US 20100004982A1

(19) **United States**

(12) **Patent Application Publication**

Yacobi et al.

(10) **Pub. No.: US 2010/0004982 A1**

(43) **Pub. Date: Jan. 7, 2010**

(54) **QUANTIFYING TRUST IN COMPUTING NETWORKS**

(75) Inventors: **Yacov Yacobi**, Mercer Island, WA (US); **Jim Kajiya**, Duvall, WA (US)

Correspondence Address:
MICROSOFT CORPORATION
ONE MICROSOFT WAY
REDMOND, WA 98052 (US)

(73) Assignee: **MICROSOFT CORPORATION**, Redmond, WA (US)

(21) Appl. No.: **12/264,253**

(22) Filed: **Nov. 4, 2008**

Related U.S. Application Data

(60) Provisional application No. 61/078,068, filed on Jul. 3, 2008, provisional application No. 61/094,861, filed on Sep. 5, 2008.

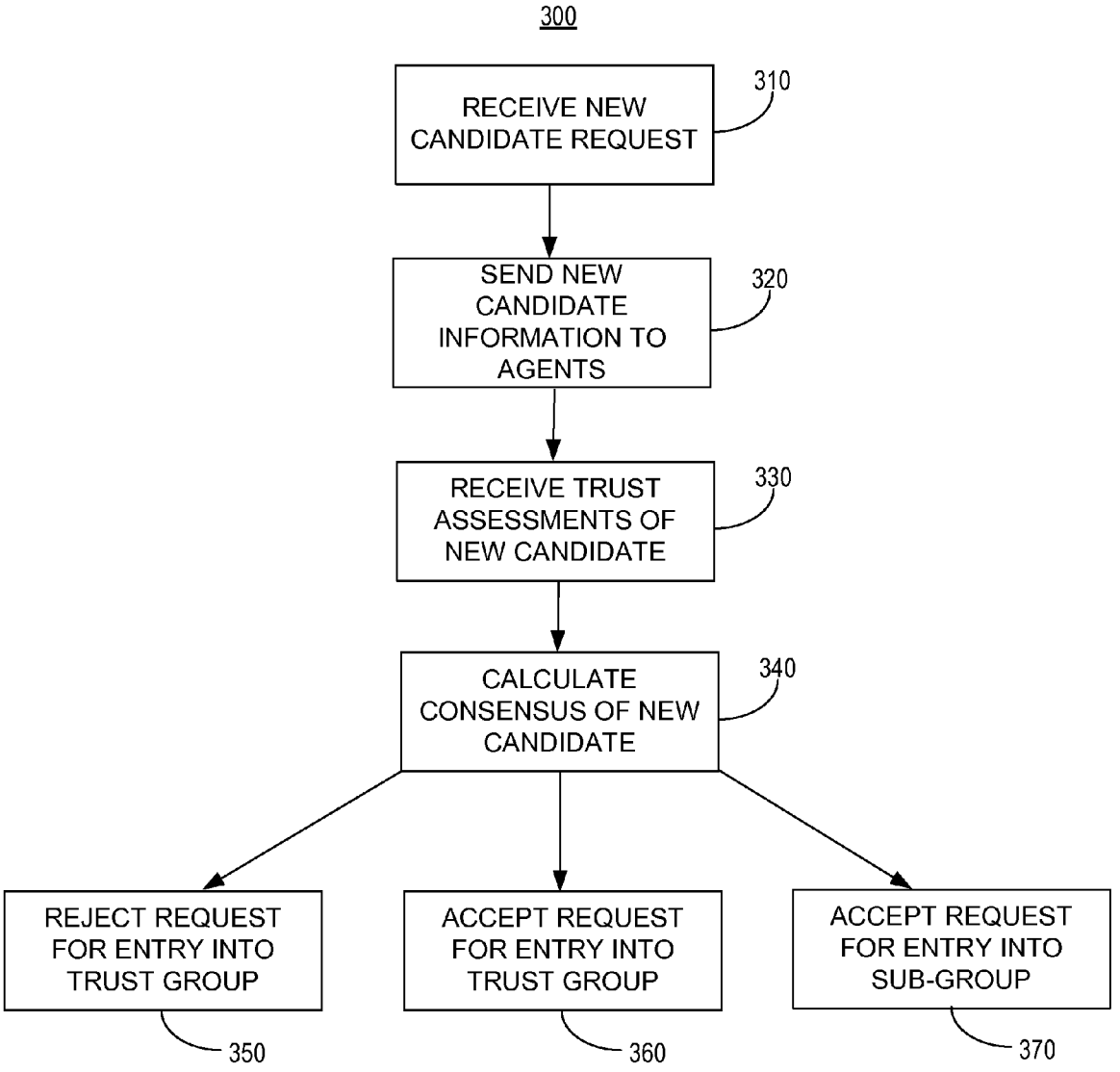
Publication Classification

(51) **Int. Cl.**
G06Q 10/00 (2006.01)

(52) **U.S. Cl.** **705/11**

(57) **ABSTRACT**

Method for calculating a trust value of an agent in a computing network. In one implementation, the method may include receiving information pertaining to a first agent's previous actions, quantifying a discrepancy between an expected behavior and an actual behavior of the first agent during the first agent's previous actions, and determining the trust value of the first agent based on the quantified discrepancy.



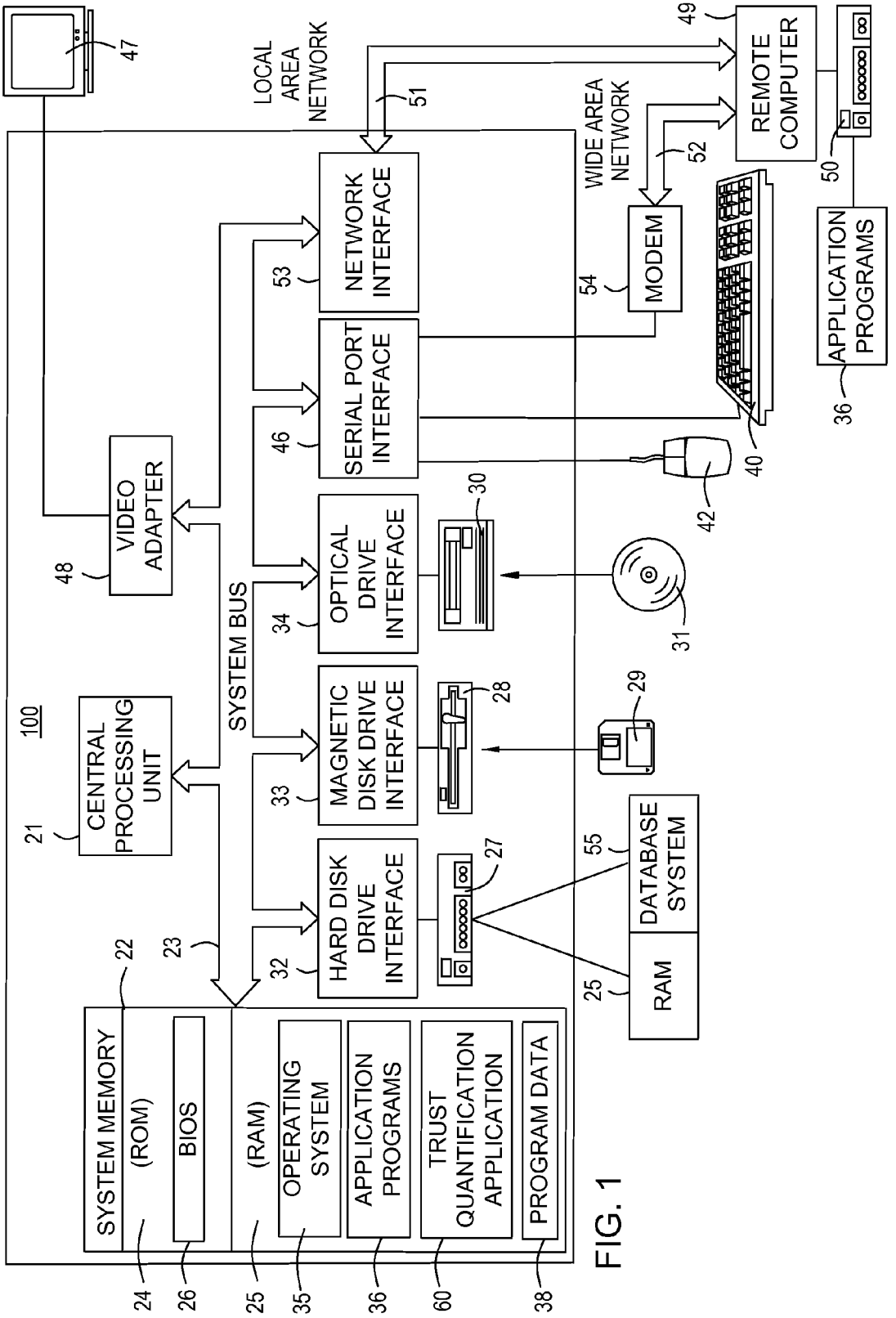

100
21
CENTRAL PROCESSING UNIT
SYSTEM BUS
LOCAL AREA NETWORK
WIDE AREA NETWORK
REMOTE COMPUTER
49
51
52
50
47
48
VIDEO ADAPTER
NETWORK INTERFACE
53
SERIAL PORT INTERFACE
46
MODEM
54
OPTICAL DRIVE INTERFACE
34
MAGNETIC DISK DRIVE INTERFACE
33
HARD DISK DRIVE INTERFACE
32
23
22
SYSTEM MEMORY
(ROM)
BIOS
(RAM)
OPERATING SYSTEM
APPLICATION PROGRAMS
TRUST QUANTIFICATION APPLICATION
PROGRAM DATA
24
26
25
35
36
60
38
RAM
DATABASE SYSTEM
55
25
27
28
29
30
31
40
42
APPLICATION PROGRAMS
36

FIG. 1

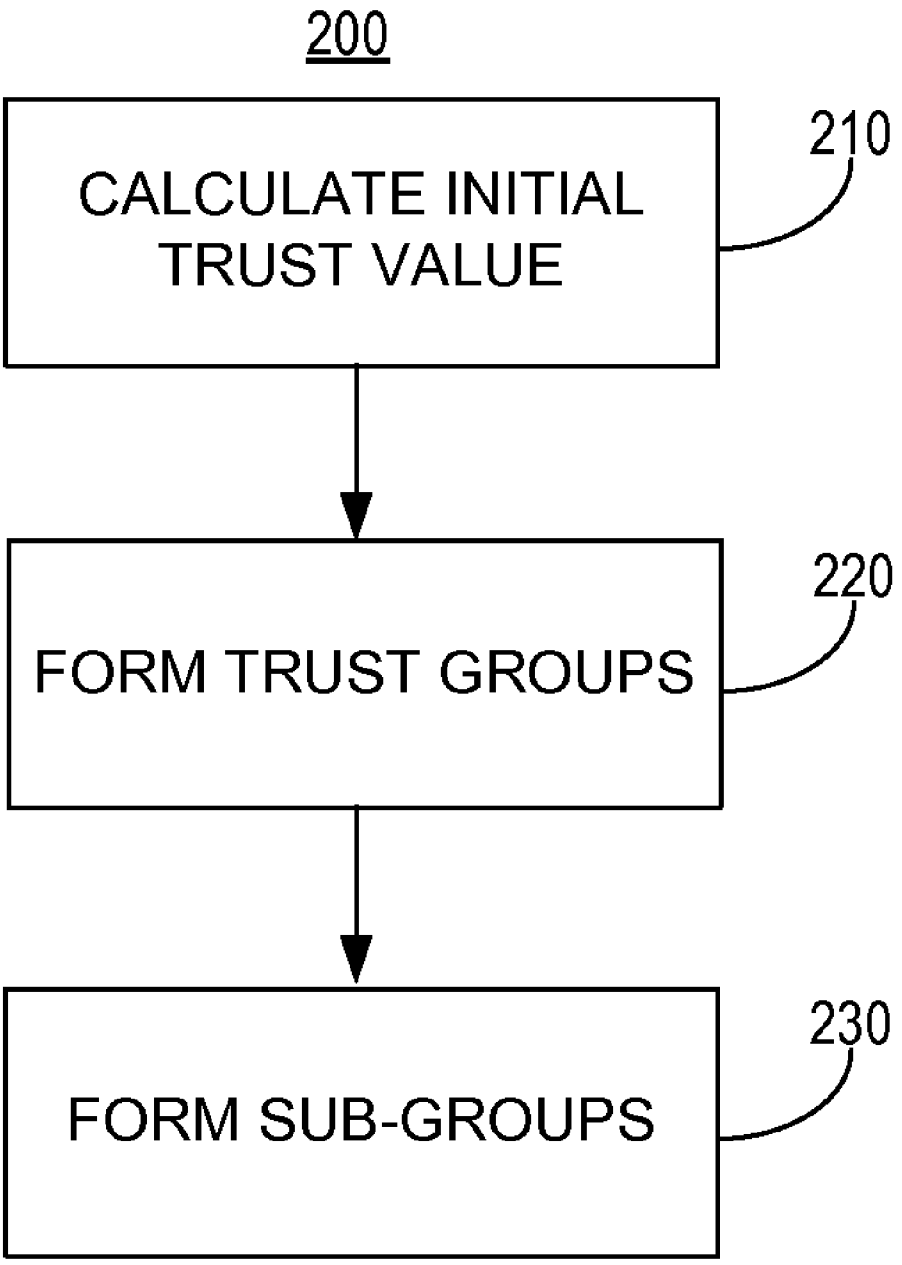

200
CALCULATE INITIAL TRUST VALUE
210
FORM TRUST GROUPS
220
FORM SUB-GROUPS
230

FIG. 2

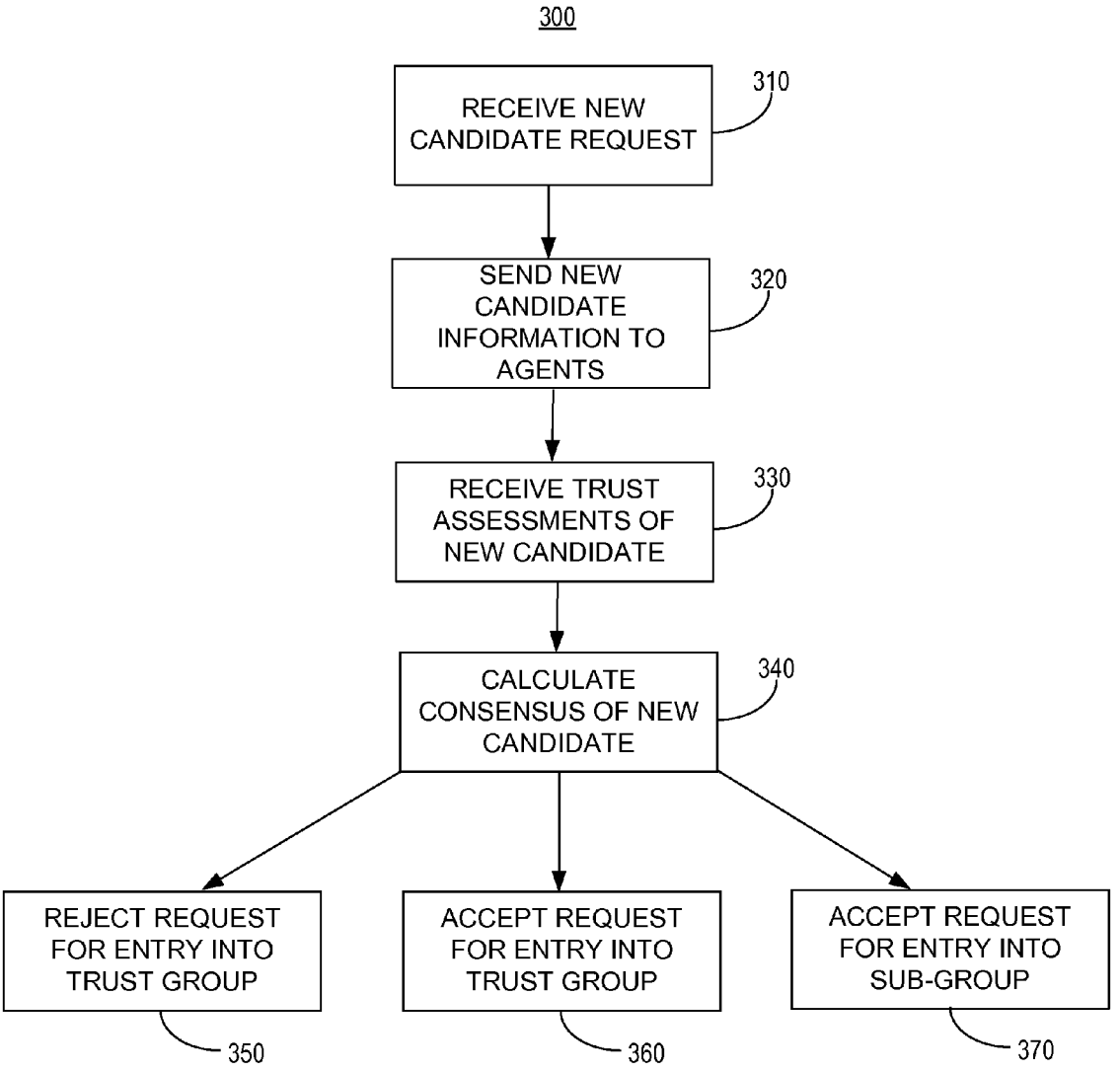

300
RECEIVE NEW CANDIDATE REQUEST
310
SEND NEW CANDIDATE INFORMATION TO AGENTS
320
RECEIVE TRUST ASSESSMENTS OF NEW CANDIDATE
330
CALCULATE CONSENSUS OF NEW CANDIDATE
340
REJECT REQUEST FOR ENTRY INTO TRUST GROUP
350
ACCEPT REQUEST FOR ENTRY INTO TRUST GROUP
360
ACCEPT REQUEST FOR ENTRY INTO SUB-GROUP
370

FIG. 3

QUANTIFYING TRUST IN COMPUTING NETWORKS

BACKGROUND

[0001] This application claims priority to U.S. provisional patent application Ser. No. 61/078,068, filed Jul. 3, 2008, titled METHOD FOR QUANTIFYING TRUST, which is incorporated herein by reference.

[0002] This application claims priority to U.S. provisional patent application Ser. No. 61/094,861, filed Sep. 5, 2008, titled TRUST AND COLLABORATION, which is incorporated herein by reference.

[0003] As digital communications, networks, and transactions increase, the need became apparent for ways in which computer users could "trust" each other. Digital trust systems, such as Public Key Infrastructure (PKI), build trust hierarchies, such as the "Web of Trust", so that users could securely communicate with each other, authenticate the identities of each other, and determine the integrity of the messages received from each other. In order to establish the trust characteristic of each user, trust systems rely on the certification or revocation of a user by one or more trusted third parties. However, the certification or revocation of a user does not explain what "trust" exactly is or how to quantify it. Instead, each user's trust characteristic is defined in a binary trust form consisting of 1 (trustworthy) or 0 (not trustworthy). The trust characteristics may be established through logical rules of inference from the certification or revocation actions of one or more trusted third parties. The trusted third parties may determine that a user is trustworthy if he has a certain set of credentials and if he complies with a local security policy, but this method does not allow the third party to explain why the user is trusted or how much the third party trusts him.

SUMMARY

[0004] Described herein are implementations of various technologies for quantifying trust in a computing network. In one implementation, a computer program may be configured to establish a trust value for an agent (user) in a computing network. The trust value may be established by comparing an agent's expected behavior to his actual behavior in past transactions. The computer program may analyze the "gap" or discrepancy between the agent's expected behavior and his actual behavior to establish an initial trust value. Trust values for each agent in the computing network may be evaluated using a similar type of analysis.

[0005] After trust values are established for each agent, the computer program may form one or more trust groups, or cliques, containing agents with similar trust values in each other. Each trust group may be created such that trust values of each agent may be within a specific tolerance ("q") of each other. If an agent's trust value is not within the specified tolerance ("q") of other agents in a trust group, the computer program may split the trust group into one or more sub-groups such that each agent within the sub-group may have similar trust values in each other but with a smaller tolerance than that of the larger trust group. If the agent's trust value is not within the specified tolerance ("q") or does not have a similar trust value with other agents in sub-groups, he may be rejected from the small and large trust groups.

[0006] After trust groups and sub-groups of agents have been created, new candidates, or new agents, may be granted entry into a trust group or sub-group based on an evaluation performed by each agent in the trust group or sub-group. Each agent within the trust group or sub-group may then assess their trust in the new candidate and create his own trust value for the new candidate. The trust value assigned to the candidate may be quantified into a value between zero and one. If the candidate meets the trust group's trust requirement, i.e., the trust value of the candidate is within the trust group's tolerance "q", he may be granted access into the trust group. If the candidate does not meet the trust requirement of each agent in a trust group, he may be accepted into a sub-group or he may be rejected from the trust group altogether.

[0007] The above referenced summary section is provided to introduce a selection of concepts in a simplified form that are further described below in the detailed description section. The summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used to limit the scope of the claimed subject matter. Furthermore, the claimed subject matter is not limited to implementations that solve any or all disadvantages noted in any part of this disclosure.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] FIG. **1** illustrates a schematic diagram of a computing system in which the various techniques described herein may be incorporated and practiced.

[0009] FIG. **2** illustrates a flow diagram of a method for initially quantifying trust and grouping agents with similar trust values in a computing network in accordance with one or more implementations of various techniques described herein.

[0010] FIG. **3** illustrates a flow diagram of a method for evaluating the trust characteristics of a new agent in accordance with one or more implementations of various techniques described herein.

DETAILED DESCRIPTION

[0011] In general, one or more implementations described herein are directed to quantifying trust in an agent and grouping agents based on their trust values. One or more implementations of various techniques for quantifying trust will be described in more detail with reference to FIGS. **1-3**

[0012] Implementations of various technologies described herein may be operational with numerous general purpose or special purpose computing system environments or configurations. Examples of well known computing systems, environments, and/or configurations that may be suitable for use with the various technologies described herein include, but are not limited to, personal computers, server computers, hand-held or laptop devices, multiprocessor systems, microprocessor-based systems, set top boxes, programmable consumer electronics, network PCs, minicomputers, mainframe computers, distributed computing environments that include any of the above systems or devices, and the like.

[0013] The various technologies described herein may be implemented in the general context of computer-executable instructions, such as program modules, being executed by a computer. Generally, program modules include routines, programs, objects, components, data structures, etc. that performs particular tasks or implement particular abstract data types. The various technologies described herein may also be implemented in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network, e.g., by hardwired

links, wireless links, or combinations thereof. In a distributed computing environment, program modules may be located in both local and remote computer storage media including memory storage devices.

[0014] FIG. **1** illustrates a schematic diagram of a computing system **100** in which the various technologies described herein may be incorporated and practiced. Although the computing system **100** may be a conventional desktop or a server computer, as described above, other computer system configurations may be used.

[0015] The computing system **100** may include a central processing unit (CPU) **21**, a system memory **22** and a system bus **23** that couples various system components including the system memory **22** to the CPU **21**. Although only one CPU is illustrated in FIG. **1**, it should be understood that in some implementations the computing system **100** may include more than one CPU. The system bus **23** may be any of several types of bus structures, including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. By way of example, and not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnect (PCI) bus also known as Mezzanine bus. The system memory **22** may include a read only memory (ROM) **24** and a random access memory (RAM) **25**. A basic input/output system (BIOS) **26**, containing the basic routines that help transfer information between elements within the computing system **100**, such as during start-up, may be stored in the ROM **24**.

[0016] The computing system **100** may further include a hard disk drive **27** for reading from and writing to a hard disk, a magnetic disk drive **28** for reading from and writing to a removable magnetic disk **29**, and an optical disk drive **30** for reading from and writing to a removable optical disk **31**, such as a CD ROM or other optical media. The hard disk drive **27**, the magnetic disk drive **28**, and the optical disk drive **30** may be connected to the system bus **23** by a hard disk drive interface **32**, a magnetic disk drive interface **33**, and an optical drive interface **34**, respectively. The drives and their associated computer-readable media may provide nonvolatile storage of computer-readable instructions, data structures, program modules and other data for the computing system **100**.

[0017] Although the computing system **100** is described herein as having a hard disk, a removable magnetic disk **29** and a removable optical disk **31**, it should be appreciated by those skilled in the art that the computing system **100** may also include other types of computer-readable media that may be accessed by a computer. For example, such computer-readable media may include computer storage media and communication media. Computer storage media may include volatile and non-volatile, and removable and non-removable media implemented in any method or technology for storage of information, such as computer-readable instructions, data structures, program modules or other data. Computer storage media may further include RAM, ROM, erasable programmable read-only memory (EPROM), electrically erasable programmable read-only memory (EEPROM), flash memory or other solid state memory technology, CD-ROM, digital versatile disks (DVD), or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by the computing system **100**. Communication media may embody computer readable instructions, data structures, program modules or other data in a modulated data signal, such as a carrier wave or other transport mechanism and may include any information delivery media. The term "modulated data signal" may mean a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media may include wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media. Combinations of any of the above may also be included within the scope of computer readable media.

[0018] A number of program modules may be stored on the hard disk **27**, magnetic disk **29**, optical disk **31**, ROM **24** or RAM **25**, including an operating system **35**, one or more application programs **36**, a trust quantification application **60**, program data **38**, and a database system **55**. The operating system **35** may be any suitable operating system that may control the operation of a networked personal or server computer, such as Windows® XP, Mac OS® X, Unix-variants (e.g., Linux® and BSD®), and the like. The trust quantification application **60** will be described in more detail with reference to FIG. **2** in the paragraphs below.

[0019] A user may enter commands and information into the computing system **100** through input devices such as a keyboard **40** and pointing device **42**. Other input devices may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices may be connected to the CPU **21** through a serial port interface **46** coupled to system bus **23**, but may be connected by other interfaces, such as a parallel port, game port or a universal serial bus (USB). A monitor **47** or other type of display device may also be connected to system bus **23** via an interface, such as a video adapter **48**. In addition to the monitor **47**, the computing system **100** may further include other peripheral output devices such as speakers and printers.

[0020] Further, the computing system **100** may operate in a networked environment using logical connections to one or more remote computers The logical connections may be any connection that is commonplace in offices, enterprise-wide computer networks, intranets, and the Internet, such as local area network (LAN) **51** and a wide area network (WAN) **52**.

[0021] When using a LAN networking environment, the computing system **100** may be connected to the local network **51** through a network interface or adapter **53**. When used in a WAN networking environment, the computing system **100** may include a modem **54**, wireless router or other means for establishing communication over a wide area network **52**, such as the Internet. The modem **54**, which may be internal or external, may be connected to the system bus **23** via the serial port interface **46**. In a networked environment, program modules depicted relative to the computing system **100**, or portions thereof, may be stored in a remote memory storage device **50**. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

[0022] It should be understood that the various technologies described herein may be implemented in connection with hardware, software or a combination of both. Thus, various technologies, or certain aspects or portions thereof, may take the form of program code (i.e., instructions) embodied in tangible media, such as floppy diskettes, CD-ROMs, hard drives, or any other machine-readable storage medium

wherein, when the program code is loaded into and executed by a machine, such as a computer, the machine becomes an apparatus for practicing the various technologies. In the case of program code execution on programmable computers, the computing device may include a processor, a storage medium readable by the processor (including volatile and non-volatile memory and/or storage elements), at least one input device, and at least one output device. One or more programs that may implement or utilize the various technologies described herein may use an application programming interface (API), reusable controls, and the like. Such programs may be implemented in a high level procedural or object oriented programming language to communicate with a computer system. However, the program(s) may be implemented in assembly or machine language, if desired. In any case, the language may be a compiled or interpreted language, and combined with hardware implementations.

[0023] FIG. **2** illustrates a flow diagram of a method **200** for initially quantifying trust in an agent in a computing network in accordance with one or more implementations of various techniques described herein. The following description of flow diagram **200** is made with reference to computing system **100** of FIG. **1** in accordance with one or more implementations of various techniques described herein. It should be understood that while the operational flow diagram **200** indicates a particular order of execution of the operations, in some implementations, certain portions of the operations might be executed in a different order. In one implementation, the method for quantifying trust may be performed by the trust quantification program **60**.

[0024] At step **210**, the trust quantification program **60** may calculate a trust value for each agent present in the computing network. In one implementation, the trust value, or a-priori trust value, may be generated based on previous knowledge about an agent. Such previous knowledge may include information about the agents expected behavior and his corresponding actual behavior in past transactions. The "gap" or discrepancy between the expected and actual behavior of the agent may be quantified and normalized per bit to create a discrepancy value in the interval between 0 and 1. In one implementation, the discrepancy value may correspond to a conditional entropy that may be normalized to a value in the interval between 0 and 1. The discrepancy value, or average uncertainty, of the agent based on another agent's previous knowledge about the agent. Alternatively, symmetric measure based on conditional entropy may be used to determine the discrepancy value (D(x,y)) such that D(x,y)=H_x(y)/H(x,y)+H_y(x)/H(x,y).

[0025] For example, suppose that x, y, and z are random variables or random agents in a network. If x is an ideal agent that does not lie or err, y is an agent in a well defined role whose trustworthiness is being evaluated, and z is a random variable describing the evaluator's trustworthiness in x. The average uncertainty or discrepancy value of y's trustworthiness given x's trustworthiness may be represented as a normalized conditional entropy of y given x's previous knowledge of y. The average uncertainty or discrepancy value of y given x may be denoted as $H_x(y)$. Since x may be considered to be an ideal agent that does not lie or err, the absolute trustworthiness value of y may then be determined by subtracting the discrepancy value from **1**, such that y's absolute trustworthiness, or t_y, may be defined as $t_y=1-H_x(y)$. Since z is the evaluator's determination of how agent y should behave, agent z's trustworthiness in agent y, or t_{zy}, may be defined as $t_{zy}=1-H_z(y)$.

[0026] For example, in an online marketplace where computer users may buy or sell merchandise on the Internet, previous knowledge pertaining to a purchaser (agent Y) may be used to generate an initial trust value for the purchaser. The purchaser's expected behavior and his corresponding actual behavior may correspond to his promise to pay a specified amount and the actual amount he paid in previous transactions. In one implementation, information pertaining to the purchaser's previous transactions may be provided by a credit card company (agent X). The discrepancy between the purchaser's promise to pay and his actual payment may be used to create a discrepancy value, or average uncertainty that the credit card company may have in the purchaser. The average uncertainty of the credit card company in the purchaser may be defined as $H_{credit\text{-}card\text{-}co}$(purchaser). The trust value of the purchaser may then be determined by subtracting the discrepancy value from 1, such that the credit card company's trustworthiness in the purchaser may be defined as $t_{credit\text{-}card\text{-}co\text{-}purchaser}=1-H_{credit\text{-}card\text{-}co}$(purchaser).

[0027] In one implementation, the trust quantification application **60** may help gather the input data for trust evaluation. For example, the trust quantification application **60** may detect truth-in-ads discrepancies (the ad promised price x, and the buyer was charged y>x) made by merchants. Furthermore, the trust quantification application **60** may detect discrepancies in a revocation list, such as complaints about truth-in-ads, and it may gather input data about the trustworthiness of revocation authority.

[0028] Although the above example based the gap value of the purchaser on his previous transaction's promise to pay and his subsequent actual payment, it should be noted that the gap value may be based on one or more other factors, such as information pertaining to the date in which the purchaser paid, the manner in which he paid it, and/or combinations of the like. Similarly, previous knowledge pertaining to a merchant (agent) may be used to generate an initial trust value for the merchant. The merchant's expected behavior and his corresponding actual behavior may correlate to his advertised price on a product and the actual amount he charged for the product in previous transactions. In one implementation, information pertaining to the purchaser's and/or the merchant's previous transactions may be provided by one or more credit card companies, banks, peer reviews, or the like.

[0029] At step **220**, the trust quantification application **60** may group agents with trust values in each other into a trust group. The group of agents within a trust group may have trust values within a specified tolerance 'q1' of each other. In one implementation, the specified tolerance 'q1' may correspond to a high trust value. The trust values of each agent on each other may naturally converge to the extremes such that the result is the formation of maximal-trust trust groups among peers. For example, in extremely large trust groups, trust values of each agent in each other may converge to the extremes such that each agent may be deemed as either trustworthy or not. At step **230**, the trust quantification application **60** may divide or split the trust group formed at step **220** into one or more sub-groups Splits may occur when a new agent does not meet the trust value requirement (specified tolerance 'q1') for the whole group but it may meet the trust requirement for a subgroup. If the new agent meets the trust requirements for a subgroup, the trust quantification application **60**

may decide to split the group and accept the new agent into a subgroup as opposed to rejecting the new agent altogether. In one implementation, the sub-groups may contain agents with similar trust values in each other but with a smaller tolerance 'q2' than those agents in the trust group. The agents in the sub-group may be considered to "trust" each other more than those agents in the original trust group.

[0030] In one implementation, the trust quantification application **60** may split the larger trust group into two or more sub-groups based on one or more economic utility functions. Economic utility functions may be used to maximize the utility or purpose for establishing trust groups. Economic utility functions may measure the relative satisfaction of an agent based on a correlation between an agent's economic behavior and his desire for consuming various goods and services. Examples of some economic utility functions may include a trust group of sellers and buyers that may wish to maximize overall market share, a user who may wish to maximize the number of features (plug-in modules) in his machine, assigning various non-uniform weights to various features, or other types of utility functions. Unfortunately, all economic utility functions may not be able to exist in harmony with each other because there may be interdependency as well as conflicts between different economic utilities. Therefore, an agent may switch opportunistically among its cliques when performing distinct tasks that may depend on his economic utility functions. In one implementation, the trust values change as sub-groups grow, therefore, the trust quantification application **60** may have to verify each agent's acceptance into every sub-group. In one implementation, the trust quantification application **60** may gradually evaluate each agent's trustworthiness and allow some tolerance $q>0$ in the acceptance criteria and then a limit on the clique size may be quantified as n. If the trust quantification application **60** evaluates the agents and cliques instantaneously, there may not be a limit on the clique size. The overall uncertainty may be represented as $O(q\cdot\exp(n))$, hence n may most likely be small, and $q<<\exp(-n)$. In that case, the acceptance criteria may require that every pair of agents may have close to mutual maximal trust ($>1-q$) and that the consensus among agents in the sub-clique may be allowed to oscillate within tolerance $\delta=O(q\cdot\exp(n))$, away from the maximal trust.

[0031] FIG. **3** illustrates a flow diagram of a method **300** for evaluating the trust characteristics of a new agent in a computing network in accordance with one or more implementations of various techniques described herein. The following description of flow diagram **300** is made with reference to computing system **100** of FIG. **1** in accordance with one or more implementations of various techniques described herein. It should be understood that while the operational flow diagram **300** indicates a particular order of execution of the operations, in some implementations, certain portions of the operations might be executed in a different order. In one implementation, the method for evaluating the trust characteristics of a new agent may be performed by the trust quantification program **60**.

[0032] At step **310**, the trust quantification application **60** may receive a request from a new candidate to gain entry into one or more trust groups or sub-groups. For example, with respect to the purchaser/merchant example described in FIG. **2**, a new candidate may be a new purchaser or merchant in a computing network.

[0033] At step **320**, the trust quantification application **60** may provide information pertaining to the new candidate to each agent in the trust group to receive a consensus trust evaluation on the new candidate. In the preceding example, the new purchaser may base its initial trust value on information obtained from a credit card company. Each agent of a trust group in the computing network may be provided the new purchaser's initial trust value which may be determined using information from the credit card company. Each agent may then evaluate the new purchaser's discrepancy value as described in step **210** and assess his own trust value for the new purchaser. In one implementation, each agent in a trust group or sub-group may assess their trust value in the new candidate based on their trust value in the credit card company that assigned the initial trust value in the candidate.

[0034] In another implementation, masters or users may program their agents to evaluate new candidates based on his initial trust value and the trust value given by other agents within the trust group. In another implementation, agents may not have the space to hold trust data, so they may request for trust values from their masters. In some implementations, agents may not even analyze the candidate's trust value; instead, they may request their masters to provide conclusions.

[0035] At step **330**, the trust quantification application **60** may receive from each agent in the computing network a trust value assessment of the new candidate. Based on the received trust values of the agents within the trust group or sub-groups, the trust quantification application **60** may determine the consensus of all of the agents in the trust group at step **340**.

[0036] At step **340**, the trust quantification application **60** may determine a consensus value for the new candidate. In one implementation, at a discrete time ($\tau=0$), the trust values may be defined as $t_{i0}(0)$ where i may represent an agent within the trust group. For each agent i, the consensus values at time ($\tau+1$) may be. In one implementation, the process for calculating the consensus values may be iterated until a stable consensus is reached. Here, the weights may be uniform for each value; however, in some implementations, the weights may not be uniform yet the sum of all of the weights may be equal to one.

[0037] In one implementation, an estimation error in the values of a stable consensus trust may be defined as $\delta(n, N)=O(\exp(n-N))$where N is the size of message space and n is the total number of agents. Each random variable may be defined over a message space. For example, when evaluating a merchant and his truth in ads, each data point ("message") may be an advertisement and the actual price charged as reported by many users. When a trust group grows gradually, adding one agent at a time, then the error in the trust quantification application **60** estimations may become $\exp(n-N)$, where n=number of agents in the group, and N=the number of messages in the message space used to evaluate the gaps. However, when evaluations are instantaneous the error may be much smaller, ~2 $\exp(-N)$. Based on the consensus, the trust quantification application **60** may reject the new candidate's entry into a trust group (step **350**), accept his entry into a trust group (step **360**), or accept his entry into a sub-group (step **370**).

[0038] At step **350**, the trust quantification application **60** may reject the new candidate's entry into a trust group based on the consensus values received at step **340**. In one implementation, rejection into a trust group may indicate to agents in a network that the candidate is not trustworthy. The new candidate may be rejected from a trust group if one or more

members of the trust group do not have trust values within a specified tolerance for the new candidate.

[0039] At step **360**, the trust quantification application **60** may accept the new candidate's entry into a trust group. In one implementation, if each member of a trust group has high trust value in the candidate, and the candidate thus has a high trust value in each member of the trust group, then the candidate may be accepted into the trust group. Acceptance into the trust group may indicate to agents in a network that the candidate is trustworthy within a certain degree. The trust value between each member in a trust group may be within a specified tolerance.

[0040] At step **370**, the trust quantification application **60** may accept the new candidate's entry into a sub-group. In one implementation, if the new candidate is accepted into a sub-group, then the trust quantification application **60** may split the larger trust group into two or more sub-groups based on one or more economic utility functions as described in FIG. **2**.

[0041] In addition to the example of quantifying trust in an online purchaser and merchant relationship, the trust quantification application **60** may apply the same method **200** and method **300** to various scenarios such as trust between humans and certification authority, Tit-for-tat strategy in the iterative prisoner's dilemma game, the inter-relations among software modules in a system, the stock market, and other trust oriented applications.

[0042] Although the subject matter has been described in language specific to structural features and/or methodological acts, it is to be understood that the subject matter defined in the appended claims is not necessarily limited to the specific features or acts described above. Rather, the specific features and acts described above are disclosed as example forms of implementing the claims.

What is claimed is:

1. A method for calculating a trust value of an agent in a computing network, comprising:
- receiving information pertaining to a first agent's previous actions;
- quantifying a discrepancy between an expected behavior and an actual behavior of the first agent during the first agent's previous actions; and
- determining the trust value of the first agent based on the quantified discrepancy.

2. The method of claim **1**, wherein the discrepancy corresponds to a conditional entropy of the first agent based a second agent's experience in dealing with the first agent.

3. The method of claim **1**, wherein the discrepancy is characterized as a gap value.

4. The method of claim **1**, wherein quantifying the discrepancy comprises normalizing the discrepancy in an interval between 0 and 1.

5. The method of claim **4**, wherein determining the trust value of the first agent comprises subtracting the normalized discrepancy from 1.

6. The method of claim **1**, wherein the expected behavior comprises the first agent's promise to pay a specified amount in a previous transaction.

7. The method of claim **6**, wherein the actual behavior comprises an actual amount the first agent paid in the previous transaction.

8. The method of claim **1**, wherein the information pertaining to the first agent's previous actions comprises information regarding previous transactions provided by a credit card company.

9. The method of claim **1**, wherein the expected behavior comprises the first agent's promise to sell an item at a specified amount in a previous transaction.

10. The method of claim **9**, wherein the actual behavior comprises an actual amount the first agent sold the item in the previous transaction.

11. A method for establishing trust groups in a computing network, comprising:
- receiving one or more trust values for each agent in a computing network;
- identifying one or more agents having trust values that differ within a first specified tolerance; and
- grouping the one or more agents into a trust group.

12. The method of claim **11**, wherein the trust group comprises a limit on the number of agents.

13. The method of claim **12**, further comprising splitting the trust group into sub-groups if the limit is exceeded.

14. The method of claim **11**, wherein the sub-groups comprise one or more agents having trust values that differ within a second specified tolerance.

15. The method of claim **11**, wherein the second specified tolerance is smaller than the first specified tolerance.

16. A method for granting a new agent entry into a trust group within a computing network, comprising:
- receiving a request for entry into the trust group from the new agent;
- sending information pertaining to the new agent to each member of the trust group;
- receiving a trust value from each member of the trust group; and
- forming a consensus of the trust values received from member of the trust group.

17. The method of claim **16**, further comprising: rejecting the request of the new agent for entry into one or more trust cliques if the consensus is below a predetermined value.

18. The method of claim **16**, further comprising: accepting the request of the new agent for entry into one or more trust cliques if the consensus is above a predetermined value.

19. The method of claim **16**, further comprising:
- splitting the trust group into two or more subgroups if the consensus is above a predetermined value and a limit on the number of members in the trust group has been exceeded; and
- accepting the request of the new agent for entry into a subgroup.

20. The method of claim **16**, wherein the trust group is split based on one or more economic utility functions.

* * * * *