Topic

- Sharing bandwidth between flows
  - WFQ (Weighted Fair Queuing)
  - Key building block for QOS
Sharing with FIFO Queuing

- **FIFO “drop tail” queue:**
  - Queue packets First In First Out (FIFO)
  - Discard new packets when full
  - Typical router queuing model

- Sharing with FIFO queue
  - Multiple users or flows send packets over the same (output) link
  - What will happen?
Sharing with FIFO Queuing (2)

- Bandwidth allocation depends on behavior of all flows
  - TCP gives long-term sharing – with delay/loss, and RTT bias
  - Aggressive user/flow can crowd out the others

Inputs

Flow 1

Flow 2

Flow 3

FIFO queue

Output order same as arrival order

Output
Round-Robin Queuing

• Idea to improve fairness:
  – Queue packets separately for each flow; take one packet in turn from each non-empty flow at the next output time
Round-Robin Queuing (2)

• Idea to improve fairness:
  – Queue packets separately for each flow; take one packet in turn from each non-empty flow at the next output time
  – How well does this work?

Inputs

Flow 1
Flow 2
Flow 3

Per flow queues

Output order is no longer arrival order

Round robin output
Round-Robin Queuing (3)

- Flows don’t see uncontrolled delay/loss from others!
- But different packet sizes lead to bandwidth imbalance
  - Might be significant, e.g., 40 bytes vs 1500 bytes

Inputs

- Flow 1
- Flow 2
- Flow 3

Per flow queues

Round robin output

Output order is no longer arrival order
Fair Queuing

• Round-robin but approximate bit-level fairness:
  – Approximate by computing virtual finish time
  – Virtual clock ticks once for each bit sent from all flows
  – Send packets in order of their virtual finish times, Finish(j)\textsubscript{F}
  – Not perfect – don’t preempt packet being transmitted

\begin{align*}
\text{Arrive}(j)\textsubscript{F} & = \text{arrival time of j-th packet of flow } F \\
\text{Length}(j)\textsubscript{F} & = \text{length of j-th packet of flow } F \\
\text{Finish}(j)\textsubscript{F} & = \max (\text{Arrive}(j)\textsubscript{F}, \text{Finish}(j-1)\textsubscript{F}) + \text{Length}(j)\textsubscript{F}
\end{align*}
Fair Queuing (2)

• Suppose:
  – Flow 1 and 3 use 1000B byte packets, flow 2 uses 300B packets
  – What will fair queuing do?
Fair Queuing (3)

• Suppose:
  – Flow 1 and 3 use 1000B packets, flow 2 uses 300B packets
  – What will fair queuing do?

Let Finish(0)_F=0, queues backlogged [Arrive(j)_F < Finish(j-1)_F]
Finish(1)_F1=1000, Finish(2)_F1=2000, ...
Finish(1)_F2=300, Finish(2)_F2=600, Finish(3)_F2=900, 1200, 1500, ...
Finish(1)_F3=1000, Finish(2)_F3=2000, ...
Fair Queuing (4)

• Suppose:
  – Flow 1 and 3 use 1000B byte packets, flow 2 uses 300B packets
  – What will fair queuing do?

Inputs

Flow 1

Flow 2

Flow 3

Per flow queues

Fair queuing

Output order is no longer round-robin
WFQ (Weighted Fair Queuing)

- WFQ is a useful generalization of Fair Queuing:
  - Assign a weight, \( \text{Weight}_F \), to each flow
  - Higher weight gives more bandwidth, e.g., 2 is 2X bandwidth
  - Change computation of \( \text{Finish}(j)_F \) to factor in \( \text{Weight}_F \)

\[
\begin{align*}
\text{Arrive}(j)_F &= \text{arrival time of j-th packet of flow F} \\
\text{Length}(j)_F &= \text{length of j-th packet of flow F} \\
\text{Finish}(j)_F &= \max (\text{Arrive}(j)_F, \text{Finish}(j-1)_F) + \text{Length}(j)_F / \text{Weight}_F
\end{align*}
\]
Using WFQ

• Lots of potential!
  – Can prioritize and protect flows
  – A powerful building block

• Not yet a complete solution
  – Need to determine flows (user? application? TCP connection?)
  – Difficult to implement at high speed for many concurrent flows
  – Need to assign weights to flows
Where we are in the Course

• Revisiting the layers
  – Network security affects all layers because each layer may pose a risk

<table>
<thead>
<tr>
<th>Application</th>
<th>Transport</th>
<th>Network</th>
<th>Link</th>
<th>Physical</th>
</tr>
</thead>
</table>
Topic

• Network security designs to protect against a variety of threats
  – Often build on cryptography
  – Just a brief overview. Take a course!
Security Threats

• “Security” is like “performance”
  – Means many things to many people
  – Must define the properties we want

• Key part of network security is clearly stating the threat model
  – The dangers and attacker’s abilities
  – Can’t assess risk otherwise
Security Threats (2)

• Some example threats
  – It’s not all about encrypting messages

<table>
<thead>
<tr>
<th>Attacker</th>
<th>Ability</th>
<th>Threat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eavesdropper</td>
<td>Intercept messages</td>
<td>Read contents of message</td>
</tr>
<tr>
<td>Intruder</td>
<td>Compromised host</td>
<td>Tamper with contents of message</td>
</tr>
<tr>
<td>Impersonator</td>
<td>Remote social engineering</td>
<td>Trick party into giving information</td>
</tr>
<tr>
<td>Extortionist</td>
<td>Remote / botnet</td>
<td>Disrupt network services</td>
</tr>
</tbody>
</table>
Uses of Cryptography

• Encrypting information is useful for more than deterring eavesdroppers
  – Prove message came from real sender
  – Prove remote party is who they say
  – Prove message hasn’t been altered

• Designing a secure cryptographic scheme is full of pitfalls!
  – Use approved design in approved way
Topic

- Encrypting information to provide confidentiality
  - Symmetric and public key encryption
  - Treat crypto functions as black boxes
Goal and Threat Model

- Goal is to send a private message from Alice to Bob
  - This is called confidentiality
- Threat is Eve will read the message
  - Eve is a passive adversary (observes)
Encryption/Decryption Model

- Alice encrypts private message (plaintext) using key
- Eve sees ciphertext but can’t relate it to private message
- Bob decrypts using key to obtain the private message
Encryption/Decryption (2)

- Encryption is a reversible mapping
  - Ciphertext is confused plaintext
- Assume attacker knows algorithm
  - Security does not rely on its secrecy
- Algorithm is parameterized by keys
  - Security does rely on key secrecy
  - Must be distributed (Achilles’ heel)
Encryption/Decryption (3)

Two main kinds of encryption:

1. Symmetric key encryption », e.g., AES
   - Alice and Bob share secret key
   - Encryption is a bit mangling box

2. Public key encryption », e.g., RSA
   - Alice and Bob each have a key in two parts: a public part (widely known), and a private part (only owner knows)
   - Encryption is based on mathematics (e.g., RSA is based on difficulty of factoring)
Symmetric (Secret Key) Encryption

- Alice and Bob have the same secret key, $K_{AB}$
  - Anyone with the secret key can encrypt/decrypt
Public Key (Asymmetric) Encryption

- Alice and Bob each have public/private key pair \((K_B / K_B^{-1})\)
  - Public keys are well-known, private keys are secret to owner
Public Key Encryption (2)

- Alice encrypts with Bob's public key $K_B$; anyone can send
- Bob decrypts with his private key $K_B^{-1}$; only he can do so
Key Distribution

• This is a big problem on a network!
  – Often want to talk to new parties

• Symmetric encryption problematic
  – Have to first set up shared secret

• Public key idea has own difficulties
  – Need trusted directory service
  – We’ll look at certificates later
Symmetric vs. Public Key

- Have complementary properties
  - Want the best of both!

<table>
<thead>
<tr>
<th>Property</th>
<th>Symmetric</th>
<th>Public Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key Distribution</td>
<td>Hard – share secret per pair of users</td>
<td>Easier – publish public key per user</td>
</tr>
<tr>
<td>Runtime Performance</td>
<td>Fast – good for high data rate</td>
<td>Slow – few, small, messages</td>
</tr>
</tbody>
</table>
Winning Combination

• Alice uses public key encryption to send Bob a small private message
  – It’s a key! (Say 256 bits.)
• Alice and Bob send large messages with symmetric encryption
  – Using the key they now share
• The key is called a session key
  – Generated for short-term use