Network Security

- Network, such as Internet, is open to everybody
    - Possibility of misbehavior or misuse of network resources

    $\rightarrow$ Compromise network utility

- Network security is about
    - "Appropriate" use of network resources
    - That is, high utility of resources in a proper manner

- Network security is *not* restricted to
    - Secure private communications as in classical cryptograph

Network Security

- Security of network can be threatened in many possible ways
- *Two* prominent ways in which network security is compromised:

  (i) Protocol level security:

  - Prevention against exploitation of "weakness" of current network protocol, e.g.
  - Routing: false route announcements, or greedy routing
  - TCP: users not behaving according to TCP protocol by sending too much traffic or sending false ACK to receive more data

  (ii) Security against malicious users:

  - Prevention of *unwanted* traffic that is sent to disrupt network utility, e.g.
  - worms
  - denial of service attack, flooding, etc.

Network Security

- Security concerns demand
  - Design of secure network architecture based on distributed protocols
    - when possible
  - Identification of network *vulnerability*, and
  - Policing mechanism
    - when not possible to have secure architecture

- We will address the above issues
  - In the context of
    - Routing, and
    - Congestion control

Secure Routing

- Current routing architecture is vulnerable to attacks
- Primary vulnerabilities are:
  - False path announcement
    - that is, intermediate nodes provide wrong information

    $\longrightarrow$ can lead to serious consequence (credit card information !)

    $\longrightarrow$ we need path verification mechanism
  - Greedy routing rather than cooperative
    - that is, individual ISPs do not route data in socially optimal manner

    $\longrightarrow$ how bad is such behavior?

    $\longrightarrow$ if very bad, how to prevent it?
- First, we'll talk about security against "false path announcement"

Secure Routing I

- False path announcement'
  - Consider a malicious node pretending to have a "short" path from itself to some popular destination "`cnn.com`"
  - Then, all of its neighbors will route data for "`cnn.com`" through malicious node

  $\rightarrow$ any node in the network can potentially become "`cnn.com`"

- A clever solution
  - Well, if a node announces existence of path,
    - it must prove its existence
  - Question:
    - how to design verification scheme for the proofs produced by potentially malicious node ?

Secure Routing I

- We'll present a simple scheme that uses existence of public-key and private-key
  - Let `Pub` and `Priv` be public and private key of a node, then
    - it can sign any data using `Priv` key (no one else can)
    - everyone else can unsign the signed data using `Pub` key

- Here is verifiable way to produce "proof of path-existence"
  - Let M claim to have path to `cnn.com` to node A (neighbor of M)
  - Suppose M is the only bad node
  - Suppose each node has unique identity and signature which can be signed by that node only
  - Let M claim to have path

$$M \to x_1 \to \cdots \to x_k \to \texttt{cnn.com}$$

Secure Routing I

- Then, M asks $x_1, \ldots, x_k$ and `cnn.com` to sign as follows:

  (0) $\mathsf{SIGN}_A(\mathsf{PROVE}) \rightarrow \mathsf{MSG}$: give to M

  (1) $\mathsf{SIGN}_M(\mathsf{MSG}) \rightarrow \mathsf{MSG}_0$

  (2) Repeated obtain signatures as follows:

$$
\begin{array}{lcl}
\mathsf{MSG}_1 & \rightarrow & \mathsf{SIGN}_{x_1}(\mathsf{MSG}_0) \\
\vdots & & \\
\mathsf{MSG}_k & \rightarrow & \mathsf{SIGN}_{x_k}(\mathsf{MSG}_{k-1}) \\
\mathsf{MSG}_{\texttt{cnn.com}} & \rightarrow & \mathsf{SIGN}_{\texttt{cnn.com}}(\mathsf{MSG}_k)
\end{array}
$$

  - A unsigns $\mathsf{MSG}_{\texttt{cnn.com}}$ one-by-one using public signature of `cnn.com`, $x_k, \ldots, x_1$, M, and A.
    - If PROVE is what it gets, then M has path
    - If not, then M does not have path

- Existence of cryptographic Public-Private key mechanism helps in making algorithm secure

Secure Routing II

- Next, we consider the question of *greedy* routing
  - ISPs route data so as to maximize their own utility
  - Without worrying for social utility maximization

- First, we evaluate the possible "degradation"
  - Popularly known as *Price of anarchy*
  - We will find that it's not "too much"

$\rightarrow$ No need of designing prevention mechanism

Secure Routing II

- Recall, ROUTE-OPT (social optimal)

$$\min \quad \sum_{e \in \mathcal{L}} D_e(F_e) F_e$$

subject to

$$\sum_{p \in \mathcal{P}_i} f_p = r_i \ ; \quad i \in \{1, \ldots, k\}$$

$$f_p \geq 0 \ ; \quad f_e = \sum_{p : e \in p} f_p \ ; \quad e \in \mathcal{L}$$

- A feasible $f = (f_p)$ w.r.t. $r = (r_i)$ satisfies the above constraints
  - Here, $i \in \{1, \ldots, k\}$ represents a source-destination pair
  - $\mathcal{P}_i$: set of all possible paths between source-destination pair i
  - f_p: value of flow along path P
  - r_i: demand for source-destination pair i

Greedy Routing

- Greedy routing
  - ∘ Always route demand on the minimal delay path
  - ∘ Not the same as fixed shortest path routing
    - – since, delay is load dependent
- In presence of non-cooperative environment, such behavior is expected
  - ∘ "Selfish" or "rational" thing to do
- Question:
  - ∘ How to make sure that performance does not degrade!
  - ∘ Or, is there a need of any such mechanism?
- In routing: we find that performance does not degrade much!

Greedy Routing

- A natural way to evaluate greedy-routing
  - Study performance of equilibrium point of greedy routing
  - Question: what is equilibrium point?

- Notation: given feasible flow $f = (f_p)$ for (G, r)
  - $D_p(f) = \sum_{e \in p} D_e(f_e)$: (delay of flow on p)

- In equilibrium of greedy routing
  - There should not be a flow i with two paths p_1 and p_2 such that
    - $f_{p_1}, f_{p_2} > 0$ and for some $\delta \in [0, f_{p_1}]$

$$D_{p_1}(f_{p_1} - \delta) > D_{p_2}(f_{p_2} + \delta).$$

$\rightarrow$ This leads to definition of Nash equilibrium

Nash Equilibrium

- **Nash Equilibrium.** A feasible flow f for (G, r) is at *Nash equilibrium* if and only if
  - for all $i \in \{1, \ldots, k\}$, $p_1, p_2 \in \mathcal{P}_i$, and $\delta \in [0, f_{p_1}]$

$$D_{p_1}(f) \leq D_{p_2}(\tilde{f}),$$

  - where

$$\tilde{f}_p = \begin{cases} f_p^* - \delta & p = p_1 \\ f_p^* + \delta & p = p_2 \\ f_p^* & p \neq p_1, p_2 \end{cases}$$

- **Wardrop's Principle.** A feasible flow f for (G, r) with delay function D is called a Nash Equilibrium if and only if
  - $\forall i \in \{1, \ldots, k\}$; $p_1, p_2 \in \mathcal{P}_i$ with $f_{p_1} > 0$

$$D_{p_1}(f) \leq D_{p_2}(f).$$

Greedy Routing

- Cost of flow f:

$$C(f) = \sum_e D_e(f_e) f_e = \sum_p D_p(f) f_p$$

- Given (G, r):
  - $G^*(G, r)$: cost of ROUTE-OPT
  - $G_N(G, r)$: maximal cost of Nash Equilibrium
- **Goal.** Evaluate

$$\rho(G, r, D) = \frac{G_N(G, r)}{G^*(G, r)}$$

- Next,
  - Characterization of Nash Equilibrium as solution to another optimization problem
  - Bound on $\rho(G, r, D)$ using above characterization
    - simple bound for special case of delay
    - general bound

Nash Equilibrium

- Let $D_e(\,\cdot\,)$ be continuous, strictly increasing and strictly convex
- Let $f^N = (f_p^N)$ be a Nash Equilibrium
  - Define $h_e(x) = \displaystyle\int_0^x D_e(t)dt$.
    - $h_e(\,\cdot\,)$ is strictly convex, increasing
- Consider a Convex Optimization Problem:

NCP: $$\min \quad \sum_{e \in \mathcal{L}} h_e(f_e)$$

subject to

$$\sum_{p \in \mathcal{P}_i} f_p = r_i \; ; \quad \forall i \in \{1, \ldots, k\}$$

$$f_p \geq 0 \; ; \quad f_e = \sum_{p : e \in p} f_p \; ; \quad \forall e \in \mathcal{L}$$

Nash Equilibrium

- **NCP** is strictly convex with convex constraints
  - There is a unique optimal solution
    - let it be f^*

- By property of convex optimization
  - There is no descent direction at f^*.
    - we will use this property to relate it to Nash Equilibrium

- Define,

$$C_h(f) = \sum_{e \in \mathcal{L}} h_e(f_e).$$

Nash Equilibrium

- Descent direction at f^*
  - There is $i \in \{1, \ldots, k\}$ and $p_1, p_2 \in \mathcal{P}_i$ s.t.
    - $f^*_{p_1} > 0$; and $C_h(\tilde{f}) < C_h(f^*)$ s.t.

$$\tilde{f}_p = \begin{cases} f^*_p - \delta & p = p_1 \\ f^*_p + \delta & p = p_2 \\ f^*_p & p \neq p_1, p_2 \end{cases} \quad ; \forall \delta \in (0, \epsilon) \text{ for some } \epsilon > 0.$$

- $C_h(f^*) - C_h(\tilde{f}) = \sum_{e \in p_1} \Big[h_e(f^*_e) - h_e(f^*_e - \delta) \Big] + \sum_{e \in p_2} \Big[h_e(f^*_e) - h_e(f^*_e + \delta) \Big]$
- Hence: $\sum_{e \in p_1} \left[\frac{h_e(f^*_e) - h_e(f^*_e - \delta)}{\delta} \right] > \sum_{e \in p_2} \left[\frac{h_e(f^*_e + \delta) - h_e(f^*_e)}{\delta} \right]$
- Taking $\delta \to 0$, we obtain

$$\sum_{e \in p_1} h'_e(f^*_e) > \sum_{e \in p_2} h'_e(f^*_e) \Rightarrow \sum_{e \in p_1} D_e(f^*_e) > \sum_{e \in p_2} D_e(f^*_e) \textbf{ or } D_{p_1}(f^*) > D_{p_2}(f^*)\,.$$

Nash Equilibrium

* Thus,

- $\circ$ f^* is optimal for NCP

$\Leftrightarrow$ f^* does not have descent direction

$\Leftrightarrow$ $\forall i \in \{1, \ldots, k\}$; and $p_1, p_2 \in \mathcal{P}_i$ s.t. $f_{p_1} > 0$, then

$$\sum_{e \in p_1} D_e(f_e^*) \leq \sum_{e \in p_2} D_e(f_e^*)$$

i.e. $D_{p_1}(f^*) \leq D_{p_2}(f^*)$,

$\Leftrightarrow$ f^* is Nash Equilibrium for (G, r) with delay $(D_e(\,\cdot\,))$.

- Next,
  - $\circ$ Use the above characterization to compute bound on $\rho(G, r, D)$.

Nash Equilibrium

- Suppose, $(D_e(\ \cdot\ ))$ satisfies property

$$x \cdot D_e(x) \leq \alpha \int_0^x D_e(t)dt \ ; \ \alpha \geq 1 \ .$$

Then, $\rho(G, r, D) \leq \alpha$.

Proof.

$$\begin{aligned} C(f^N) &= \sum_e D_e(f_e^N) f_e^N \\ &\leq \alpha \sum_e \int_0^{f_e^N} D_e(t)dt \ = \ \alpha \sum_e h_e(f_e^N) \\ &\leq \alpha \sum_e h_e(f_e^*) \ \leq \ \alpha \sum_e D_e(f_e^*) f_e^* \\ &= \alpha C(f^*) \\ \Rightarrow \rho(G, r, D) &= \frac{C(f^N)}{C(f^*)} \leq \alpha \qquad \square \end{aligned}$$

Nash Equilibrium

- If delay is linear function, then
  - $\alpha = 2$ works

  $\rightarrow \rho(G, r, D) \leq 2$.
- Thus, penalty of greedy performance
  - No more than twice optimal delay when delay is linear
- **Theorem. [Roughgarden-Tardos]** For any strictly increasing, nonnegative delay D,
  - Let f^N be any Nash Equilibrium for (G, r, D), and
  - f^* be the optimal solution for $(G, 2r, D)$, then

$$\sum_e D_e(f_e^N) f_e^N \leq \sum D_e(f_e^*) f_e^* \ .$$

$\rightarrow$ Double the capacity of network !

Secure Congestion Control

- Congestion control: two key parts
  - User algorithm: TCP
  - Network/router algorithm: Queue-management

- Security
  - Prevention of user misbehavior or misuse of TCP
  - Malicious router algorithm

- First, we'll talk about TCP misbehavior
  - Later, we talk about router algorithms

Secure Congestion Control I

- Misbehavior of user
  - User does not follow TCP, i.e.
    - not reducing its traffic when required by protocol
  - User can possibly hijack all bandwidth on its path when other users are well-behaved

  $\longrightarrow$ Need some mechanism to penalize malicious users

- Queue-management scheme can help
  - We'll see a simple scheme to prevent misbehavior of TCP source

  $\longrightarrow$ Choke algorithm

Choke Algorithm

- Consider a simple setup:

- TCP users: adapt rate according to packet drop
- Malicious user: does not adapt its rate, sends data at very high rate
- Fair share: divide C equally among all users
    - ◦ If everyone followed TCP, it would happen
    - ◦ But, we've a malicious user!
- Simple solution: implement fairness at routers (in network)
    - ◦ Too much data-keeping and hence not feasible

  $\rightarrow$ Need a simple fair-mechanism

Choke

- Choke: features
  - Queue-management algorithm that punishes a flow for sending a lot of data
  - Thus, prevents malicious user from taking all bandwidth
  - Simple and implementable

- Choke: mechanism
  - Every time a packet arrives, draw another packet from queue at random
  - If their id match: drop both
  - Or else, drop arriving packet with probability proportional to queue size

Choke

- Result:
  - Choke prevents any one source from eating up more than 50% of bandwidth
  - One can show that (using fluid model) it is no more than 26%

- Better Choke:
  - In absence of any malicious user, we want it to be like TCP and RED
  - i.e., drop each incoming packet with probability proportional to the queue size

$\rightarrow$ Change choke so as to achieve this

Congestion Control II

- If malicious user
  - ◦ Prevention by penalty mechanism at router
- What if router is malicious, e.g.
  - ◦ Dropping few extra packets often enough
  - → Cause all users to operate in "low" rate TCP regime
- How to combat against it?
  - ◦ Well, greedy option is not to react
  - ◦ But this will totally ruin the performance
  - ◦ Can one do better?
    - – when all routers are okay, algorithm should be TCP
    - – else, not much performance degradation

Congestion Control II

- Essentially, is it possible to detect "malicious" packet drops?
- Malicious router can not drop most of the packet as
  - Otherwise, routing algorithm will naturally change route based on feedback
- Router can not drop packet by checking identity of all flows
  - Because, there are too many flows
  - Hence, drops are like "random"
- Drops due to congestion are usually many for the same flow
  - Hence, checking if more than half of packets dropped in last window is good check

Congestion Control II

- TCP*
  - When drop happens, user does not receive ACK
    - if too many packets dropped in past window then standard TCP
    - else, don't decrease windowsize
  - Use of the above information in clever manner can lead to better performance

- In summary,
  - TCP* can help protect against few malicious routers
  - Choke can help protect against few malicious users

- What if there are too many malicious users or routers ?

Next Set of Topics

- Guests speakers will cover topics on
  - Use of cryptographic tools for network security, e.g.
    - Light-weight email encryption
    - by Ben Adida (May 1 and 3)
  - Network security and Internet architecture
    - Thoughts and views
    - by Dave Clark (May 8 and 10)
  - Prevention of Unwanted traffic and malicious users
    - System solutions
    - by Dina Katabi (May 15)