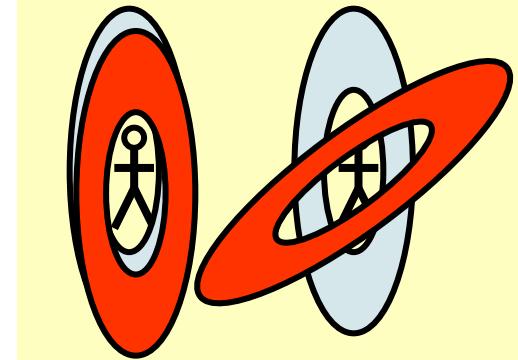
100%
radiation
captured

OK Transferred radiation NO

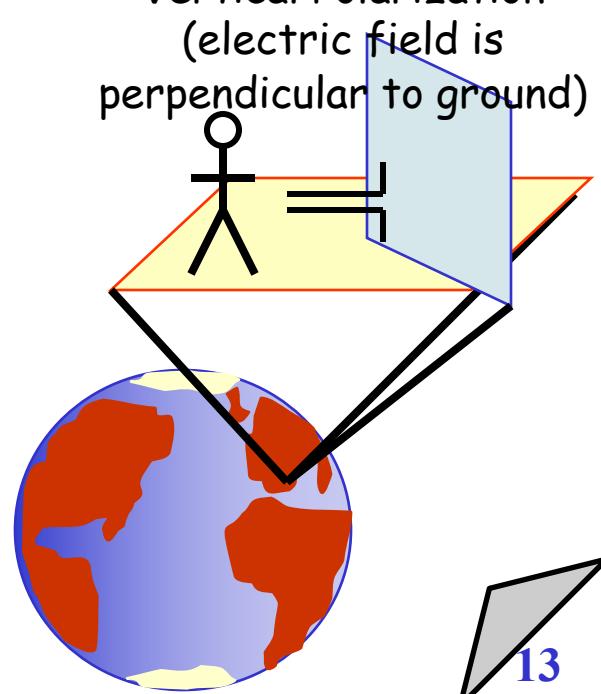


80%
radiation
captured

Intuitively....



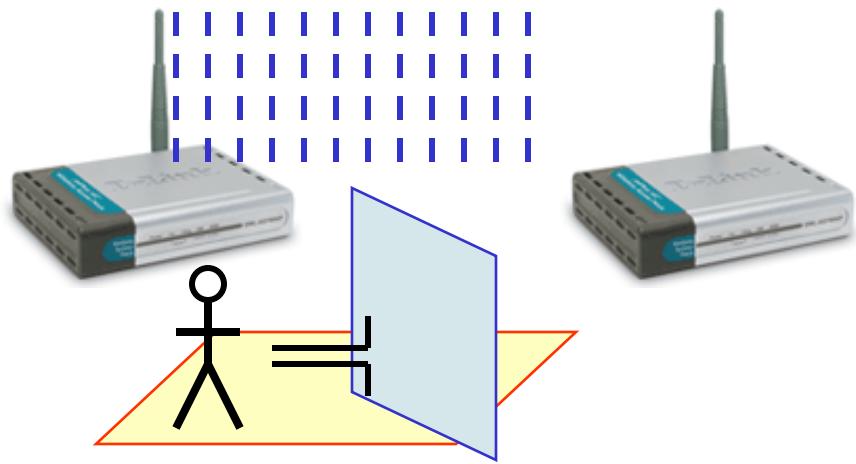
Vertical Polarization
(electric field is
perpendicular to ground)



RF Properties

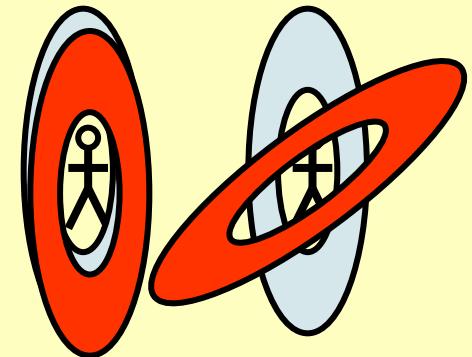
Vertical Polarization: the PCMCIA device problem

OK Transferred radiation OK

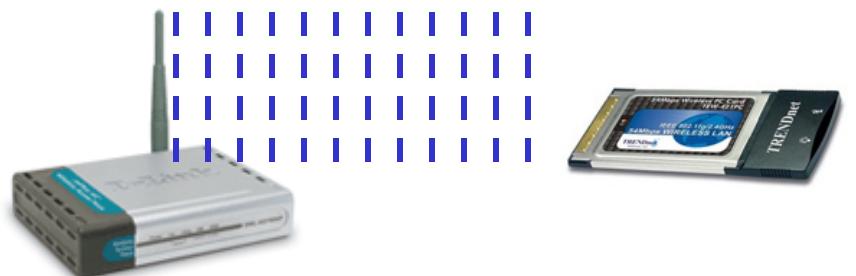


100%
radiation
captured

Intuitively....



OK Transferred radiation NO



N.B. the polarization problem
is very much important when
using distant devices and
directional antennas. With short
distances signal reflections help!

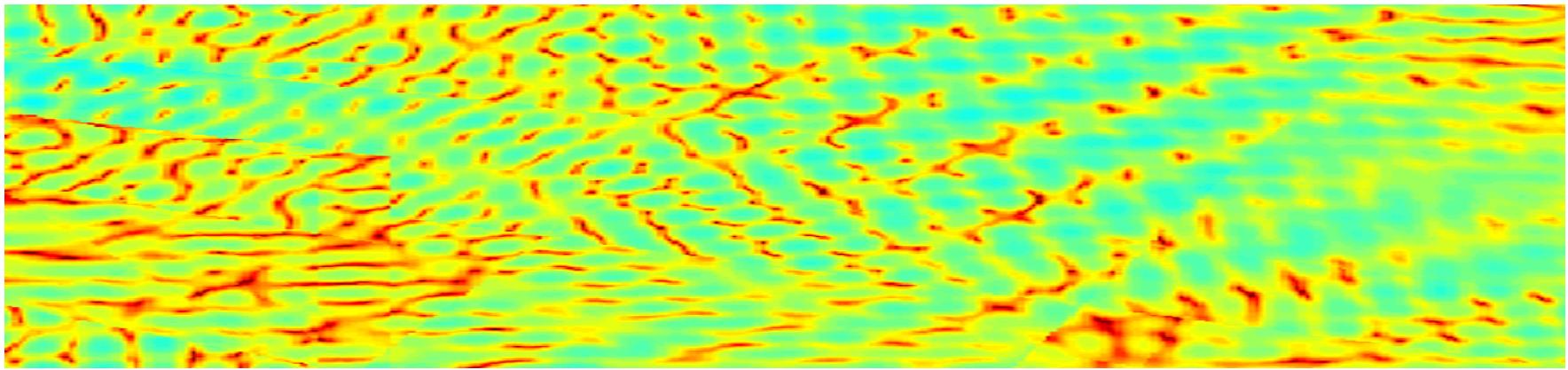
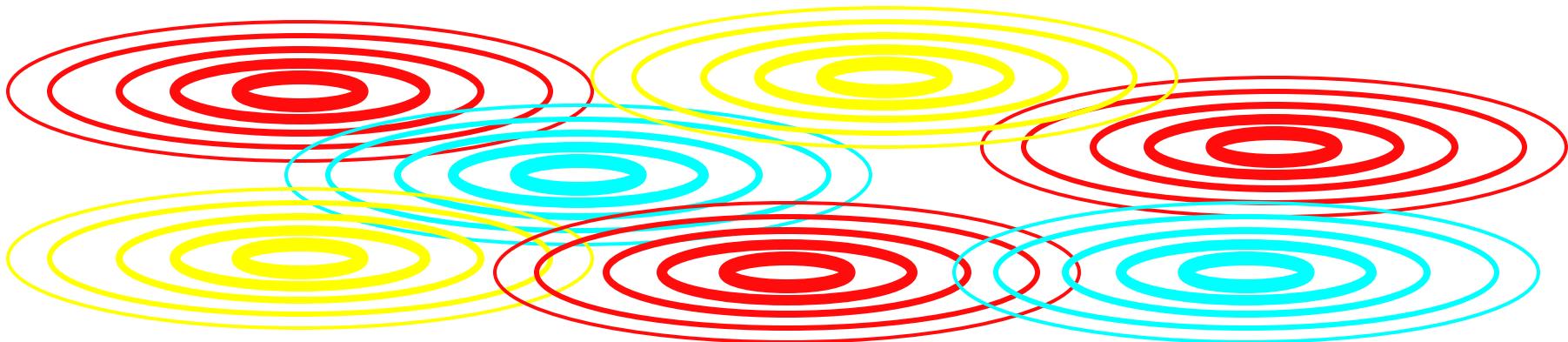
??%
radiation
captured



RF Behaviors

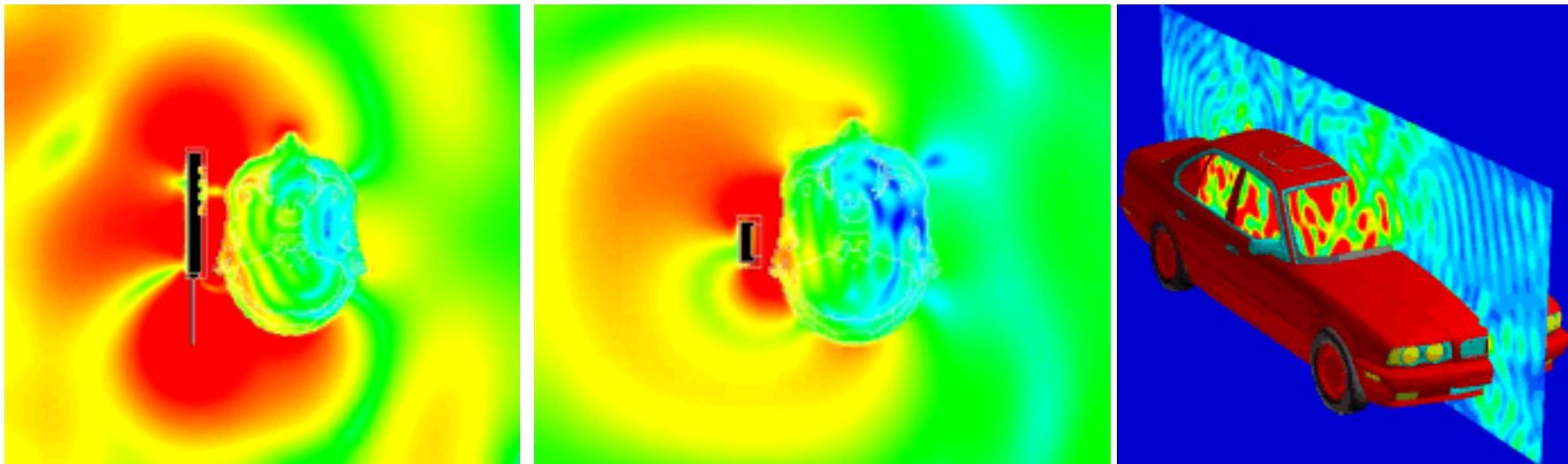
- Radio transmission interference

<http://www.met.rdg.ac.uk/clouds/maxwell/>



RF Behaviors

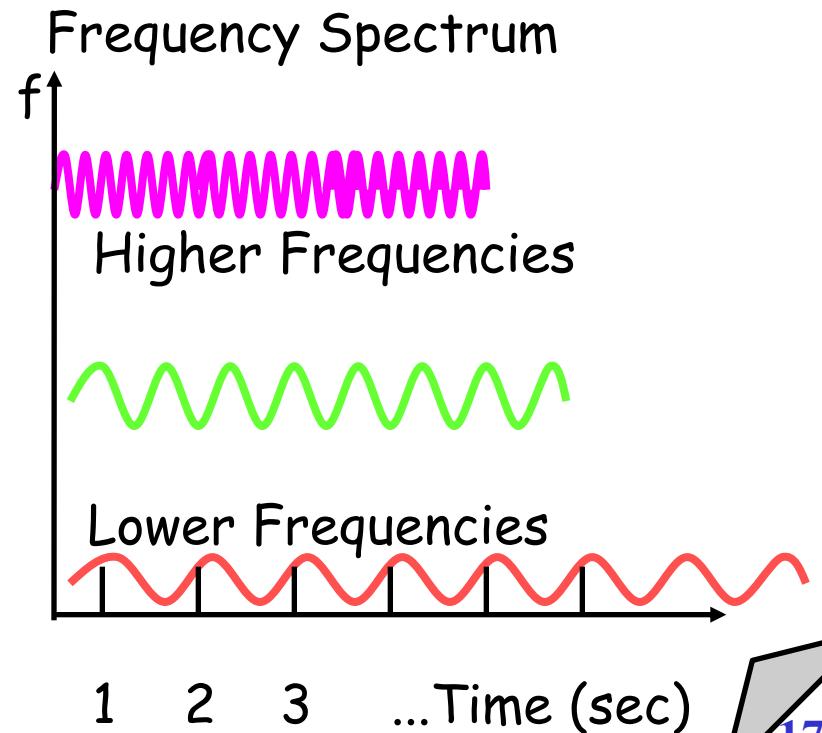
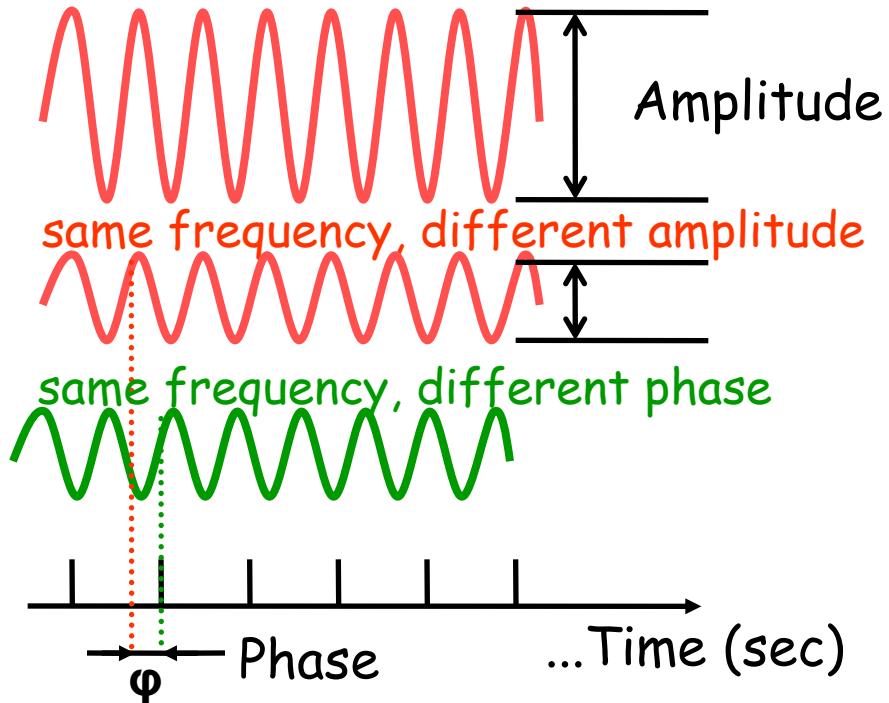
- Radio effects on human head (do not try this at home ☺)
<http://temf.de/Radiation-of-a-mobile-phone-P.58.0.html?&L=1>
- Credits: Technische Universität Darmstadt, Computational Electromagnetics Laboratory



Wireless transmission: Electromagnetic waves

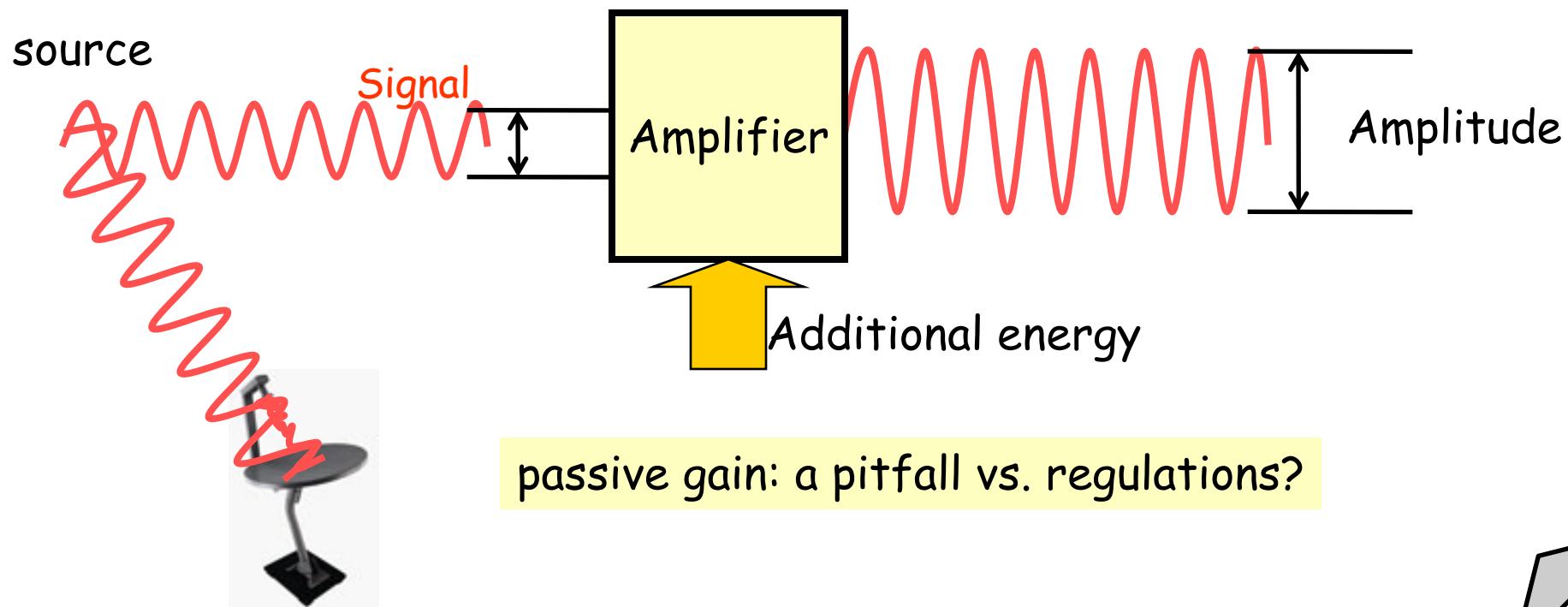
- Different parameters of electromagnetic waves:

- amplitude **M** proportional to transmission energy (loudness)
- frequency **f** (tone) measured in Hertz (Cycle/sec)
- phase **φ** (peak shift with respect to reference signal) (rad)



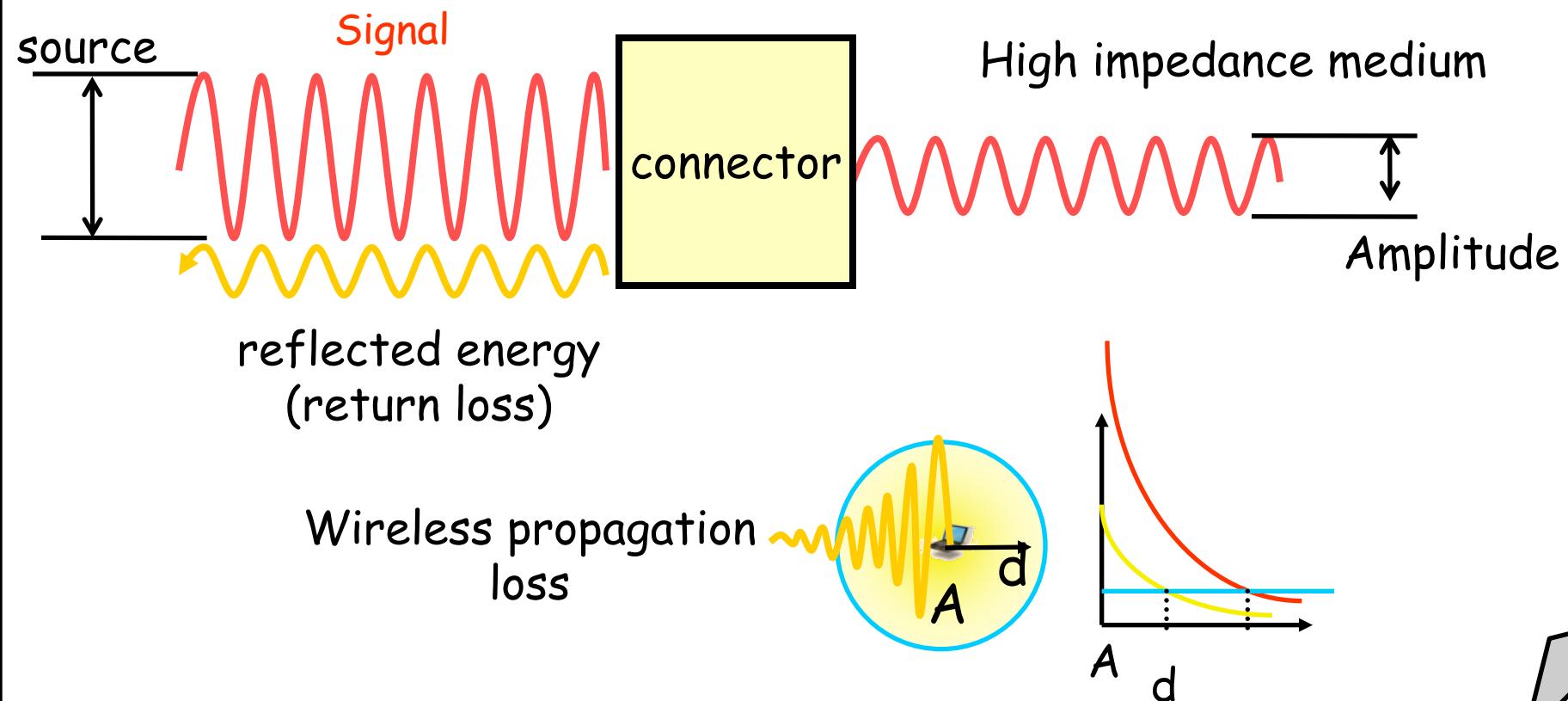
Wireless transmission

- **Signal Gain: (measured in Decibels, Db)**
 - Increase in amplitude **M** proportional to transmission energy
 - Active gain (amplifiers)
 - Passive gain (antennas focusing signal energy, and additive signal effects)



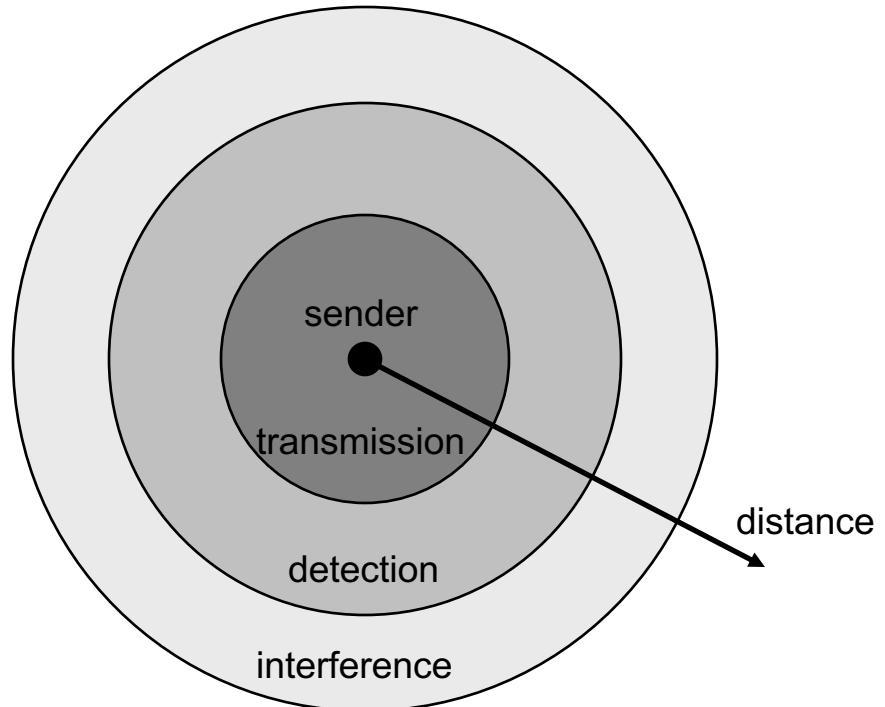
Wireless transmission

- **Signal Loss: (Db)**
 - Decrease in amplitude **M** proportional to energy waste
 - Intentional (resistance, signal attenuation -> heat)
 - Obstacles, e.g. (walls, water for 2.4 Ghz) and distance (wireless)



Wireless signal propagation ranges (reprise)

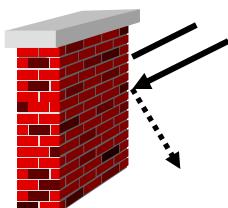
- **Transmission range**
 - communication possible
 - low error rate
- **Detection range**
 - detection of the signal possible
 - no communication possible
- **Interference range**
 - signal may not be detected
 - signal adds to the background noise



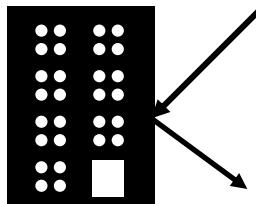
Ranges depend on receiver's sensitivity!

Wireless Signal propagation effects

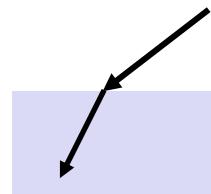
- Propagation in free space always like light (straight line)
- Receiving power proportional to $1/d^2$
(d = distance between sender and receiver)
- Receiving power additionally influenced by
 - fading (frequency dependent)
 - shadowing
 - reflection at large obstacles
 - refraction depending on the density of a medium
 - scattering at small obstacles
 - diffraction at edges



shadowing



reflection



refraction
Reti di Calcolatori

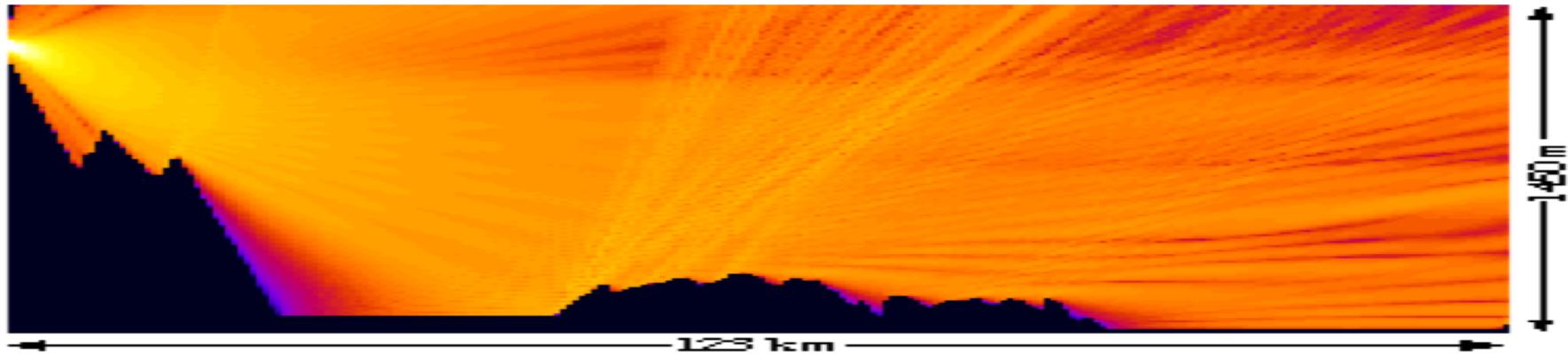


scattering

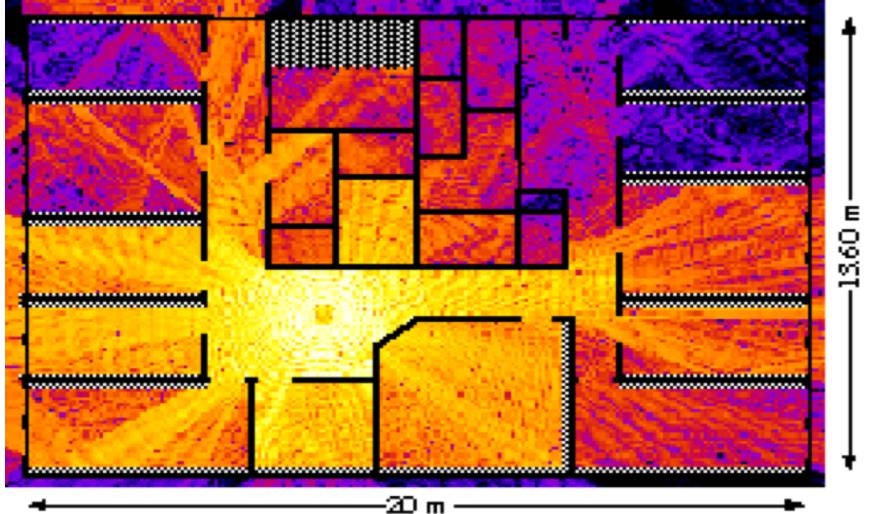


diffraction

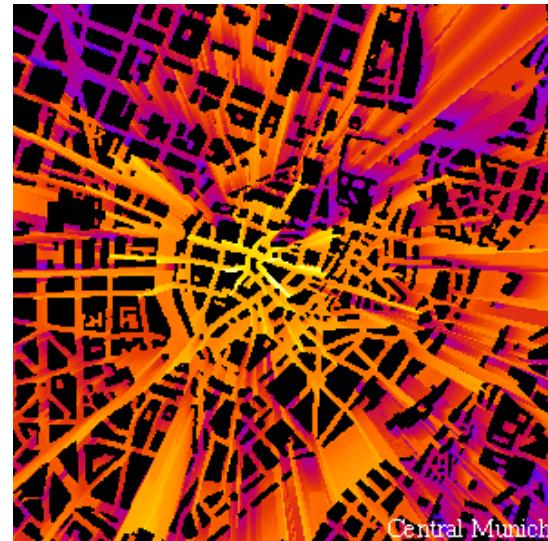
Real world example



Raytracing examples



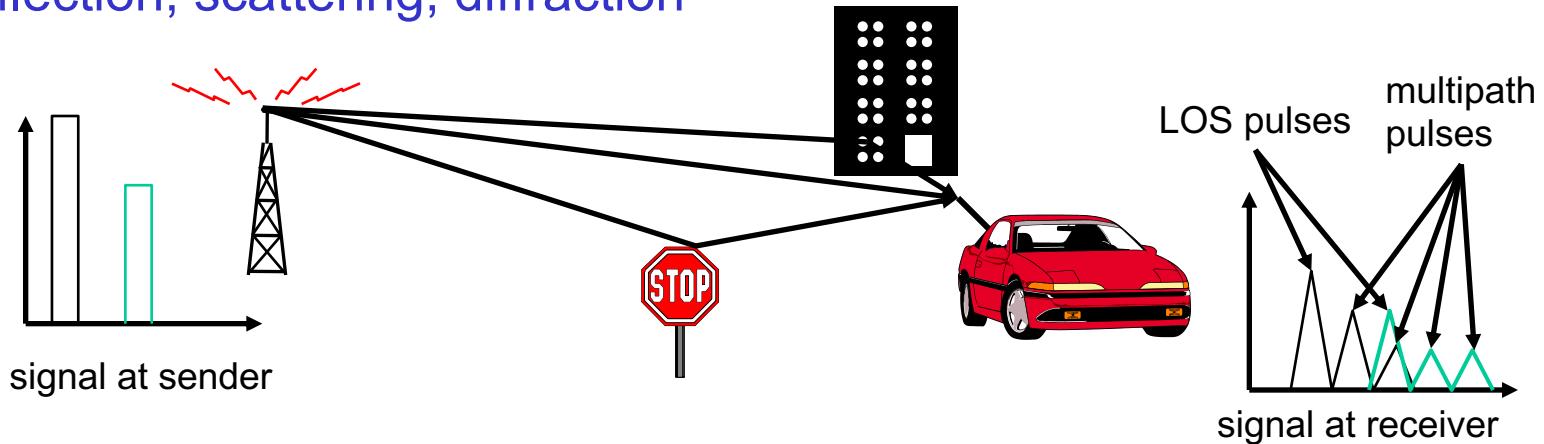
Low signal



high signal

Multipath propagation

- Signal can take many different paths between sender and receiver due to reflection, scattering, diffraction



- Time dispersion: signal is dispersed over time
→ interference with “neighbor” symbols, Inter Symbol Interference (ISI)
- The signal reaches a receiver directly and phase shifted
→ distorted signal depending on the phases of the different parts

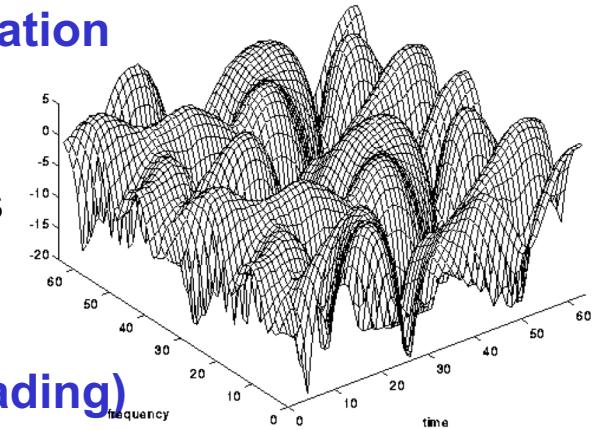
Effects of mobility

- **Channel characteristics change over time and location**

- signal paths change
- different delay variations of different signal parts
- different phases of signal parts

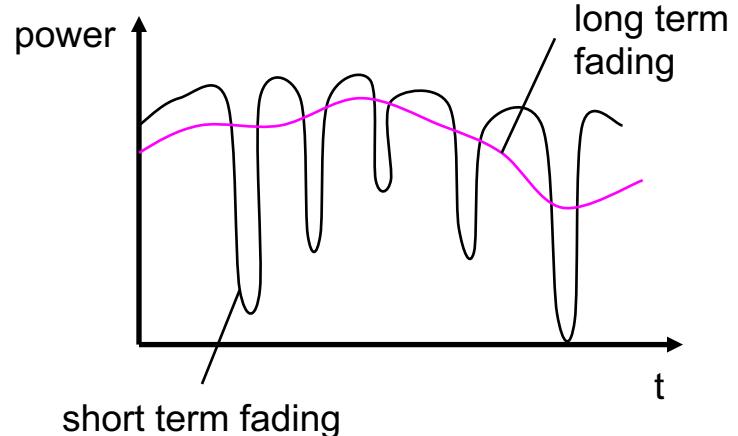
→ quick changes in the power received (short term fading)

<http://www.sps.ele.tue.nl/members/j.p.linnartz/web/reference/chaptr03/rayjava/rayjava.htm>



- **Additional changes in**

- distance to sender
- obstacles further away

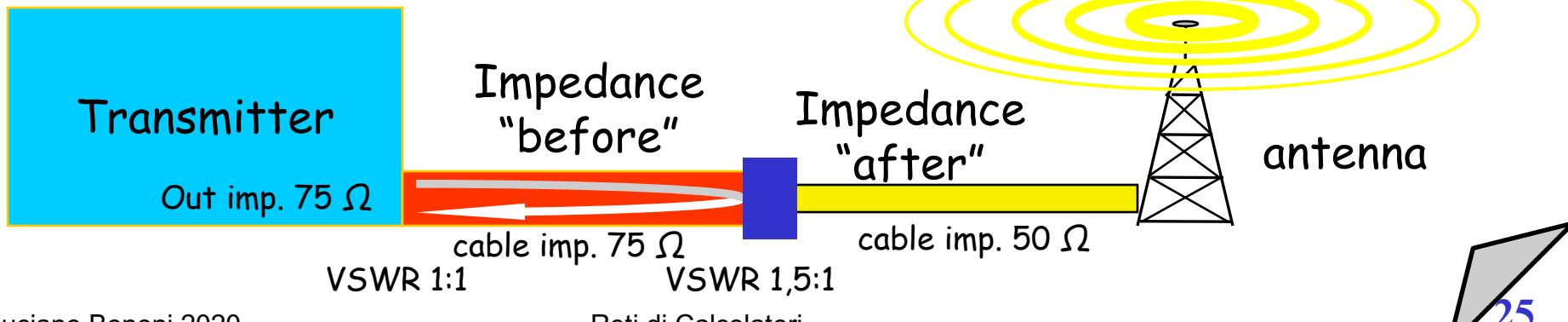


→ slow changes in the average power received (long term fading)

Voltage Standing Wave Ratio (VSWR)

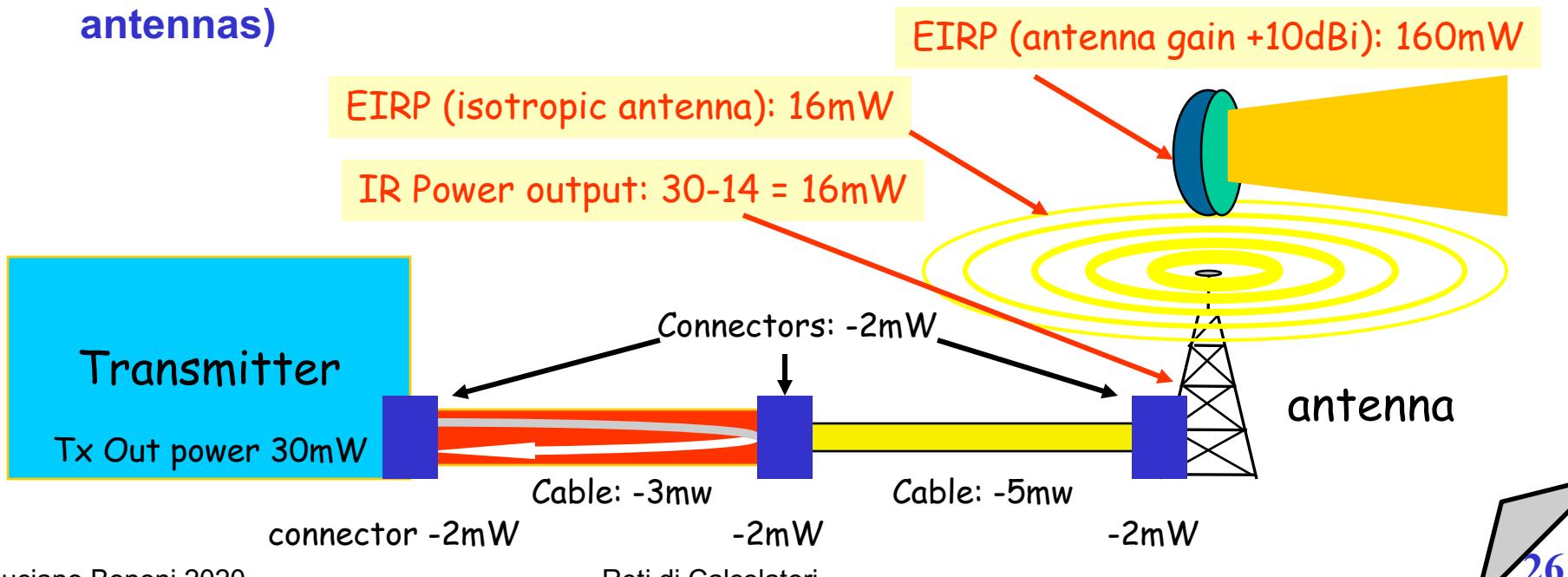
- **VSWR occurs with different impedance (Ohm) = resistance to AC current flow between transmitter and antenna**
 - VSWR is the cause of “return loss” energy towards the transmitter
 - Measured as ratio between impedance (before and after)
 - E.g. 1,5:1 (impedance ratio before/after is 1,5 times the ideal value)
 - 1 = normalized ideal impedance (1:1 means perfect VSWR)
 - VSWR Causes burnout of transmitter circuits, and unstable output levels

VSWR solution:
always use same impedance
circuits, cables, connectors
(typical 50Ω in LANs)



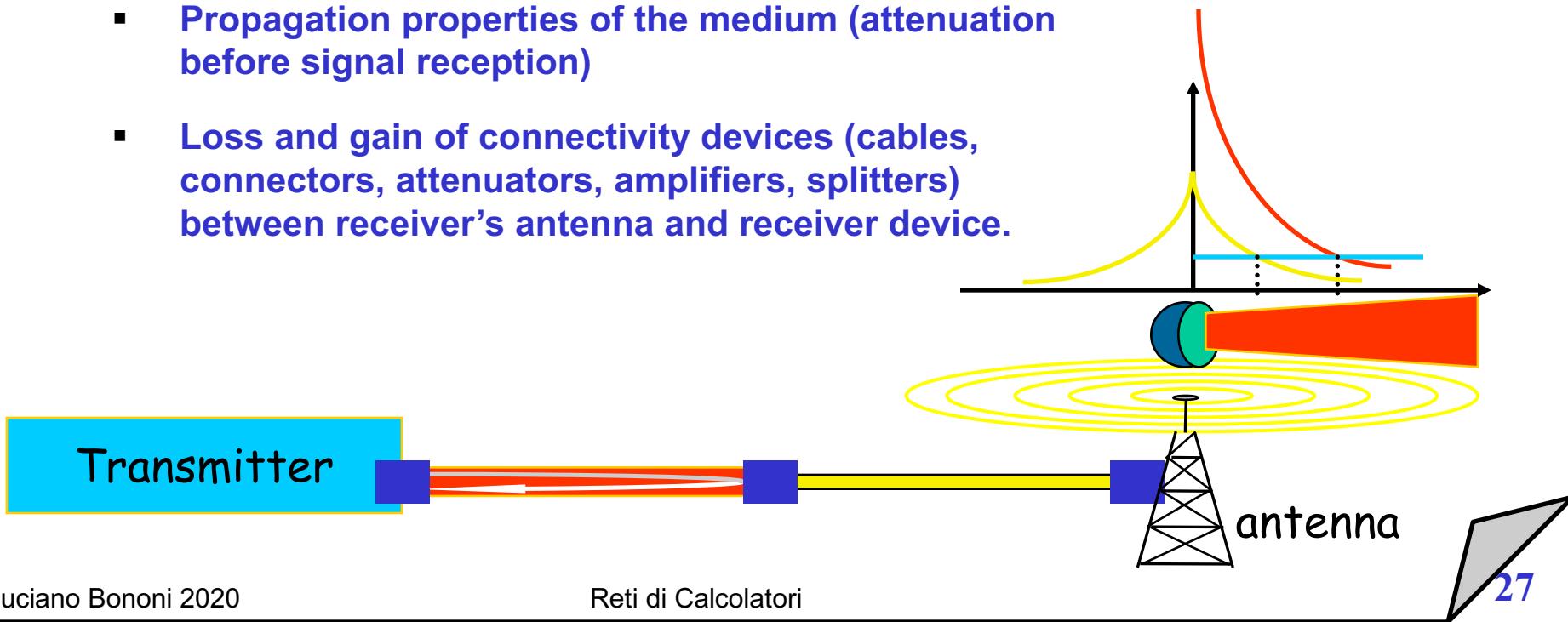
Intentional radiator and EIRP

- (Intentional) radiator: (def.) RF device specifically designed to generate and radiate RF signals.
 - ...Includes Tx RF device, cables and connectors (antenna excluded)
 - IR Power output: (subject to regulations) is the power output of last connector just before the antenna
- Equivalent Isotropically Radiated Power (EIRP): the power radiated by the antenna (including the passive antenna gain effect of directional antennas)



System design (under power viewpoint)

- Many factors must be considered in the design of Wireless systems:
 - Power of transmitting device
 - Loss and gain of connectivity devices (cables, connectors, attenuators, amplifiers, splitters) between transmission device and transmitter's antenna
 - Power of the intentional radiator (last connector just before antenna)
 - Power radiated by antenna element (EIRP)
 - Propagation properties of the medium (attenuation before signal reception)
 - Loss and gain of connectivity devices (cables, connectors, attenuators, amplifiers, splitters) between receiver's antenna and receiver device.



Power measurement

- **WATT:** electric power unit
 - **1 Watt = 1 Ampere * 1 Volt (P=V*I)** also $P = R \cdot I^2$ and $P = L/t$
 - **Current (ampere)** is the amount of charge (electrons) flowing as current in a wire
 - **Voltage (Volt)** is the “pressure” applied to the flow of charge
 - **Resistance (impedance)** is the obstacle to current flow
 - **Power is the energy needed (in a given time unit) to apply a given “pressure” to a given “amount of charge”, by resulting in a flow of current.**
 - **Watt and dBm** are units used for absolute power measurement
 - **Typical RF power for WLANs:**
 - **AP: 30..100 mW (up to 250 mW outdoor), PCMCIA: 15..30 mW**

Power measurement

- Decibel (dB): a power measurement unit designed to express power loss
 - It is more practical to use given the logarithmic decay of wireless signals
 - It allows to make easy calculations on “resulting power”
- Decibel (dB) measures the logarithmic relative strength between two signals (mW are a linear absolute measure a energy)
 - $\text{Log}_{10}(X) = Y \iff 10^Y = X$
 - $1 = 10^0, \log_{10} (1) = 0$
 - $10 = 10^1, \log_{10} (10) = 1$
 - $100 = 10^2, \log_{10} (100) = 2$
 - $1000 = 10^3, \log_{10} (1000) = 3$
- How strong is a 10 dB signal? (it depends on the reference signal)

Exponential growth

Linear growth
“BEL” units (B)

Power measurement

- Decibel (dB): 1/10 of a Bel
- E.g. 1000 is one Bel greater than 100 => 1000 is 10 dB greater than 100

Linear
signal
difference
(factor)

- $1 = 10^0, \log_{10}(1) = 0$
 - $10 = 10^1, \log_{10}(10) = 1$
 - $100 = 10^2, \log_{10}(100) = 2$
 - $1000 = 10^3, \log_{10}(1000) = 3$
-

- How strong is a 10 dB signal? (it depends on the reference signal)
 - Positive dB value is power gain, negative dB value is power loss
 - e.g. given 7 mW power, a +10 dB signal gain is 70 mW
 - e.g. given 7 mW power, a -10 dB signal gain (loss) is 0.7 mW
- Power Difference (in dB) between Tx and Rx signal:
 - Power Difference (dB) = $10 * \log(\text{Power Rx(Watt)} / \text{Power Tx (Watt)})$
- Gain and Loss are relative power measurements: dB is the unit

Power measurement

- Advantage of dB: what is better?
 - E.g.: A signal transmitted at [TX] 100 mW is received at [RX] 0.000005 mW
 - Power Difference (dB) = $10 * \log([\text{RX}] / [\text{TX}]) = 10 * \log(0.000005\text{mW}/100\text{mW}) = -73$
 - A signal transmitted at 100 mW is received with gain (loss) -73 dB
- Advantage of dB: what is better?
 - E.g.: A signal transmitted at 100 mW is received at 0.000005 mW, then it is amplified (*100) to 0.0005 mW ???
 - A signal transmitted at 100 mW is received with gain (loss) $-73 + 20 = -53$ dB

-3 dB	½ power in mW (/ 2)
+3 dB	2x power in mW (* 2)
-10 dB	1/10 power in mW (/ 10)
+10 dB	10x power in mW (* 10)

Approximated table (values defined for ease of calculations)

Power measurement

- Practical example:
 - Signal Tx at 100 mW, cable **-3dB** loss, amplifier **+10 dB** gain
 - $100 \text{ mW} / 2 \text{ (-3dB)} = 50 \text{ mW} * 10 \text{ (+10 dB)} = 500 \text{ mW}$ IR power output
 - Signal TX at **30 mW** is received at the antenna as **6 mW (2/10 of TX power)**
 - Intentional Radiator Gain (loss) = $30 \text{ mW} / 10 = 3 \text{ mW} * 2 = 6 \text{ mW}$
 - Intentional Radiator Gain (loss) = $-10 \text{ dB} + 3 \text{ dB} = -7 \text{ dB}$ ($\approx 1/5$, $7 \text{ dB} \approx 5x$)
- N.B. **dBs are additive measures** of gain (loss): e.g. $6 \text{ dB} = +3+3 \text{ dB}$, $7 \text{ dB} = 10-3 \text{ dB}$
 - E.g. $100 \text{ mW} -6 \text{ dB} = 100 \text{ mW} -3 -3 \text{ dB} = 100 / 2 / 2 = 25 \text{ mW}$
 - E.g. $100 \text{ mW} +7 \text{ dB} = 100 \text{ mW} +10 -3 \text{ dB} = 100 * 10 / 2 = 500 \text{ mW}$
 - E.g. $10 \text{ mW} + 5 \text{ dB} = 10 \text{ mW} (+10+10-3-3-3-3) \text{ dB} = 1000/32 = 31.25 \text{ mW}$
 - E.g. $10 \text{ mW} + 11 \text{ dB} = ?$
 - E.g. $50 \text{ mW} - 8 \text{ dB} = ?$

N.B. Approximated values (values defined for ease of calculations)

Power measurement

- **dBm: dB-milliWatt, the absolute measure of signal power**
 - Assumption: reference signal is 1 mW = 0 dBm(normalization factor)
 - Useful for gain/loss calculation without passing through mW
 - E.g. access point transmits 100 mW = 1mW (*10*10) = +20 dBm
 - PCMCIA card transmits at 30 mW = 1mW (*10*3) = +14.7 dBm
 - E.g. Tx= 30 mW, cable -2 dB, amplifier +9 dB:
 - $30 \text{ mW} = 1\text{mW} *10 *3 = 14.7 \text{ dBm}$
 - IR power : $14.7 \text{ dBm} -2\text{dB} +9\text{dB} = 21.7 \text{ dBm (147.91 mW)}$
 - In general, for converting mW to dBm and viceversa:
 - $P_{\text{dBm}} = 10 \log(P_{\text{mW}})$ and $P_{\text{mW}} = 10^{(P_{\text{dBm}} / 10)}$

Power measurement

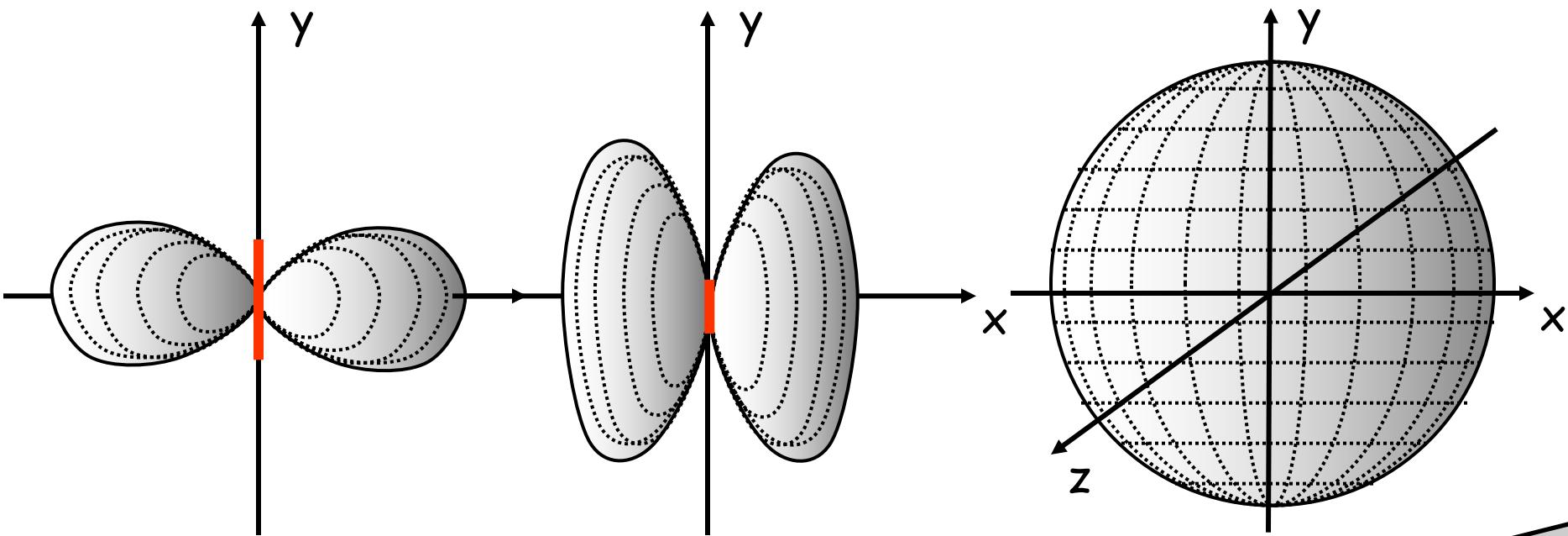
- mW - dBm: conversion table

-40 dBm	-30 dBm	-20 dBm	-10 dBm	0 dBm	+10 dBm	+20 dBm	+30 dBm	+40 dBm
100 nW	1 μ W	10 μ W	100 μ W	1 mW	10 mW	100 mW	1 W	10 W

-12 dBm	-9 dBm	-6 dBm	-3 dBm	0 dBm	+3 dBm	+6 dBm	+7 dBm	+9 dBm	+12 dBm
62,5 μ W	125 μ W	200 μ W	250 μ W	500 μ W	1 mW	2 mW	4 mW	5 mW	8 mW

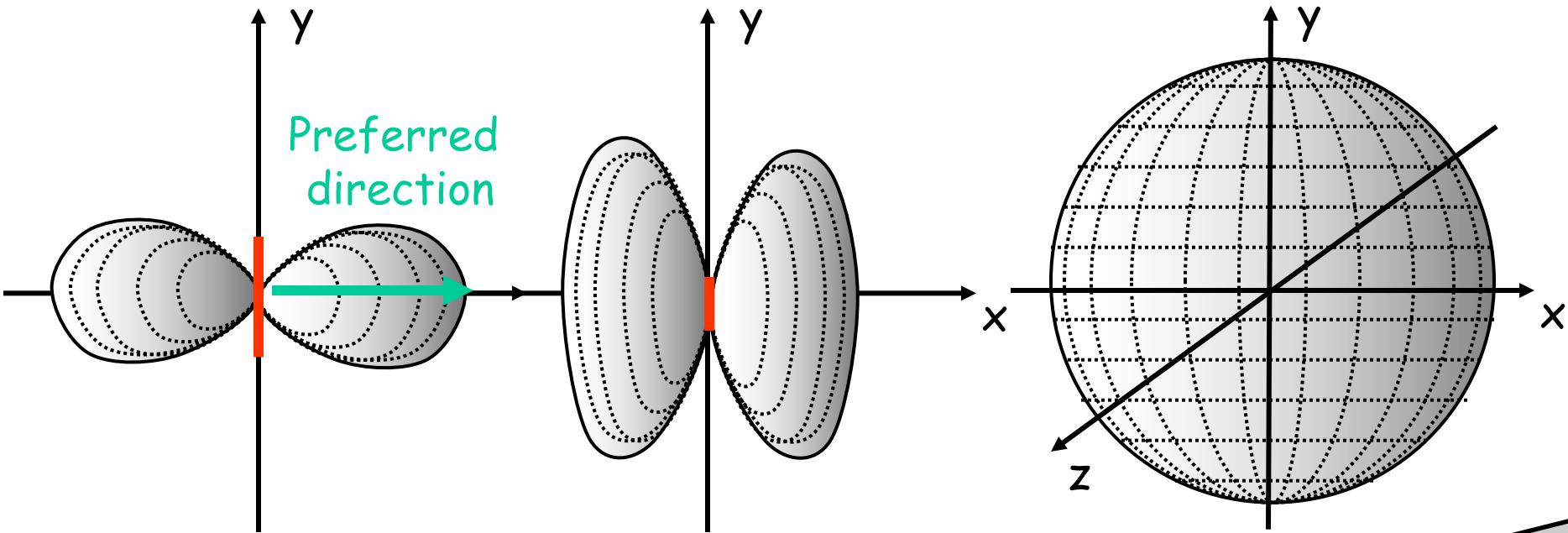
Power measurement

- **dB_i: dB-isotropic, the normalized measure of antenna passive gain**
 - Assumption: an isotropic radiator has 100% efficiency in radiating energy in uniform way in every direction (e.g. the Sun)
 - Antennas concentrate energy in non-isotropic way, resulting in **passive gain (space dependent)**. Ideal antenna: zero length dipole



Power measurement

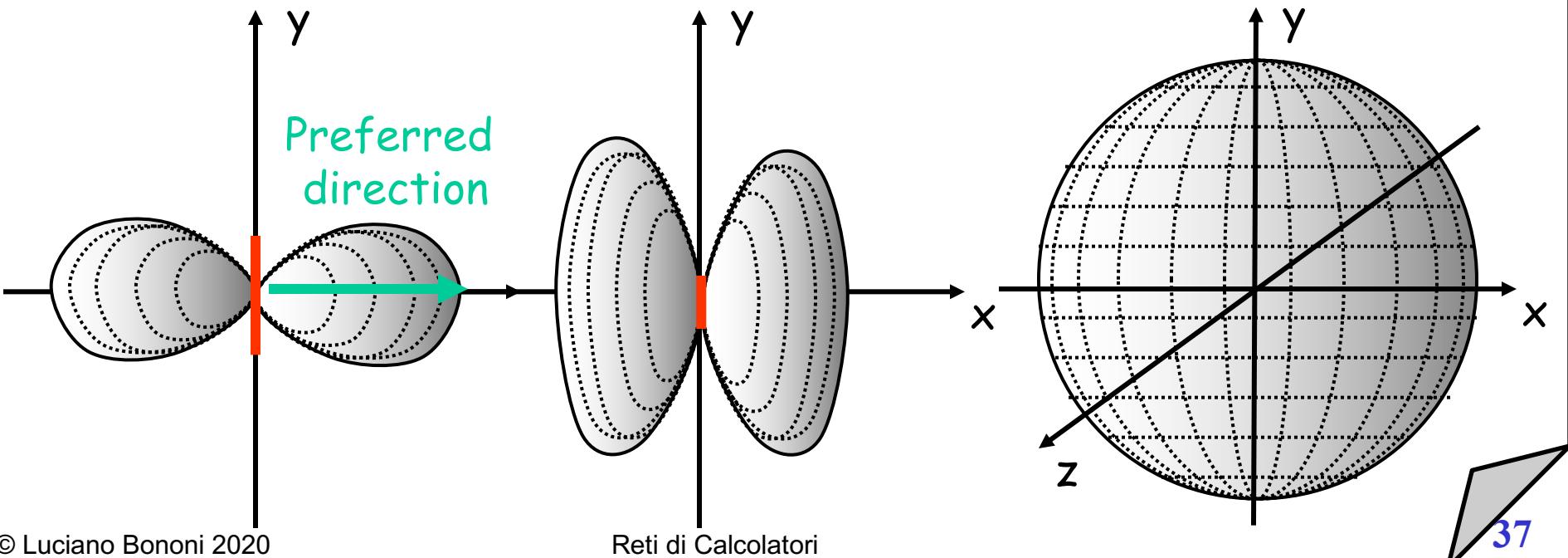
- **dB_i: dB-isotropic, the normalized measure of antenna passive gain**
 - If an antenna located in the origin (0,0,0) has twice the radiated energy of an isotropic radiator in a given point (x,y,z), then the antenna gain in (x,y,z) can be defined as +3 dB_i. If the energy is 10x the isotropic radiator, the gain is +10 dB_i, etc.etc.
 - Q: If the antenna gain is 7 dB_i in (x,y,z)?



Power measurement

- **dB_i: dB-isotropic, the normalized measure of antenna passive gain**
 - Real antennas always have a preferred direction where the power is greater than isotropic radiator: **gain is always positive in the preferred direction!**
 - Example: 1 mW IR power applied to directional antenna with +10 dB_i gain in the preferred direction, would translate in EIRP?
 - **EIRP = 1mW + 10 dB_i = (10x) = 10 mW EIRP**

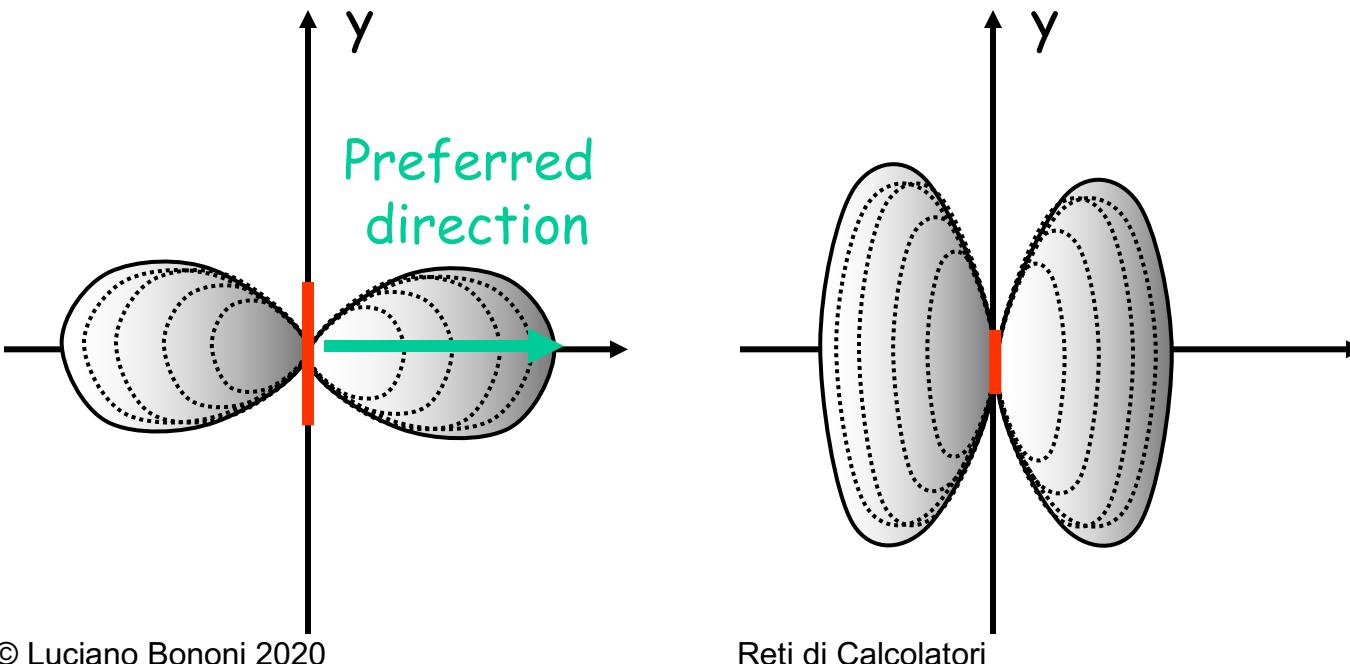
N.B. this does not mean that antenna generates more power !
Antenna concentrates power in preferred direction.



Power measurement

- **dBd: dB-dipole, the normalized antenna passive gain vs. 2,14 dBi half-wave dipole**
 - Reference is a half wave dipole with 2.14 dBi gain in preferred direction!
 - Conversion rule:
 - $0 \text{ dBd} = 2.14 \text{ dBi}$, $\text{dBd} = (\text{dBi} - 2.14)$, $\text{dBi} = (\text{dBd} + 2.14)$

Reference dipole



Power monitoring (e.g. IEEE 802.11 devices)

- (received) Power monitoring in IEEE 802.11 devices is needed for making device driver to work properly (typical sensitivity range is [-90..+10] dBm):
 - Detect signal (below or above the sensitivity threshold?)
 - Detect signal power (selection of coding technique... That is bitrate!)
 - Detect channel status: idle? Ok, transmit! Busy? Ok, wait.
- Received Signal Strength Indicator (RSSI)
 - Index defined for IEEE 802.11 devices (check device analyzer, if any)
 - RSSI = function (dBm or mW received) = pure number reported to device driver!
 - Unfortunately the RSSI scale is not standard, that is, device dependent!
 - This fact does not allow to compare if device A receives better than device B (assuming different manufacturer) based on RSSI measurement
 - Problem: device A indicates maximum RSSI=255 (8 bits) with -10 dBm signal (0.1 mW), and device B indicates maximum RSSI=32 (5 bits) with -15 dBm (0.03 mW). Q: when both A and B in (x,y,z) receive -15 dBm, which one is better device? That is, which one would you buy if you are a system admin?

Antennas

- **Illustration of general issues**
 - Convert electrical energy in RF waves (transmission), and RF waves in eletrical energy (reception)
 - Size of antenna is related to RF frequency of transmission and reception
 - Shape (structure) of the antenna is related to RF radiation pattern
- **Radiation patterns of different antenna types**
- **Positioning antennas**
 - Maximum coverage of workspace
 - Security issues
- **Real antenna types: omni-directional, semi-directional, highly-directional**

Omnidirectional antenna

- Omni-directional antenna: radiates RF power equally in all directions around the vertical axis.
- Most common example: dipole antenna (see Access Points)
 - See how to make it (disclaimer: do not try this at home):
<http://www.nodomainname.co.uk/Omnicolinear/2-4collinear.htm>
<http://www.tux.org/~bball/antenna/>
 - Info & fun: <http://www.wlan.org.uk/antenna-page.html>
 - More info: <http://www.hdtvprimer.com/ANTENNAS/types.html>

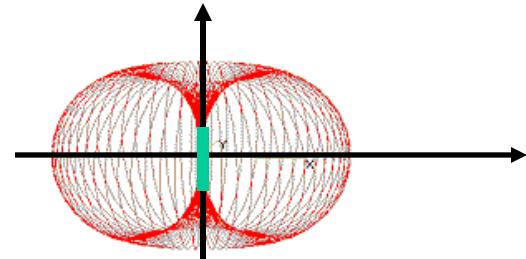


TV dipole

Q: Why TV dipole is bigger?
A: 100 Mhz vs. 2.4 Ghz

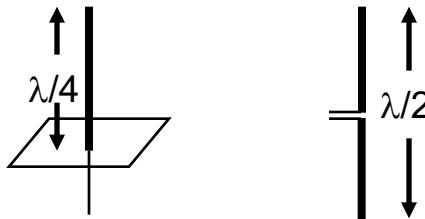


AP dipole

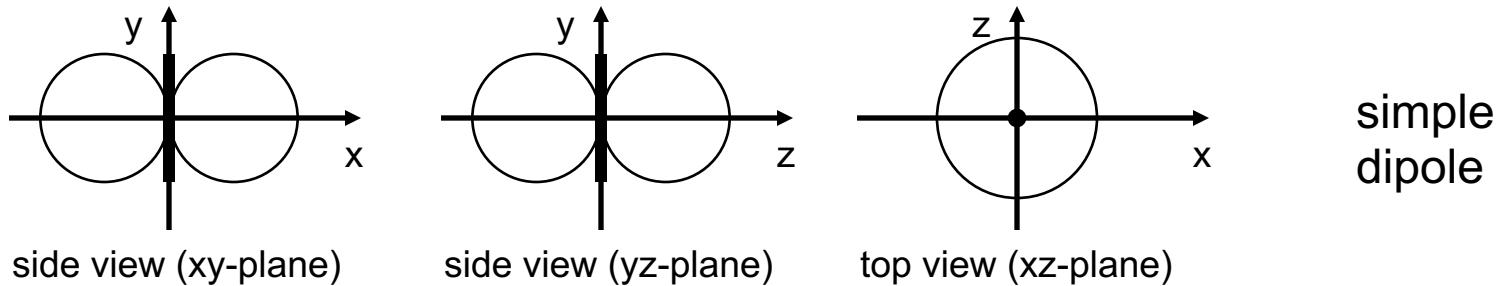


Omnidirectional antennas: simple dipoles

- Real antennas are not isotropic radiators but, e.g., dipoles
→ shape of antenna proportional to wavelength



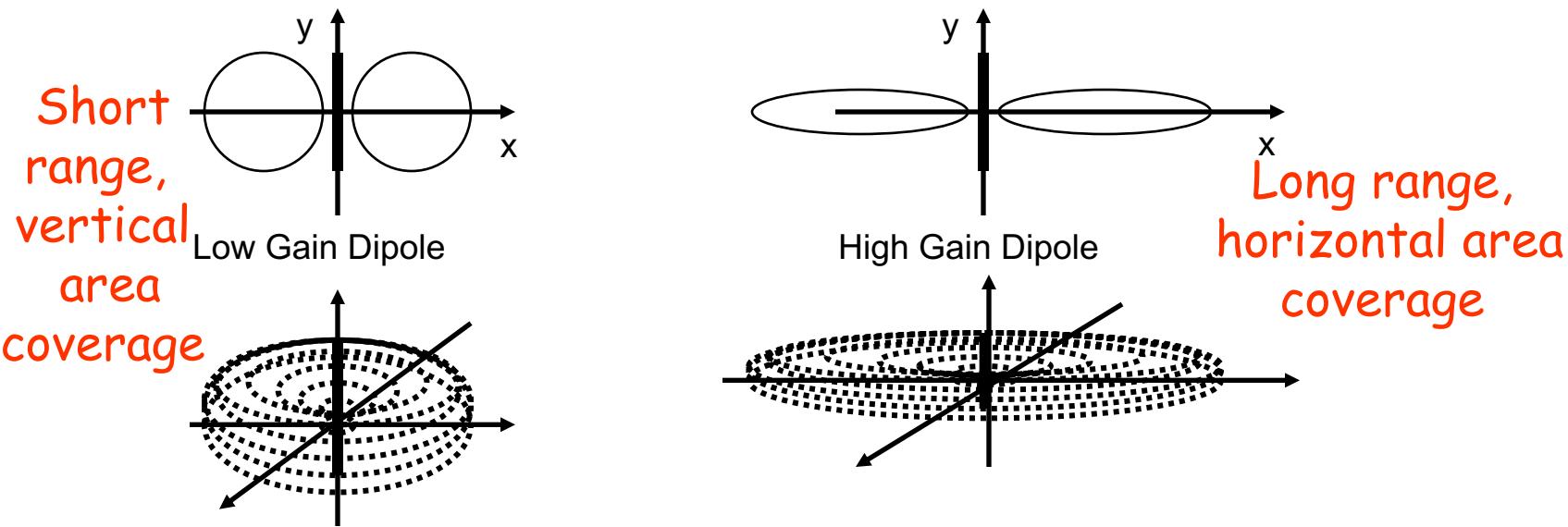
- Example: Radiation pattern of a simple Hertzian dipole



- Gain: maximum power in the direction of the main lobe compared to the power of an isotropic radiator (with the same average power)

Omnidirectional antennas: simple dipoles

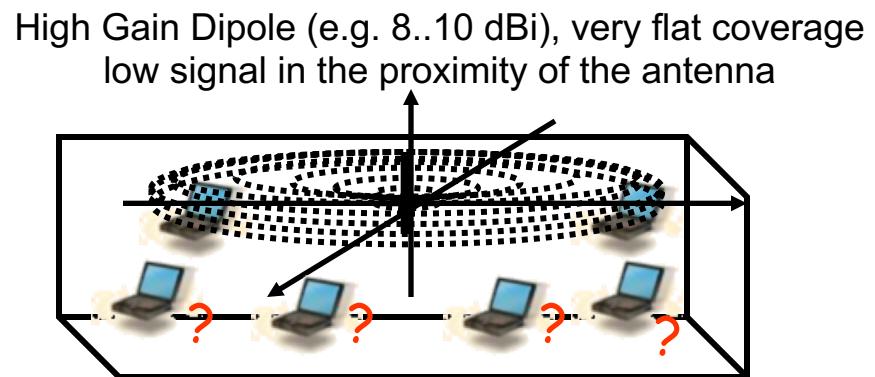
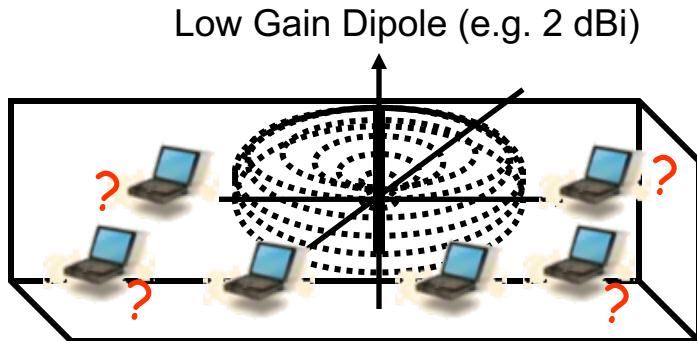
- Dipole: passive gain is due to concentration (shape) of radiation



- Dipole: active gain is obtained with power amplifiers (needs external source of energy)
- N.B. near (below) the dipole the signal is weak! And better radiation is obtained in sub-areas around the dipole!

Omnidirectional antennas: simple dipoles

- Problem: how and when to mount omnidirectional antennas?
And which gain is ok?



- How: Ceiling? Wall? Client positions? Area? Many factors influence the planning...

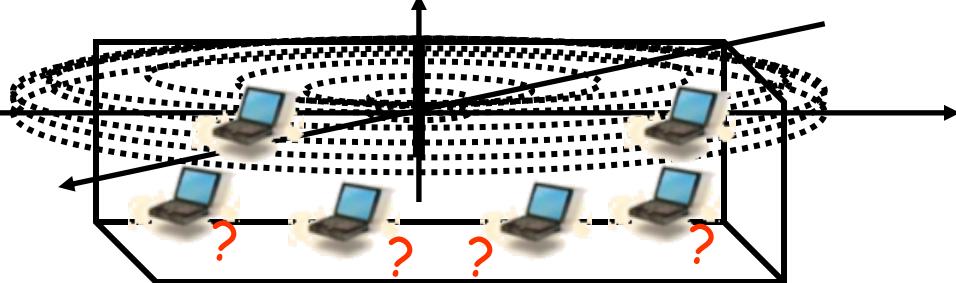
- When:

- need for uniform radio coverage around a central point
- Outdoor: point-to-multipoint connection (star topology)

Omnidirectional antennas: simple dipoles

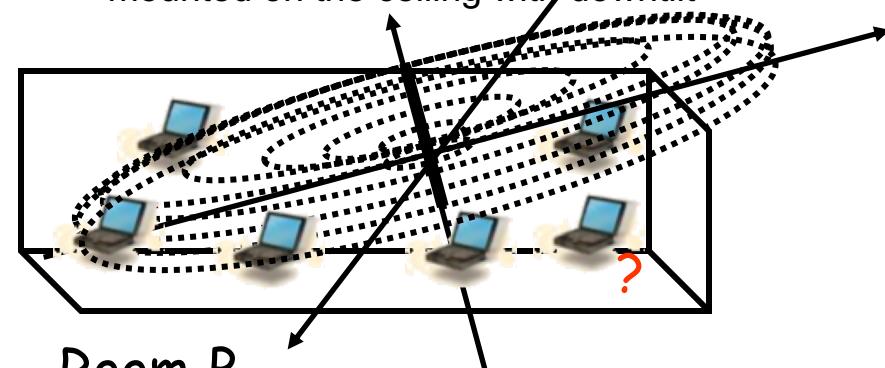
- **Antenna Tilt:** degree of inclination of antenna with respect to perpendicular axis

High Gain Dipole (e.g. 8..10 dBi), very flat coverage mounted on the ceiling



Room A

High Gain Dipole (e.g. 8..10 dBi), very flat coverage mounted on the ceiling with downtilt

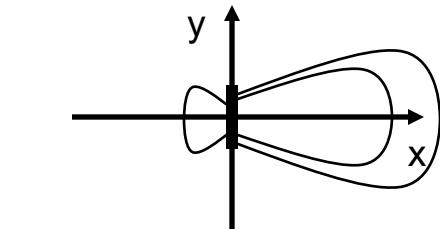


Room B

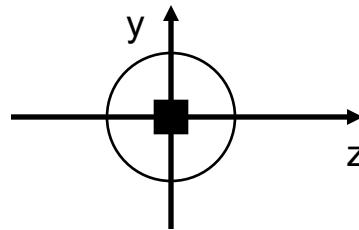
- Some antennas allow a variable degrees **downtilt**.
- Half signal dispersed “in the sky”, 2nd half better exploited.

Semi-directional antennas

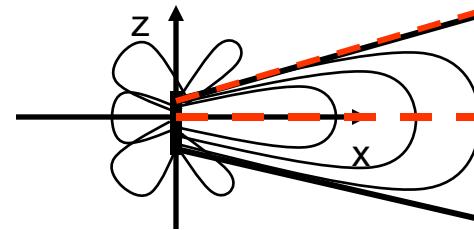
- **Patch (flat antennas mounted on walls)**
- **Panel (flat antennas mounted on walls)**
- **Yagi (rod with tines sticking out)**



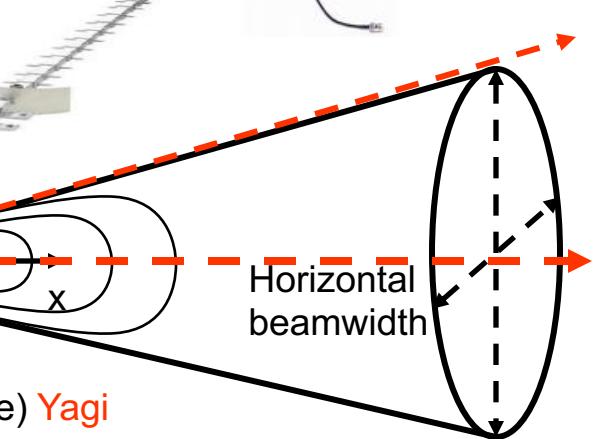
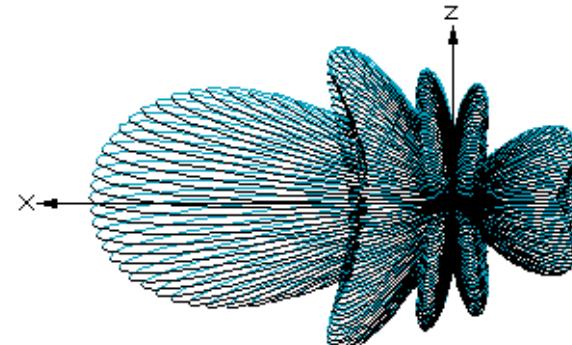
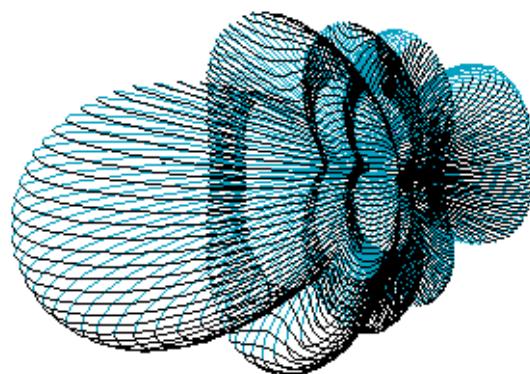
side view (xy-plane) **Patch**



side view (yz-plane)



top view (xz-plane) **Yagi**



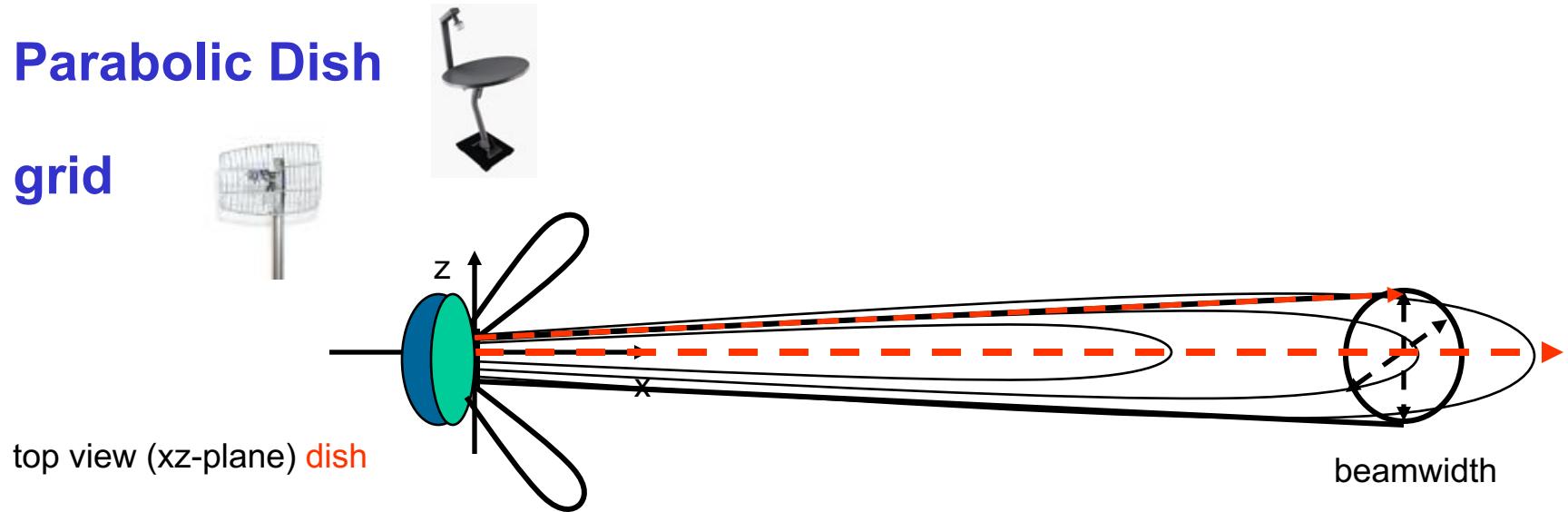
Semi-directional
antenna

**Beamwidth
cone:
-3dB signal
boundary
off-axis**

Credits: <http://www.hdtvprimer.com/ANTENNAS/types.html>

highly-directional antennas

- Parabolic Dish
- grid



Antenna type	H beamwidth	V beamwidth
Omni-dir.	360°	7°.. 80°
Patch/panel	30° .. 180°	6° .. 90°
Yagi	30° .. 78°	14° .. 64°
Parabolic dish	4° .. 25°	4° .. 21°

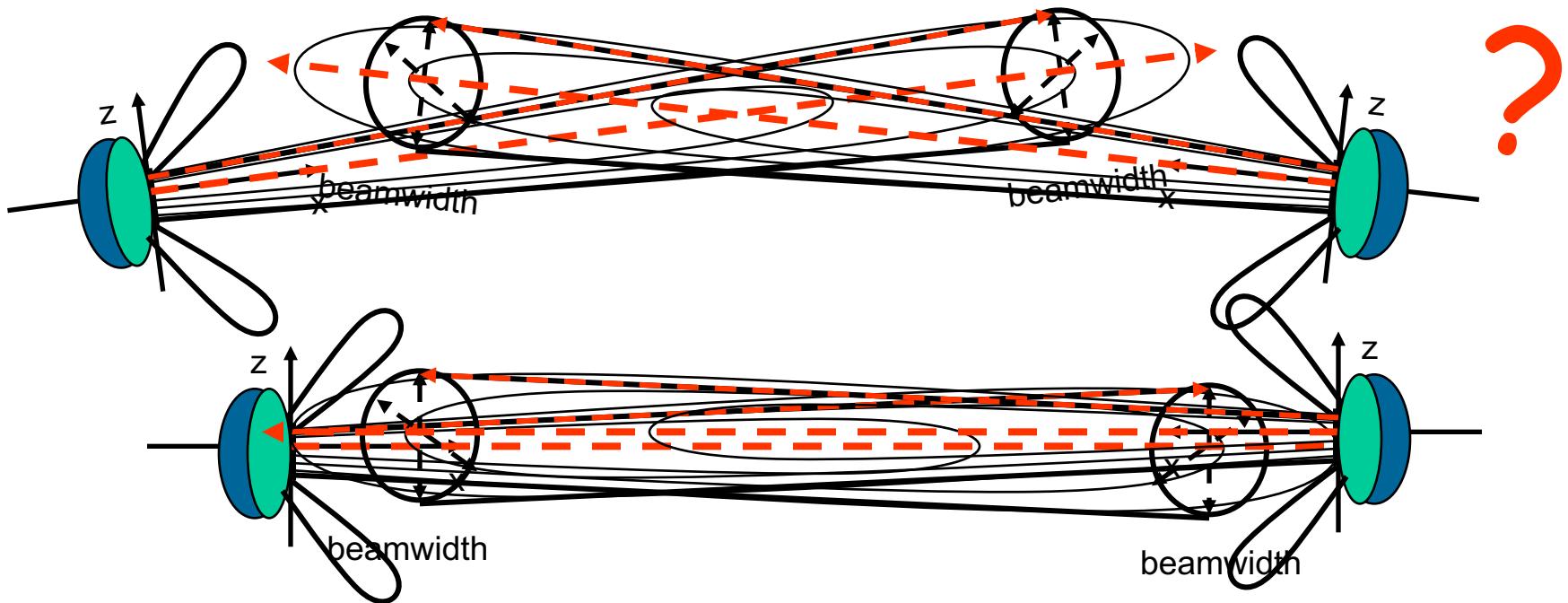
Semi-directional antenna

Beamwidth cone:
-3dB signal boundary off-axis

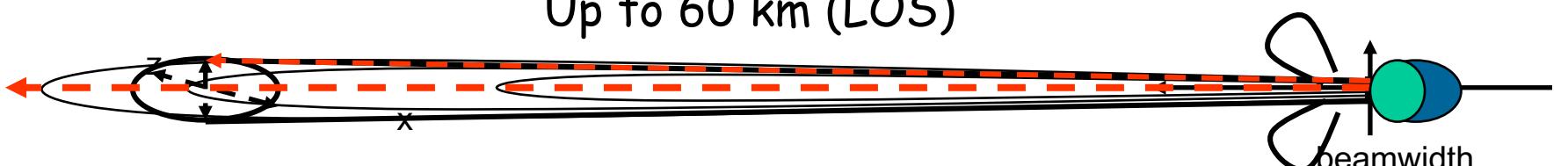
highly-directional antennas

- Common use: Point-to-point link

Out of beam alignment



Up to 60 km (LOS)

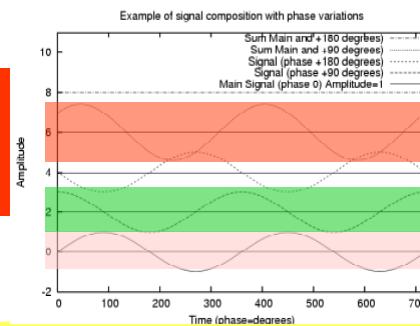


Wind effect: better to have lower gain and wider beam

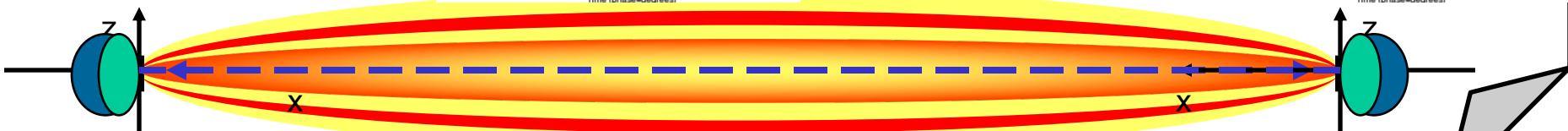
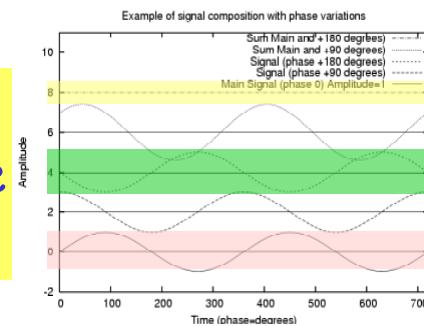
highly-directional antennas

- **Line of sight (LOS):**
 - Straight line between transmitter and receiver
 - No obstructions (outdoor long range reduces reflections)
 - Polarization is more important than in indoor scenarios
- **Fresnel Zone: RF is not laser light, RF signals diffuse energy in space**
 - Ellipse shaped area centered on the LOS axis
 - Most additive RF signal is concentrated in the Fresnel Zone
 - It is important that Fresnel Zone is free from obstacles

Red zone:
additive phase signal



Yellow zone:
inverse phase
signal



highly-directional antennas

▪ Fresnel Zone (FZ)

- Blockage of Fresnel Zone causes link disruption
 - Caused by buildings, (growing) trees, foliage, etc.
 - Rule of thumb: < 20% obstruction of Fresnel Zone
 - Practical rule: calculate the radius of FZ leaving 60% unobstructed radius
 - $R_{60\%} = 43.3 \times \sqrt{d/4f}$
 - $R_{100\%} = 72.2 \times \sqrt{d/4f}$

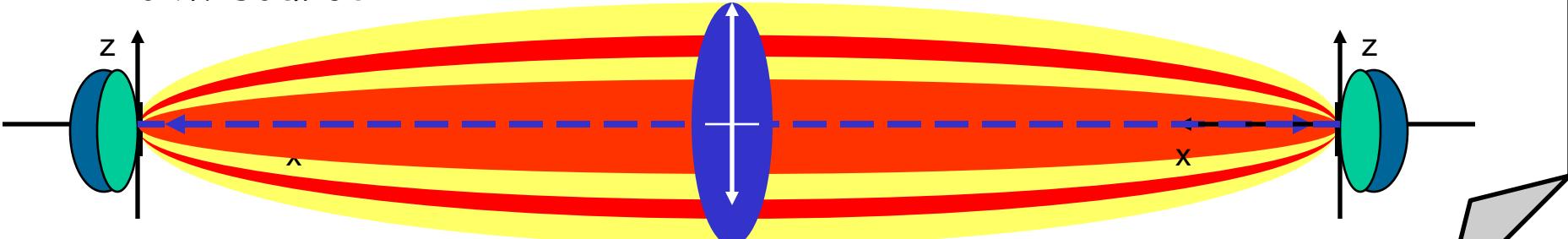
1st FZ



Point source

R=radius of 60% central FZ (feet)
d=distance(Miles), f=freq (GHz)

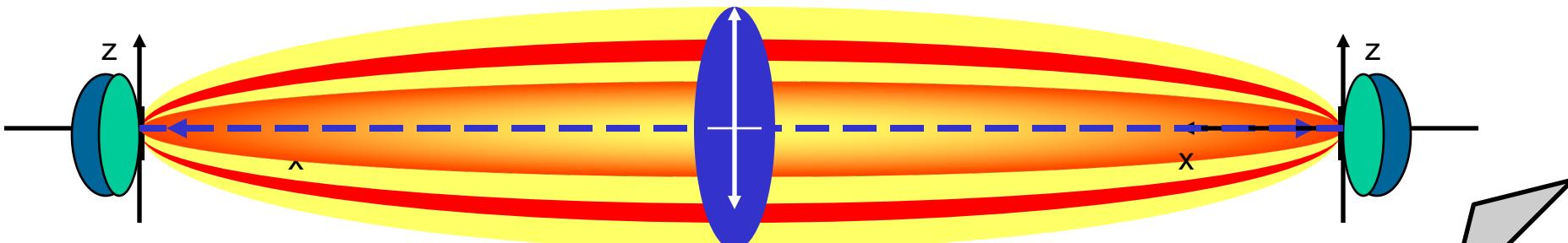
R=radius of 100% FZ (feet)
d=distance(Miles), f=freq (GHz)



highly-directional antennas

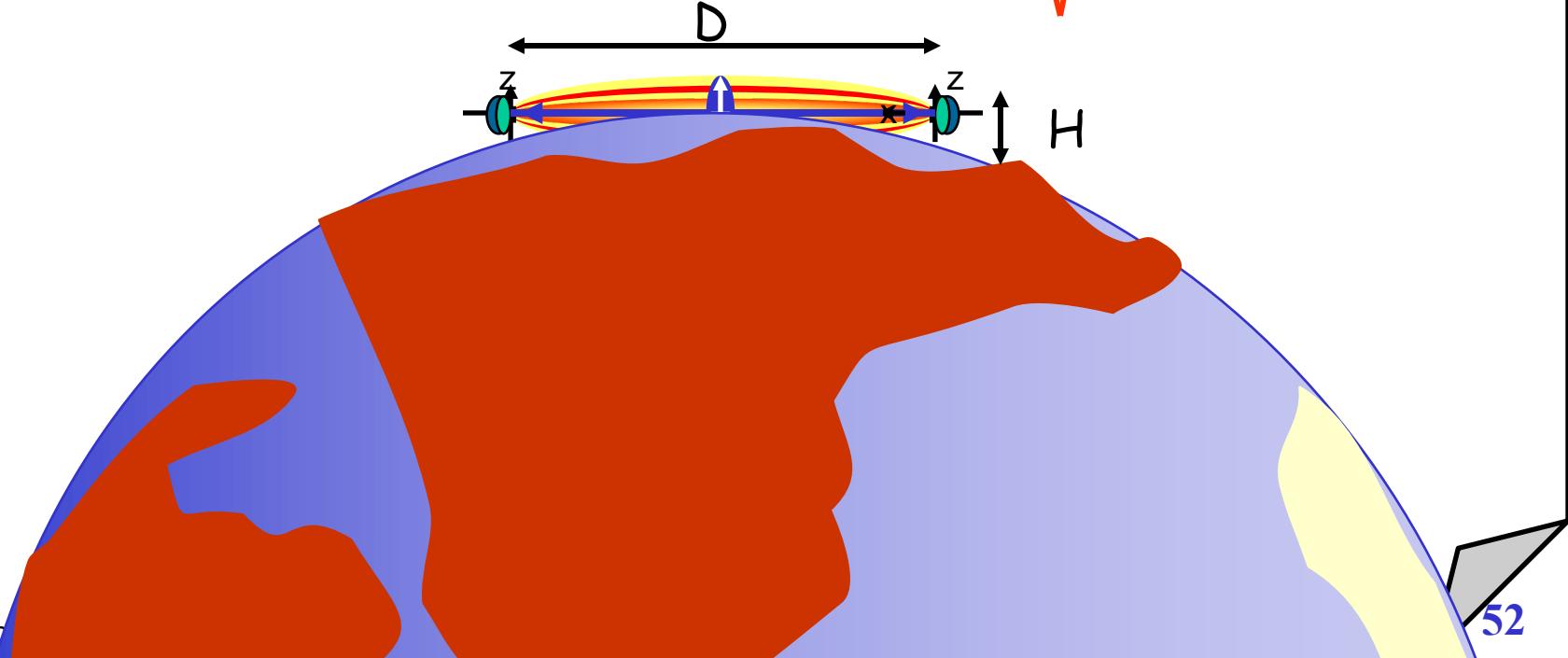
▪ Fresnel Zone (FZ)

- N.B. the FZ radius depends only on the distance d between antennas, and frequency f of RF signal!
- Type of antenna, beam width (degree), and gain (dBi) have no effects!
 - E.g. +13 dBi Yagi (30 degree beam) vs. +24 dBi Dish (5 degrees) have the same FZ!!!!
- In practice: if FZ is partially obstructed, it is not useful to use higher gain antennas (with small degree beam) !!!



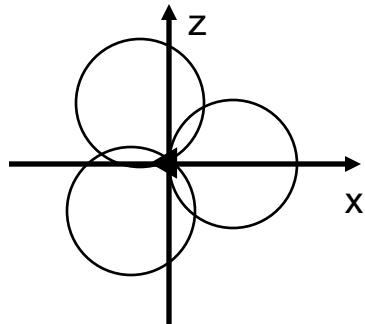
highly-directional antennas

- **Fresnel Zone (FZ)**
 - Is not relevant in indoor scenarios (due to reflections...)
- **Consider the Earth bulge!!!**
 - Very long point-to-point connections may have more than 40% FZ obstructed by Earth surface! **Earth Bulge height = h (feet) = $D^2/8$**
 - Minimum antenna height (link > 7 miles) $H = (43.3 \sqrt{D/4F}) + D^2/8$

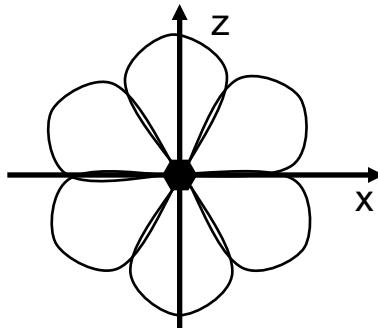


Sectorized-directional antennas

- **Arrays of sectorized directional antennas**



top view, 3 sector



top view, 6 sector



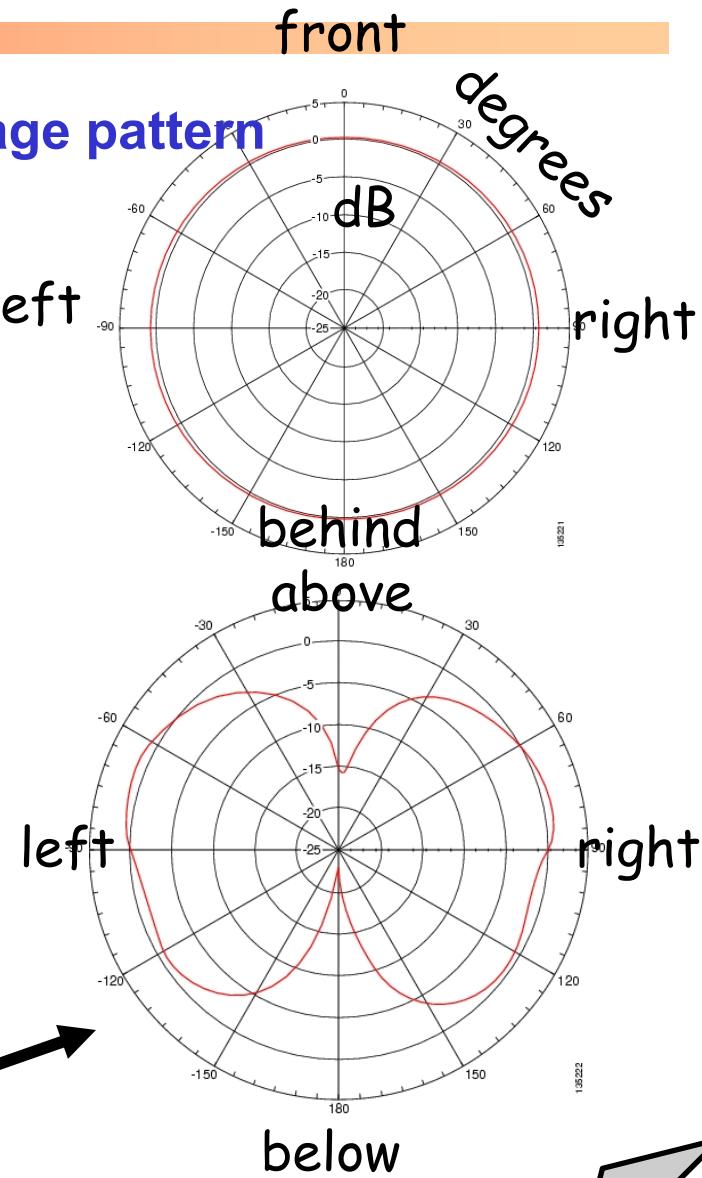
sectorized
antenna

- **Space multiplexing (channel reuse)**

Azimuth and Elevation antenna charts

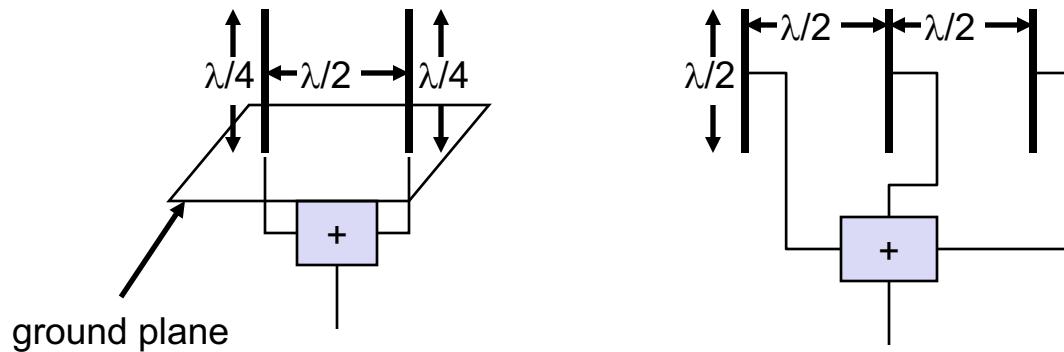
Charts for understanding antenna coverage pattern

- **Azimuth chart** (pattern seen from front/right/behind/left)
 - Obtained with spectrum analyzer with central antenna frequency
 - Signal measured in dB around the antenna
 - E.g. Dipole pattern: almost circular
 - E.g. yagi pattern: high in front, low beside
 - N.B. distance and Tx power is not relevant (signal strength in a location is relative to every other location in the chart, like with dB)
 - **Elevation chart** (pattern seen front/below/behind/above)



Antennas: diversity

- **Grouping of 2 or more antennas**
 - multi-element antenna arrays
- **Antenna diversity**
 - switched diversity, selection diversity
 - receiver chooses antenna with largest output
 - diversity combining
 - combine output power to produce gain
 - cophasing needed to avoid cancellation (phased antenna array... Requires processor)



Path Loss

- Path Loss: RF signal “dispersion” (attenuation) as a function of distance
 - E.g. Possible formulas (36.6 or 32.4)
 - Free space: Loss (in dB) = $36.6 + (20 \cdot \log_{10}(F)) + (20 \cdot \log_{10}(D))$
 - F (Mhz), D (miles)
- Link budget issue: 6 dB rule
 - Each 6 dB increase in EIRP (signal x 4) implies double Tx range (e.g. see table below: 2.4Ghz Path Loss vs distance)

100 meters	- 80.23 dB
200 meters	- 86.25 dB
500 meters	- 94.21 dB
1000 meters	- 100.23 dB
2000 meters	- 106.25 dB
5000 meters	- 114.21 dB
10000 meters	- 120.23 dB

The diagram illustrates the relationship between distance and path loss. It shows a table of path loss values for various distances. To the right of the table, a series of arrows points downwards, indicating a decrease of 6 dB for each subsequent distance entry. The first arrow points from the 100m row to the 200m row. Subsequent arrows point from the 200m row to the 500m row, from the 500m row to the 1000m row, from the 1000m row to the 2000m row, from the 2000m row to the 5000m row, and from the 5000m row to the 10000m row.

Link Budget Calculation

- “Link Budget” or “System Operating Margin”
 - Excess of signal between transmitter and receiver
 - Calculated for outdoor point-to-point connections
 - Measured in dB (relative) or dBm or mW (absolute)
 - Calculation:
 - Receiver sensitivity RS (weakest detectable signal)
 - The lower the better: e.g. IEEE 802.11 card (see device manual), -95 dBm (1Mbps), -93 dBm (2 Mbps), -90 dBm (5.5 Mbps), -87 dBm (11 Mbps)
 - Link Budget: received power (in dBm) - RS (in dBm)
 - E.g. RS = -82 dBm, received power = -50 dBm
 - $\text{Link budget} = -50 - (-82) = +32 \text{ dBm}$
 - This means the signal has margin of +32 dB before it becomes unviable
- **Fade margin: extra margin for link budget (to cope with multipath variation in indoor/outdoor scenarios): typical [+10..+20] dB**

Link Budget Calculation: example

- Example: design of transmission system, needs amplifier?

